AN EMPIRICAL STUDY OF THE RELATIONSHIP BETWEEN STATIC SOFTWARE COMPLEXITY METRICS AND DYNAMIC MEASUREMENTS OF PASCAL AND C PROGRAMS

By

KEITH EUGENE MOLL

Bachelor of Science in Arts and Sciences

Oklahoma State University

Stillwater, Oklahoma

1987

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 1989

AN EMPIRICAL STUDY OF THE RELATIONSHIP
BETWEEN STATIC SOFTWARE COMPLEXITY
METRICS AND DYNAMIC MEASUREMENTS
OF PASCAL AND C PROGRAMS

Thesis Approved:

M. Samadzadeh-H.
Thesis Adviser

R. E. Hale

Marilyn G. Kletke

Norman N. Durham
Dean of the Graduate College

PREFACE

Over the past 10 to 15 years, several studies showing relationships among static complexity metrics have been performed. These include the number of lines of code, McCabe's cyclomatic complexity, Halstead's Software Science metrics, control flow metrics, and information flow metrics. Other studies have examined the relationships between static metrics and effort, clarity, productivity, quality, faults, and reliability. However, there have been very few studies that explore the relationship between static complexity metrics and dynamic measurements of programs. This exploratory, empirical study examines this relationship. The issues considered in this work include data collection procedures, the development of a counting strategy, the analysis of the static and dynamic measurements collected, and the examination of the significance between pairs of these measurements. A goal is to arrive at possible hypotheses to be tested in future, more extensive, controlled experiments. The results of this study show that there are significant correlations between some of the static and dynamic measurements.

ACKNOWLEDGMENTS

I owe a great deal of gratitude and appreciation to my major adviser Dr. Mansur H. Samadzadeh for his guidance, motivation, dedication, and valuable instruction during my thesis work. I especially wish to thank the Samadzadeh family who had a baby girl during the time that this work was being completed. Dr. Samadzadeh continued to spend endless hours reviewing my work and offering suggestions for further refinements.

I wish to thank my other committee members Drs. George E. Hedrick and Marilyn G. Kletke. Their time and efforts are greatly appreciated. I also wish to thank Dr. K. M. George for his guidance during the early part of my thesis work.

I also wish to thank Mr. Terry Johnson, Mr. Brendan Machado, Mrs. Farideh Samadzadeh, and Dr. Mansur Samadzadeh (again), the instructors of each of the classes for allowing me to help them evaluate the student programs used in this study.

I would also like to thank Mr. Changhyun Jo, Mr. James Alexander, Mr. Keith Lovelace, and Mrs. Beverly Dale for helping me collect the data that I needed. In addition, I would like to thank Mr. Eric Cloninger for proofreading this thesis.

Finally, but certainly not least, I wish to thank my parents, Mr. & Mrs. Kenneth G. Moll, Jr. It was their examples of hard work over the many years that gave me the inspiration and motivation to complete my graduate studies. I will be forever indebted to them for this.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The field of software engineering has existed for over 25 years [Goldberg86]. An increasingly important subfield of software engineering is the study of static and dynamic measurements of programs. Several studies on the relationship between various static software complexity metrics and effort [Walston77, Halstead77b, Basili83b, and Lind89], clarity [Gordon79], productivity and quality [Jones78], faults [Crawford85], and reliability [Lew88] have been carried out. However, few studies have been performed that explore the relationship between static software complexity metrics and dynamic measurements of programs [Shaw89 and Plattner81].

Walston and Felix [Walston77] discussed a method of programming project productivity estimation and presented results of other research into methods of measuring and estimating programming project duration, staff size, and computer cost. Their study used a data base of 60 completed software development projects with delivered source lines of code ranging from 4,000 to 467,000. Development efforts ranged from 12 to 11,758 man-months. The projects represented eight different high-level languages and 66 different computers. They identified five major parameters (productivity, schedule, cost, quality, and size) that can help programming project personnel make estimates

and derived the following formula for total effort, E, in man-months to develop a project

$$E = 5.2\ L^{0.91}$$

where L is thousands of lines of delivered source code. (Of course, to be able to estimate development effort of a project before it is completed, one would have to estimate L.) Walston and Felix also derived a productivity index based on twenty-nine variables that correlate highly with productivity. They concluded that the present approach to productivity estimation is far from being optimal and that further research is required to determine the potential of regression as an estimating tool.

Halstead [Halstead77b] refutes some of the conclusions of Walston and Felix. Specifically, he states that their formula for effort, which is given above, is not supported by their data. In the *Author's response* following Halstead's remarks, Walston and Felix explain that the range of delivered source lines of code, as stated in their paper, was in error and show that their formula is reasonable.

Basili, Selby, and Phillips [Basili83b] analyzed the relationship of Software Science metrics, cyclomatic complexity, and various standard program measures, such as lines of code, with effort and development errors. They collected data from a production FORTRAN environment and found that the metrics' correlations with actual effort were the strongest when the data came from individual programmers or certain validated projects. They concluded that no single metric correlated best with effort and that the number of revisions

appeared to correlate with development errors better than either Software Science metrics, cyclomatic complexity, or source lines of code.

Lind and Vairavan [Lind89] studied such software metrics as lines of code, Halstead's program length, and McCabe's cyclomatic complexity and their relationship with software development effort. They used real-time, commercial, medical imaging software written mostly in Pascal and FORTRAN in their study. They showed that significant correlations exist between total lines of code, cyclomatic complexity, and program length. They found that the conceptually simple complexity measures, such as lines of code, correlated with development effort just as well as other more complex measures.

Gordon [Gordon79], motivated by the rising cost of the development of quality software, assessed the clarity of programs--this, according to Gordon, can be used as a guideline to efficient software production and language development--as a function of the number and frequency of operators and operands. He studied ALGOL, FORTRAN, PL/I, COBOL, and Pascal programs. In order to minimize the effect of program impurities [Halstead77a], he expressed program effort as

$$E_p = (\hat{N} \log_2 n) / \hat{L}$$

where $\hat{N}$ is the estimated program length, n is the program vocabulary, and $\hat{L}$ is the estimated program level. He also expressed the estimated amount of mental effort required for comprehension of a program (clarity) as

$$E_c = V / \hat{L}$$

where V is the program volume and again $\hat{L}$ is the estimated program level. He concluded that E_p is a good estimator of the amount of effort required to

understand the initial version of a program, while E_c is a good estimator of the effort required to understand the revised versions of the program.

Jones [Jones78] investigated various definitions of units of measure as they relate to program quality and programmer productivity. He explained how some measures can be misleading. For example, when comparing programmer productivity across languages, the lines of code measure generously rewards Assembler language programmers while penalizing high-level language programmers: a high-level language requires fewer source statements (lines of code) to program a given function than does Assembler language. He subdivided program quality into the subcategories of defect prevention and defect removal. Jones explained productivity as a function of work units and cost units. In his analysis, he used what he calls a probability rectangle to describe the relationships quality and productivity have with various static measures.

Crawford, McIntosh, and Pregibon [Crawford85] studied the correlations between a set of static software measures and the number of faults and total estimated repair effort for a large, commercial software system. The subset of software they selected included 89 modules containing 2040 files with 516K deliverable source lines or 258K noncommentary lines of C code, all of which was new code. They used variants of the lines of code measure and McCabe's and Halstead's metrics in their study. They performed logarithmic transformations on the data to adjust the correlations between measures that shared an underlying relationship with a third variable such as size. All their static measures correlated strongly with each other. After they adjusted each

measure for the effect of size, the only significant correlations existed between McCabe's cyclomatic complexity (a count of the number of conditions for which the code segment **IF c1 AND c2 THEN** would have a count of two) and the number of decisions (essentially a count of the keywords identifying branches in a program, such as **IF** and **WHILE**) and between the total number of operators and total operands. Crawford *et al.* concluded that the number of noncommentary source lines, number of decisions per function, number of tokens per noncommentary source line, ratio of noncommentary to total source lines, operator vocabulary adjusted for size, and operand vocabulary adjusted for size accounted for most the variation in the number of code faults, effort to isolate errors, and effort to repair the errors. They derived models for the number of code faults, effort to isolate errors, and effort to repair the errors.

Lew, Dillion, and Forward [Lew88] developed a software complexity metric that includes both internal and external complexity of a module in order to evaluate its reliability during the development phase. External complexity included the analysis of both static and dynamic data structures used within and among modules. Internal complexity included McCabe's cyclomatic complexity, Halstead's metrics, and other metrics such as the number of program statements. Lew *et al.* evaluated their new metric using two alternate designs for a text processing program. They concluded that fault tolerant techniques to increase software reliability should be used as a supplement to fault avoidance techniques and that complexity measures are useful in guiding the design of reliable software.

Shaw, Howatt, Maness, and Miller [Shaw89] studied the relationship between Halstead's Software Science metrics and compile time statistics of Ada programs. They presented a nonlinear model of compile time for four Ada compilers as well as a compiler performance index. Their results suggested that their model has a high predictive power and provides interesting insights into compiler performance.

The studies just described deal with various issues of software complexity metrics and their relationships with other conceptual entities. However, none address the issue of static software complexity metrics and their relationship to dynamic measurements of programs.

This thesis presents the design and results of an empirical study exploring the relationship between static software complexity metrics and dynamic measurements of Pascal and C programs. This study is designed as pilot study to find out if such a relationship might exist between these two sets of measures. It is not a controlled experiment. It is believed that this study, by the formation of hypotheses to be tested in the context of more extensive studies, will encourage further research in this area.

The sections that follow define the metrics used in this study, describe the experimental design including how the data were collected, discuss the analysis of the measurements, and summarize and conclude with recommendations for future work.

CHAPTER II

DEFINITION OF METRICS

Software metrics have been classified in different ways. One classification offered by Conte *et al.* [Conte86] is to categorize metrics as either process or product metrics depending on whether they measure the development process and environment or the software product, respectively. The classification used in this work divides metrics into static and dynamic classes [Basili83a and Li87]. Static measures are collected by performing static analysis on the source code. Dynamic measures, on the other hand, are collected at run time and may vary depending on the execution environment (system load, input data, etc.). In this thesis, the focus is on static and dynamic classes of product metrics.

2.1 Static Metrics

Static metrics can be further subdivided [Basili83a and Li87] into either data organization, volume, control organization, or hybrid categories. Data organization metrics measure the use of data and interactions among data. They include data binding [Basili75], span [Elshoff76], and slicing [Weiser84]. Volume metrics measure the size of a product and include the number of lines of code, number of statements, and the Software Science metrics [Halstead77a],

as well as others. Control organization metrics measure the comprehensibility of control structures within a program. They include McCabe's cyclomatic complexity [McCabe76], Woodward *et al.*'s Knots metric [Woodward79], and others. Other more recent, hybrid metrics such as [Henry81, Li87, Oviedo80, and Davis88] fall into one or more of these categories.

Lines of code, McCabe's cyclomatic complexity [McCabe76], and the Software Science metrics [Halstead77a] have been chosen for this study. Some of these metrics have been criticized in the literature, because the experiments which claim to validate them make use of small sample sizes and/or small programs [Shen83], or because the metrics are not consistently sensitive to structuring and program style rules [Halstead77a, Weyuker88, Evangelist83, and Evangelist84]. However, they continue to be used because of the volumes of empirical data supporting them.

2.1.1 Lines of Code

The definition for the lines of code metric is not universally agreed upon [Conte86 and Jones78]. Perhaps the simplest is the number of physical lines in a program. However, inserting or deleting blank and/or comment lines from a program would not change the program's performance but would change the measurement and possibly the program's understandability. Another definition is the number of semicolons in a program for programs written in Pascal or other languages that use semicolons to separate statements. The definition for the lines of code metric used in this thesis is the number of nonblank, noncomment lines of code.

2.1.2 McCabe's Cyclomatic Complexity Number

McCabe's cyclomatic complexity number [McCabe76], VG, is based on graph theory [Tucker84] and measures the conceptual complexity of a program. It is the number of linearly independent control paths through a program. As originally defined, this metric is not easily computable unless the control graph is a planar graph [Berge83], in which case the cyclomatic complexity is the same as the number of regions in the control graph. In order to automate its computation, it is easier to compute VG as the number of predicates, π, plus one

$$VG = \pi + 1.$$

This alternative representation of the cyclomatic complexity number is due to McCabe [McCabe76]. For this study, the cyclomatic complexity was computed using the "Pascal Counting Strategy" presented in [Conte86] (see Appendix A) and a "C Counting Strategy" modelled after Conte *et al.*'s (see Appendix B).

Using the formula, $VG = \pi + 1$, presents some potential problems. How should the code segment,

IF c1 AND c2 THEN s,

where **c1** and **c2** are conditionals and **s** is a statement, be counted? The code segment is equivalent to

IF c1 THEN IF c2 THEN s

and

IF c2 THEN IF c1 THEN s.

It is clear that these two equivalent code segments have a cyclomatic complexity of two; therefore, the code segment above with the compound condition would have a cyclomatic complexity of two also. The same reasoning applies to the **OR** logical connective and for other branching/looping constructs such as **WHILE DO**.

2.1.3 Software Science Metrics

Software Science metrics [Halstead77a] basically measure program bulk. They are derived from the four basic metrics:

1) n_1, number of unique operators;
2) N_1, total number of operators;
3) n_2, number of unique operands; and
4) N_2, total number of operands.

Conte *et al.* [Conte86] present a concise discussion of the Software Science metrics.

Vocabulary, n, of a program is the number of unique operators plus the number of unique operands. It represents the total number of unique tokens used in coding the program,

$$n = n_1 + n_2.$$

Program length, N, is the total number of operators plus the total number of operands. It represents the total number of tokens used in coding the program,

$$N = N_1 + N_2.$$

Given the number of unique operators and operands, n_1 and n_2, respectively, the estimated program length, $\hat{N}$, is

$$\hat{N} = n_1 \lg n_1 + n_2 \lg n_2$$

which eliminates the need for computing N_1 and N_2. (All logarithms in this thesis are log base 2 unless explicitly stated otherwise.) Therefore, the length of a program that has not yet been completed can be estimated assuming n_1 and n_2 can be found somehow.

Program volume, V, is the size of a program in terms of either bits or number of mental comparisons needed to write it. A program having a vocabulary of size n and a length of N tokens has a volume given by the following formula

$$V = N \lg n.$$

Potential volume, V^*, is the program volume of the minimal implementation for a particular algorithm. This metric can be difficult to compute because it requires previous knowledge of the algorithm being implemented. Therefore, there is a formula for computing the estimated potential program volume, $\hat{V^*}$, which is

$$\hat{V^*} = V \hat{L}.$$

Estimated program level, $\hat{L}$, is an approximation to the program level, $L = V^* / V$. A program level of one, $L = 1$ which would be the maximum, represents a program of the minimum implementation. Typically, the program level is much less than one. Furthermore, program level is a function of potential program volume which, as discussed above, is difficult to compute;

therefore, program level is estimated by the formula

$$\hat{L} = (2 / n_1) (n_2 / N_2).$$

Program difficulty, D, is the reciprocal of program level. Because program level is a function of the potential program volume which is difficult to compute, program difficulty is estimated by the formula

$$\hat{D} = 1 / \hat{L}.$$

The amount of effort, or number of elementary mental discriminations, E, required to implement a program is

$$E = V / \hat{L} = \hat{D} V$$

$$= (n_1 N_1 N \lg n) / (2 n_2).$$

Because of possible errors in computing $\hat{L}$ and $\hat{D}$, the last expression was used to compute program effort E in this study.

The time, T, to develop a program is given by

$$T = E / ß$$

where ß is the Stroud number which "... is normally set to 18 since this seemed to give the best results in Halstead's earliest experiments," [Conte86].

Language level, λ, as hypothesized by Halstead, should remain fixed for a given programming language. Its formula is

$$\lambda = L V^* = L^2 V.$$

The second expression was used to compute the value of language level with $\hat{L}$ estimating program level L.

The following formulas were used to compute the Software Science metrics for this study:

1) vocabulary, $n = n_1 + n_2$;

2) program length, $N = N_1 + N_2$;

3) estimated program length, $\hat{N} = n_1 \lg n_1 + n_2 \lg n_2$;

4) program volume, $V = N \lg n$;

5) estimate potential program volume, $V^{*\hat{}} = V \hat{L}$;

6) estimated program level, $\hat{L} = (2 / n_1) (n_2 / N_2)$;

7) estimated program difficulty, $\hat{D} = 1 / \hat{L}$;

8) effort, $E = (n_1 N_1 N \lg n) / (2 n_2)$;

9) development time, $T = E / ß$; and

10) language level, $\lambda = \hat{L}^2 V$.

2.2 Dynamic Measurements

Dynamic measurements are collected at run time and vary depending on the environment (system load, input data, etc.). Such external factors are outside the program and may be present during one execution but not during the next. These factors can distort the measurements; and therefore, great care must be taken when collecting them. The dynamic measures are described in the following subsections.

2.2.1 Execution Time

Execution time is the amount of CPU time a program requires to process all input, produce all output, and run to completion, either normally or

abnormally. This is not real time elapsed. The execution time should not include time spent by the process in a suspended state, for instance in a blocked queue waiting for I/O. As will be discussed later, measuring this time can be difficult. Other similar measures include the amount of CPU time spent processing a job as a whole, the amount of CPU time spent compiling a program, and the amount of real time elapsed.

2.2.2 Memory Used During Execution

Memory used during execution is the amount of memory the program requires while executing. This space includes the program segment, the data segment, and the stack segment. The size and content of the stack segment can change over the duration of program execution. Therefore, its size is sometimes reported as an average number of bytes over execution time. Other similar measures include the object code size of the executable module and various counts associated with the symbol table.

2.2.3 Number of Statements Executed

The number of statements executed is the number of source statements executed. This number depends on the input into the program. Some compilers optionally output this number as one of the run-time statistics.

CHAPTER III

EXPERIMENTAL METHODOLOGY

The experimental methodology used in the design of this study follows the framework described in Basili *et al.* [Basili86]. According to the framework, there are four phases of the experimentation process: 1) definition, 2) planning, 3) operation, and 4) interpretation. The definition, planning, and operation phases are described in the following subsections. The interpretation phase and data analysis are presented in Chapter IV.

3.1 Experiment Definition

The motivation of this study is to understand the relationship between static software complexity metrics and dynamic measurements of programs (objects) better. The purpose is to conduct an exploratory empirical study of academic programs (domain) written in Pascal [Findlay87] and C [Kernighan78]. Several programs written by individual students for a particular programming assignment in each of four classes (scope) are evaluated from the perspective of the instructor, realizing there are potential problems of using students in programming classes [Brooks80].

3.2 Experiment Planning

An objective assessment of static and dynamic measurements involving correct program solutions submitted by students in a multivariate design was proposed. The students would be classified either as "novices" or "experts" [Curtis84 and Wiedenbeck85]. The programs were to be evaluated based on the number of nonblank, noncomment lines of code, McCabe's cyclomatic complexity, most of the Software Science metrics, and dynamic measurements such as execution time. Correlation analysis would be used to study possible relationships between the static and dynamic measurements.

Computer Science I (Comsc 2123) and Operating Systems I (Comsc 4323) were to be selected as the two "novice" classes. Both are undergraduate courses at Oklahoma State. Computer Science II (Comsc 2133), an undergraduate course, and Operating Systems II (Comsc 5323), a graduate course, were to be selected as the two "expert" classes. The terms novice and expert have relative meanings here: on a larger scale, the novices might be considered beginners and the experts intermediates [Wiedenbeck85 and Curtis84].

3.3 Experiment Operation

The next two subsections explain the software tools that were developed and used to collect the measurements.

3.3.1 Programs Used to Collect the Data

Because pre-written software packages [Bishop87, de Freitas78, Graham83, and Power83] that collect various static and dynamic measurements were not available, software tools to collect the measurements had to be developed.

On the IBM 3081K mainframe running MVS/XA, two particular tools were developed using the VS Pascal [VSPascal87a,b,c,d] language. The first tool to be developed was a program (see Appendix C) that expects as input a syntactically correct WPascal [Boswell87] program--this was the language used by most students in Comsc 2123 and 2133--and produces as output the number of nonblank, noncomment lines of code, McCabe's cyclomatic complexity, and most of the Software Science metrics. The "Pascal Counting Strategy" presented in [Conte86] was used as a guide in writing the program. A few additions had to be made to the counting strategy to make it applicable to WPascal (see Appendix A). (One apparent omission from Conte *et al.*'s strategy was the keyword **BOOLEAN**. It was added.) The program was compiled and link-edited into a load module [OS78] so that repetitive executions could be made without the need to re-compile and link-edit every time it was used.

Several issues had to be resolved when developing this program. Perhaps the most important was the issue of whether or not to increment McCabe's VG count on the occurrence of **AND** or **OR** in assignment statements. Conte *et al.*'s strategy states that "... **AND** and **OR** include loops/branches controlled by Boolean variables." It was assumed that this means only those **AND**'s and **OR**'s that are part of conditionals. This is the

strategy used in the program developed (see use of **FCOND** in **PROCEDURE MCCABE**, lines 724-775, of Appendix C).

Another issue was how to correctly count the number of **IF-THEN** and **IF-THEN-ELSE** tokens. This was resolved by counting initially the occurrence of **IF** as an **IF-THEN** token, and then upon the occurrence of a subsequent **ELSE**, the **IF-THEN** token count was decremented by one and the **IF-THEN-ELSE** token count was incremented by one (see lines 536-540 in **PROCEDURE TOKCOUNT** of Appendix C).

Another issue was how to handle the counting of pairs of entities such as **BEGIN END, WHILE DO, ARRAY OF,** ( ), etc. The solution was to increment the respective token count on the occurrence of the first token of the entity. All remaining tokens of the entity were skipped using a no-operation instruction (see lines 599-602 in **PROCEDURE TOKCOUNT** of Appendix C). It was assumed that the WPascal programs input to this program were syntactically correct.

Other design issues included the use of constants (see lines 31-146) to represent the keywords of the language to be analyzed (WPascal in this case), the output of warning messages (see lines 270-277) if any of the token arrays become full (this is important if accurate measurements of the number of operators and operands are expected), and the ordering of **IF-THEN-ELSE-IF** statements in **PROCEDURE TOKCOUNT** so that the most frequently occurring tokens are tested for first. This last design decision was made to make the program more efficient with respect to execution time.

Appendix D shows the sample WPascal program used to test the VS Pascal metrics program. Included with the sample program are sample input used by the sample program, output produced by it, detailed metrics, and summary metrics. The detailed metrics include a list of the operators and operands with their respective frequencies. The summary metrics include the values of the lines of code metric, McCabe's cyclomatic complexity, and the Software Science metrics.

The second program (see Appendix E) was also written in VS Pascal and was designed to extract from system accounting data sets records of each compilation made by the students in developing their programs. The program expects as input a system accounting data set and produces as output a data set containing a summary of each log. This information is used later as input to the data analysis described in Chapter IV.

On the VAX* 8350 running ULTRIX** [ULTRIX87], a program (see Appendix F) written in Lex [Lesk75] and C [Kernighan78], [Ritchie78a] was developed using SCCS [Bonanni77 and Dolotta78] to extract the number of lines of code, McCabe's cyclomatic complexity, and the Software Science metrics from C programs--C was the language used by most students in Comsc 4323 and 5323. A set of "C Counting Strategy" rules (see Appendix B) were built from the Pascal counting strategy presented in [Conte86]. The program expects as input a stream of syntactically correct C programs and produces as output the static measurements.

---

* VAX is a Trademark of Digital Equipment Corp.

** ULTRIX is a Trademark of Digital Equipment Corp.

Some of the same issues as those of the VS Pascal metrics program had to be resolved. The most important being whether or not to increment McCabe's VG count on the occurrence of **&&** (logical AND) or || (logical OR) in assignment statements. Using the same strategy as was used in the VS Pascal program, only those **&&**'s and ||'s that are part of conditionals cause VG to be incremented (see use of **fcond** in lines 69, 91, 103, 108, and 129 of Appendix F). An oversight is the omission of setting **fcond** on the occurrence of **for** and **switch**: relational expressions which can include the use of **&&** and || are part of these statements. However, it is assumed that the use of compound relational expressions in **for** and **switch** statements in the programs of this study is minimal.

A problem particular to C is how to effectively recognize function declarations so that McCabe's VG count would be incremented correctly. This problem was resolved by making note of the fact that function declarations occur outside of braces.

A similar problem exists for recognizing function calls (this includes both user-defined and system functions) which are to be counted as operators (see Strategy Rule 5 of Appendix B). This problem was resolved by making note of the facts that function calls occur within at least one nesting level of braces and that they are immediately followed by a left parenthesis.

Another problem particular to C is how to effectively count the number of variables given the fact that a user may define his/her own data types and then use these new data types as he/she would use the standard C data types, for example **int** and **double**. (This is not a problem in Pascal because all

variables can be recognized by the section marker **VAR** or parentheses on function and procedure declarations.) To solve this problem, a flag called **ftype** was set each time **typedef** was recognized and then reset on the occurrence of the next semicolon (see lines 148-152 and 174 of Appendix F). This solution allows all variables defined using standard C and user data types to be counted correctly. The type definition **FILE**, defined in "stdio.h", is included as one of the standard C data types because it is used frequently. If it were not included, then variables declared of this type would not be counted as variables (see next paragraph on limitations).

There are some limitations of this program. First, the program will not correctly count constants and variables that are defined, or whose data types are defined, in external files such as the commonly used **#include** file "stdio.h". For example, the commonly used constants **NULL** and **EOF** are not counted as such. This also applies to external files defined by the programmer. To correct this problem, one could copy the contents of the external file into the file that uses it. Caution must be exercised however. The external file may contain several operators and operands that are not used in the program; and therefore, the operator and operand counts and frequencies could be artificially inflated. One also encounters the risk of including code two, three, or even more times. The metrics program does not check for syntax or duplication of code.

A second limitation is that the metrics program does not view identifiers (tag names) appearing between **struct** and the next left brace as user-defined data types. It was assumed that most usages of structure declarations would be

followed by a type definition declaring the data type of the structure; and therefore, the structure data type would be considered a user-defined data type, and all variables defined of this type would be counted as such. A modification to the program to correct this problem would require context checking for the case in which the keyword **typedef** is followed by a structure definition (which requires the use of braces and herein lies the problem, see page 140 of Kernighan and Ritchie [Kernighan78] for an example). The intervening left brace between **typedef** and the next semicolon would cause the flag **ftype** to be reset, and thus identifiers between the final right brace of the definition and the next semicolon would not be counted as user-defined data types.

When using the Lex and C program to compute metrics on a stream of C programs, one should always pipe in first any files, such as "defs.h", "global.h", or "global.c", that contain definitions of constants, variables, or data types. This is required if the variable count is to be accurate.

Appendix G shows the sample C program used to test the Lex & C metrics program. Included with the sample program are sample input used by the sample program, output produced by it, detailed metrics, and summary metrics. The detailed metrics include a list of the operators and operands with their respective frequencies. The summary metrics include the values of the lines of code metric, McCabe's cyclomatic complexity, and the Software Science metrics.

Various UNIX*** utilities, shell commands [Bourne78 and Sobell85], and the ULTRIX **sa** command [ULTRIX87] were used to extract from system

---

*** UNIX is a Trademark of AT&T Bell Laboratories.

accounting files records of each compilation made by students in developing their programs. For example, to extract the number of compilations made by the students in Comsc 4323 on March 26, 1989, the following command was used

```
/etc/sa -u /usr/local/lib/acctg/890326.sa | \
grep '^u2337.*cc$' > 890326.cc
```

The ULTRIX **sa** command with the **-u** flag extracts accounting information from the system accounting file **890326.sa** by user id. The UNIX **grep** command [UNIX86 and Sobell85] selects only those compilations (records ending with **cc**) made by students in Comsc 4323 (they were assigned project numbers beginning with **u2337**) and saves the records in the appropriate file, **890326.cc**. A **.cc** file was created for each date the students worked on their programs. This same command was used to extract the same information for Comsc 5323 (students were assigned project numbers beginning with **u0798**).

The following UNIX **awk** command [UNIX86 and Sobell85] was used to report from the **.cc** files the date and user id of each compilation made

```
awk '{print "  ", 890326, $1}' 890326.cc
```

This particular command reports from the file **890326.cc** all compilations made on March 26, 1989. Actually, a shell script containing a group of these commands that were modified for the specific dates in question was used to generate a file of all compilations made by all students. This file was transmitted to the IBM mainframe for processing by SAS.

3.3.2 Data Collection

Both static and dynamic measurements were collected on their native systems: programs for Comsc 2123 and 2133 were on the IBM and those for Comsc 4323 and 5323 were on the VAX. Once collected, all data were migrated to the IBM using Kermit [da Cruz84].

A TSO Clist [TSO86] was developed to collect the WPascal programs on the IBM for the two Pascal classes. Various UNIX commands were used to collect the C programs on the VAX for the two C classes.

In the two Pascal classes, there were approximately 100 students in Comsc 2123 and approximately 60 students in 2133. The 2123 students were asked to write an implementation of a surrogate sort for sorting numbers, while the 2133 students were asked to write a hashing (linear probing) program for storing and retrieving records.

In the two C classes, there were approximately 60 students in Comsc 4323 and approximately 45 students in 5323. The 4323 students were asked to simulate a multiprogramming operating system with variable-partitioned memory utilizing multilevel feedback queues for CPU scheduling. The 5323 students were asked to implement a multiprogramming operating system with variable-sized memory segments utilizing a round robin scheduler augmented with aging for the CPU. However, unlike the programs written by the 4323 students, the 5323 students' operating systems were required to run programs written in hexadecimal from a limited instruction set on an architectural machine simulation of their own design. For detailed descriptions of the programming

simulation of their own design. For detailed descriptions of the programming assignments, see Appendix H.

This study was performed in cooperation with each instructor as part of a general evaluation process. Only correct, working program solutions submitted by students that received full marks were used in this study. This and the fact that some students wrote their programs on other systems, for instance personal computers, and/or used other programming languages explains why not all programs were used in the study.

Dynamic measurements were collected two to three times for each program (see Appendix I) so that the effects of other processes or jobs running on the systems would be minimized. On the VAX, the UNIX **ps** command [UNIX86 and ULTRIX87] was used to check the system load, and on the IBM, the "R" option of the **IOF** facility was used for the same purpose. Programs were measured only during low system utilization times. The measurements were averaged weighing each equally.

The input data used when collecting the dynamic measurements were the same as the data used by the students when turning their programs in to be graded. The data were created by each instructor, and it is believed that they are representative of all possible input that the programs were designed to accept.

For the programs written in Comsc 5323, there were actually two input files. One contained only valid data, while the other contained some intentionally injected errors. Both sets were used when collecting the dynamic

measurements, and the execution times, after being averaged, were added together.

The dynamic measurements collected on the IBM were taken from the job statistics when the programs were executed and include the total system time [MVS87], total job time, compilation time, execution time, object code size, and memory requested during execution (see the dynamic measurements for Comsc 2123 and 2133 in Appendix I). The total system time is the amount of CPU time required to process the job on the system. The total job time is the amount of CPU time required to compile and execute a WPascal program. This time is reported by the WPascal compiler and is less than the total system time which is reported by the system. All times were reported in seconds.

The UNIX **time** command [UNIX86 and ULTRIX87] was used to collect the dynamic measurements on the VAX which include the amount of CPU time spent executing the user's code, amount of CPU time spent in the system, elapsed time, percentage of CPU utilization, average amount of memory used for the program and data segments [Ritchie78b] (reported in kilobytes and converted to bytes before being analyzed) over the CPU time involved--the sum of these two numbers represents the execution memory, number of I/O's, and number of page faults and swaps (see dynamic measurements for Comsc 4323 and 5323 in Appendix I). These last two numbers were used to guarantee that the processes did not generate any page faults and that they were not swapped out of main memory [Ritchie78b] both of which could adversely affect the execution times [Thompson78 and Feder84]. All times were reported in seconds.

Other techniques for collecting dynamic measurements are discussed in the literature [Graham83, Bishop87, and Wienberger84]. These include using the UNIX utilities **prof** and **gprof** [UNIX86] which were looked at for this study but deemed to provide more detail than what was necessary for the objectives of this work.

CHAPTER IV

ANALYSIS OF MEASUREMENTS

All data analysis was done on the IBM mainframe using SAS [SAS85a,b]. Standard statistical methods described in [Triola83, Conte86, Snedecor80, Dowdy83, and Johnson82] were used. As a note, (n-1) degrees of freedom was used in all analyses as an unbiased estimator [Conte86].

The sample sizes for this study were not arrived at statistically [Triola83]; rather, all correct solutions submitted were used. Table I shows the number of programs from each class used in the study.

Dynamic measurements were taken two to three times for each program. Means of these measurements were computed and used in the analysis (see Appendix J).

There were actually two sets of dynamic measurements for each program from Comsc 5323 (see discussion in Section 3.3.2). These two sets--the one corresponding to input data with injected errors is labeled A, and the other corresponding to input data with no errors is labeled B in Appendix J--were combined before doing the correlation analysis, i.e. execution times were added together.

T tests were used to show that the dynamic measurements for each program are representative of their populations. The results are given in

Appendix J (columns labeled "T" and "PR>|T|"). All values of PR>|T|, excluding those of extraneous measurements such as NREAD, are less than 0.05 with most being less than 0.01, meaning that the sample means or individual dynamic measurements are not significantly different. In other words, the dynamic measurements are consistent and therefore accurately represent the dynamic measurements of the populations.

TABLE I

Number of Programs from Each Class Used in the Study

Class	Number
Comsc 2123	12
Comsc 2133	12
Comsc 4323	7
Comsc 5323	10

Table II contains a list of statistics of the measurements (see Appendix K for a list of the static and dynamic measurements). The column labeled "C.V." is the percent coefficient of variation, 100 times standard deviation divided by the mean. As a point of interest, the average estimated program difficulty, EPDIF, increases from 97.5 to 160.2 to 375.3 to 617.3 for Comsc 2123, 2133, 4323, 5323, respectively. This agrees with what is known about these classes: they get harder!

TABLE II

Statistics of the Measurements

COMSC 2123

VARIABLE	LABEL	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	C.V
LOC	# OF NBNC LINES OF CODE	12	101.50000	57.00000	145.00000	23.34524	23.000
VG	MCCABE'S CYCLOMATIC COMPLEXITY	12	18.00000	8.00000	30.00000	5.41043	30.058
ETA1	# OF UNIQUE OPERATORS	12	39.58333	32.00000	42.00000	2.74552	6.936
N1	TOTAL # OF OPERATORS	12	303.58333	178.00000	448.00000	74.65248	24.590
ETA2	# OF UNIQUE OPERANDS	12	41.08333	34.00000	62.00000	7.78645	18.953
N2	TOTAL # OF OPERANDS	12	200.83333	117.00000	283.00000	43.06408	21.443
PLEN	PROGRAM LENGTH	12	504.41667	295.00000	731.00000	114.58499	22.716
EPLEN	EST. PROGRAM LENGTH	12	431.31917	339.52000	595.64000	63.90692	14.817
PVOL	PROGRAM VOLUME	12	3204.19333	1789.50000	4898.02000	797.32714	24.884
EPPVOL	EST. POTENTIAL PROGRAM VOLUME	12	33.06500	27.97000	51.10000	6.82931	20.654
EPLEV	EST. PROGRAM LEVEL	12	0.01069	0.00760	0.01870	0.00274	25.641
EPDIF	EST. PROGRAM DIFFICULTY	12	97.52417	53.48600	131.17600	18.24868	18.712
EFFORT	# OF ELEM. MENTAL DISCRIM.	12	320008.41667	95712.00000	483833.00000	109071.01032	34.084
TIME	DEVELOPMENT TIME (HOURS)	12	4.93840	1.47704	7.46656	1.68319	34.084
LAMBDA	LANGUAGE LEVEL	12	0.35417	0.21000	0.63000	0.11996	33.871
MTSYSTIM	MEAN TOTAL SYSTEM TIME (SECS)	12	0.43000	0.35000	0.69000	0.09088	21.135
MNSTMTS	MEAN # OF SOURCE STMTS. EXEC.	12	2716.50000	2081.00000	5885.00000	1029.19117	37.887
MTJOBTIM	MEAN TOTAL JOB TIME (SECS)	12	0.33076	0.24840	0.58920	0.09033	27.310
MCOMPTIM	MEAN COMPILATION TIME (SECS)	12	0.15801	0.08150	0.33435	0.06431	40.700
MEXECTIM	MEAN EXECUTION TIME (SECS)	12	0.17276	0.12265	0.25490	0.03873	22.420
MOBJSIZE	MEAN OBJECT CODE SIZE (BYTES)	12	5658.58333	3045.00000	10200.00000	1892.94343	33.453
MEXECMEM	MEAN EXECUTION MEMORY (BYTES)	12	10134.00000	10080.00000	10176.00000	41.09413	0.406
NCOMP	# OF COMPILATIONS	12	58.91667	1.00000	283.00000	73.71500	125.117

COMSC 2133

VARIABLE	LABEL	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	C.V.
LOC	# OF NBNC LINES OF CODE	12	287.75000	142.00000	412.00000	74.34211	25.836
VG	MCCABE'S CYCLOMATIC COMPLEXITY	12	45.75000	22.00000	65.00000	10.44575	22.832
ETA1	# OF UNIQUE OPERATORS	12	54.41667	44.00000	64.00000	4.85159	8.916
N1	TOTAL # OF OPERATORS	12	835.66667	387.00000	1070.00000	187.36272	22.421
ETA2	# OF UNIQUE OPERANDS	12	88.25000	40.00000	128.00000	25.75089	29.180
N2	TOTAL # OF OPERANDS	12	509.58333	181.00000	643.00000	127.30097	24.981
PLEN	PROGRAM LENGTH	12	1345.25000	568.00000	1699.00000	312.46589	23.227
EPLEN	EST. PROGRAM LENGTH	12	889.68333	453.09000	1228.47000	221.71632	24.921
PVOL	PROGRAM VOLUME	12	9672.47083	3630.84000	12701.27000	2526.79460	26.124
EPPVOL	EST. POTENTIAL PROGRAM VOLUME	12	61.57000	36.47000	94.15000	18.43522	29.942
EPLEV	EST. PROGRAM LEVEL	12	0.00652	0.00440	0.01000	0.00153	23.463
EPDIF	EST. PROGRAM DIFFICULTY	12	160.23617	99.55000	225.20300	34.05990	21.256
EFFORT	# OF ELEM. MENTAL DISCRIM.	12	1573170.50000	361450.00000	2292468.00000	512866.02629	32.601
TIME	DEVELOPMENT TIME (HOURS)	12	24.27732	5.57793	35.37759	7.91460	32.601
LAMBDA	LANGUAGE LEVEL	12	0.40750	0.17000	0.74000	0.16939	41.568
MTSYSTIM	MEAN TOTAL SYSTEM TIME (SECS)	12	1.17875	0.58000	1.54000	0.27964	23.723
MNSTMTS	MEAN # OF SOURCE STMTS. EXEC.	12	7380.50000	4312.00000	14172.00000	3569.23311	48.360
MTJOBTIM	MEAN TOTAL JOB TIME (SECS)	12	1.07700	0.47720	1.44035	0.27944	25.946
MCOMPTIM	MEAN COMPILATION TIME (SECS)	12	0.49522	0.21990	0.76610	0.13924	28.116
MEXECTIM	MEAN EXECUTION TIME (SECS)	12	0.58178	0.25735	0.90005	0.18246	31.362
MOBJSIZE	MEAN OBJECT CODE SIZE (BYTES)	12	15896.83333	6145.00000	22005.00000	4147.25885	26.089
MEXECMEM	MEAN EXECUTION MEMORY (BYTES)	12	11696.00000	10176.00000	25176.00000	4346.08288	37.159
NCOMP	# OF COMPILATIONS	12	56.41667	1.00000	264.00000	74.48302	132.023

COMSC 4323

VARIABLE	LABEL	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	C.V.
LOC	# OF NBNC LINES OF CODE	7	838.5714	694.0000	1146.0000	145.28691	17.326
VG	MCCABE'S CYCLOMATIC COMPLEXITY	7	132.2857	103.0000	195.0000	29.79214	22.521
ETA1	# OF UNIQUE OPERATORS	7	68.5714	63.0000	79.0000	5.56349	8.113
N1	TOTAL # OF OPERATORS	7	2656.1429	2222.0000	3201.0000	356.12752	13.408
ETA2	# OF UNIQUE OPERANDS	7	177.5714	156.0000	207.0000	17.47243	9.840
N2	TOTAL # OF OPERANDS	7	1945.5714	1570.0000	2448.0000	323.24854	16.615
PLEN	PROGRAM LENGTH	7	4601.7143	3792.0000	5649.0000	673.65612	14.639
EPLEN	EST. PROGRAM LENGTH	7	1746.4400	1554.2000	1969.1200	139.06640	7.963
PVOL	PROGRAM VOLUME	7	36557.8500	29867.7500	44867.2600	5503.09166	15.053
EPPVOL	EST. POTENTIAL PROGRAM VOLUME	7	98.4986	75.1300	124.5300	15.99410	16.238
EPLEV	EST. PROGRAM LEVEL	7	0.0027	0.0021	0.0032	0.00037	13.691
EPDIF	EST. PROGRAM DIFFICULTY	7	375.3191	309.2380	477.1530	56.56697	15.072
EFFORT	# OF ELEM. MENTAL DISCRIM.	7	13838202.8571	10550807.0000	21408529.0000	3736788.30840	27.003
TIME	DEVELOPMENT TIME (HOURS)	7	213.5525	162.8211	330.3785	57.66649	27.003
LAMBDA	LANGUAGE LEVEL	7	0.2714	0.1900	0.3600	0.07128	26.261
MUTIME	MEAN USER TIME (SECS)	7	1.9619	0.3000	3.2000	0.94583	48.210
MSYSTIME	MEAN SYSTEM TIME (SECS)	7	0.3714	0.1333	1.1667	0.35872	96.577
MELAPSED	MEAN ELAPSED TIME (H:MM:SS)	7	1.9048	0.0000	3.0000	0.99469	52.221
MEXECTIM	MEAN EXECUTION TIME (SECS)	7	2.3333	0.4333	3.5667	0.99350	42.578
MOBJSIZE	MEAN OBJECT CODE SIZE (BYTES)	7	22674.2857	21504.0000	24576.0000	1244.14545	5.487
MEXECMEM	MEAN EXECUTION MEMORY (BYTES)	7	110396.9524	81578.6667	141312.0000	22578.67509	20.452
NCOMP	# OF COMPILATIONS	7	84.4286	12.0000	204.0000	79.34494	93.979

TABLE II

Statistics of the Measurements (cont.)

COMSC 5323

VARIABLE	LABEL	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	C.V
LOC	# OF NBNC LINES OF CODE	10	2127.6000	1381.0000	2576.000	491.7434	23.113
VG	MCCABE'S CYCLOMATIC COMPLEXITY	10	426.1000	302.0000	587.000	94.9251	22.278
ETA1	# OF UNIQUE OPERATORS	10	120.3000	71.0000	158.000	30.1811	25.088
N1	TOTAL # OF OPERATORS	10	7097.9000	5327.0000	8740.000	1172.5966	16.520
ETA2	# OF UNIQUE OPERANDS	10	452.9000	290.0000	671.000	139.2803	30.753
N2	TOTAL # OF OPERANDS	10	4554.5000	3913.0000	5416.000	513.2503	11.269
PLEN	PROGRAM LENGTH	10	11652.4000	9240.0000	14146.000	1616.3156	13.871
EPLEN	EST. PROGRAM LENGTH	10	4859.9960	2870.1700	7307.480	1652.8204	34.009
PVOL	PROGRAM VOLUME	10	106506.6900	78937.6100	133903.560	18159.9141	17.050
EPPVOL	EST. POTENTIAL PROGRAM VOLUME	10	176.2370	137.1700	249.210	37.3662	21.202
EPLEV	EST. PROGRAM LEVEL	10	0.0017	0.0012	0.002	0.0003	18.357
EPDIF	EST. PROGRAM DIFFICULTY	10	617.3290	459.9720	832.742	116.8723	18.932
EFFORT	# OF ELEM. MENTAL DISCRIM.	10	66411808.0000	36309081.0000	107451348.000	19728363.0760	29.706
TIME	DEVELOPMENT TIME (HOURS)	10	1024.8736	560.3253	1658.200	304.4500	29.706
LAMBDA	LANGUAGE LEVEL	10	0.3010	0.1900	0.460	0.1017	33.788
MUTIME	MEAN USER TIME (SECS)	10	70.5800	41.9333	123.967	28.9676	41.042
MSYSTIME	MEAN SYSTEM TIME (SECS)	10	32.2867	1.1000	234.233	73.3360	227.140
MELAPSED	MEAN ELAPSED TIME (H:MM:SS)	10	119.7333	55.6667	351.667	87.9346	73.442
MEXECTIM	MEAN EXECUTION TIME (SECS)	10	102.8667	45.5667	336.667	89.4596	86.967
MOBJSIZE	MEAN OBJECT CODE SIZE (BYTES)	10	45977.6000	35840.0000	57344.000	8614.1876	18.736
MEXECMEM	MEAN EXECUTION MEMORY (BYTES)	10	225894.4000	162816.0000	299690.667	48156.3151	21.318
NCOMP	# OF COMPILATIONS	10	586.1000	136.0000	1525.000	458.3852	78.209

Another interesting point is that the language levels for the two Pascal classes, 2123 and 2133, are 0.35 and 0.40, respectively, while for the two C classes, 4323 and 5323, they are 0.27 and 0.30, respectively. Typically language levels range from 0.88 for the CDC assembly language to 1.53 for PL/I [Halstead77a]. The language levels of this study seem unusually low, but they are relatively correct: C is not considered a "very high level" language [Kernighan78].

Figures 1 - 4 show the distribution of the number of nonblank, noncomment lines of code for each class.

Pearson product-moment correlations [SAS85a,b and Snedecor80] were computed between all measurements within each class (see Appendix L). Use of this correlation method requires that the measurements be parametric [Conte86]. The measurements are independent, they are drawn from normally distributed populations, the populations have nearly the same standard

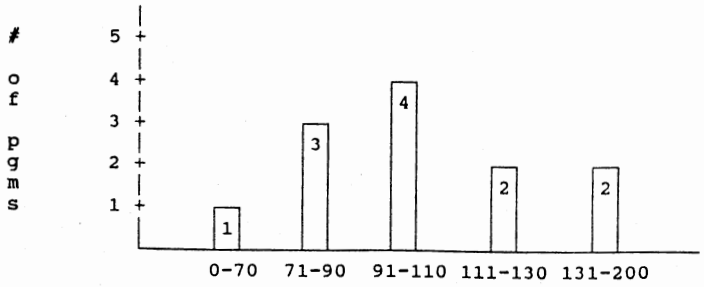

Figure 1. Number of nonblank, noncomment lines of code broken down by ranges for Comsc 2123.

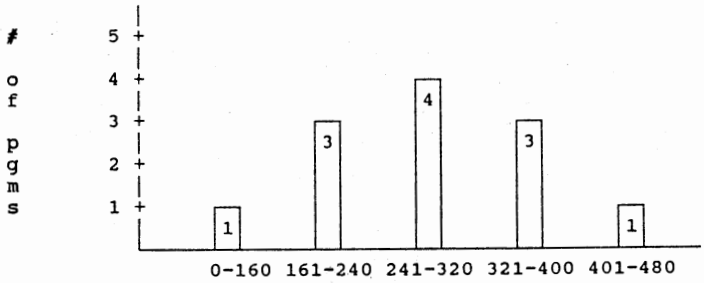

Figure 2. Number of nonblank, noncomment lines of code broken down by ranges for Comsc 2133.

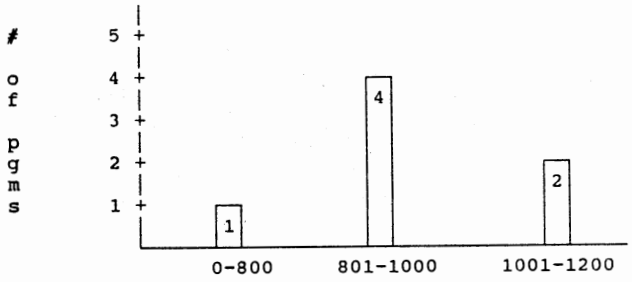

Figure 3. Number of nonblank, noncomment lines of code broken down by ranges for Comsc 4323.

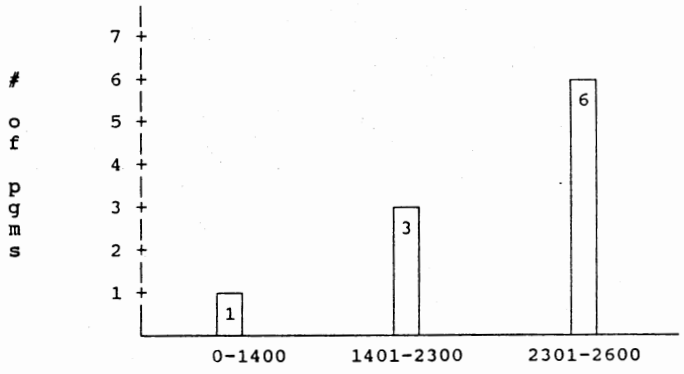

Figure 4. Number of nonblank, noncomment lines of code broken down by ranges for Comsc 5323.

deviations, and the measurements are all in at least an interval scale, meaning that the data have meaningful differences and can be ranked and categorized.

Table III shows the correlations between selected static complexity metrics. Significance levels are printed below their respective correlation coefficients.

The number of compilations does not correlate well with any of the other measurements. The only class for which it does correlate well with anything is Comsc 2133. Within this class, it correlates well with about everything. Therefore, no significance correlations exist between the number of compilations and other measurements.

With the exception of Comsc 4323, estimated and actual program lengths correlate well. This is expected [Halstead77a and Halstead79].

For Comsc 2123 and 2133, strong, positive correlations (significance level of 0.05 with most being less than 0.01) exist between LOC, VG, and most of the Software Science metrics. The only exceptions are the correlations between ETA1 and ETA2 and EFFORT and ETA2 for Comsc 2123, and between LOC and ETA1 and ETA2 and EFFORT for Comsc 2133.

For Comsc 4323, strong, positive correlations (significance level of 0.05 or less) exist between LOC and VG, LOC and N1, LOC and EFFORT, VG and EFFORT, N1 and N2, N1 and EFFORT, and N2 and EFFORT. Unusual negative correlations exist between other measurements. These might be explained by the small sample size.

TABLE III

Correlations Between Selected Static Complexity Metrics

COMSC 2123

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	LOC	VG	ETA1	N1	ETA2	N2	EFFORT
LOC	1.00000	0.94646	0.63755	0.98085	0.73242	0.90616	0.90190
# OF NBNC LINES OF CODE	0.0000	0.0001	0.0257	0.0001	0.0068	0.0001	0.0001
VG	0.94646	1.00000	0.61200	0.95635	0.75743	0.91496	0.87946
MCCABE'S CYCLOMATIC COMPLEXITY	0.0001	0.0000	0.0344	0.0001	0.0043	0.0001	0.0002
ETA1	0.63755	0.61200	1.00000	0.60718	0.39725	0.58603	0.63742
# OF UNIQUE OPERATORS	0.0257	0.0344	0.0000	0.0363	0.2010	0.0452	0.0258
N1	0.98085	0.95635	0.60718	1.00000	0.69509	0.88686	0.92435
TOTAL # OF OPERATORS	0.0001	0.0001	0.0363	0.0000	0.0121	0.0001	0.0001
ETA2	0.73242	0.75743	0.39725	0.69509	1.00000	0.75483	0.49589
# OF UNIQUE OPERANDS	0.0068	0.0043	0.2010	0.0121	0.0000	0.0045	0.1011
N2	0.90616	0.91496	0.58603	0.88686	0.75483	1.00000	0.90826
TOTAL # OF OPERANDS	0.0001	0.0001	0.0452	0.0001	0.0045	0.0000	0.0001
EFFORT	0.90190	0.87946	0.63742	0.92435	0.49589	0.90826	1.00000
# OF ELEM. MENTAL DISCRIM.	0.0001	0.0002	0.0258	0.0001	0.1011	0.0001	0.0000

COMSC 2133

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	LOC	VG	ETA1	N1	ETA2	N2	EFFORT
LOC	1.00000	0.81528	0.48904	0.91297	0.93516	0.83154	0.57460
# OF NBNC LINES OF CODE	0.0000	0.0012	0.1066	0.0001	0.0001	0.0008	0.0507
VG	0.81528	1.00000	0.79691	0.88362	0.74479	0.86692	0.82754
MCCABE'S CYCLOMATIC COMPLEXITY	0.0012	0.0000	0.0019	0.0001	0.0055	0.0003	0.0009
ETA1	0.48904	0.79691	1.00000	0.71533	0.53538	0.72715	0.84325
# OF UNIQUE OPERATORS	0.1066	0.0019	0.0000	0.0089	0.0728	0.0074	0.0006
N1	0.91297	0.88362	0.71533	1.00000	0.90987	0.97111	0.81631
TOTAL # OF OPERATORS	0.0001	0.0001	0.0089	0.0000	0.0001	0.0001	0.0012
ETA2	0.93516	0.74479	0.53538	0.90987	1.00000	0.82448	0.53309
# OF UNIQUE OPERANDS	0.0001	0.0055	0.0728	0.0001	0.0000	0.0010	0.0743
N2	0.83154	0.86692	0.72715	0.97111	0.82448	1.00000	0.89612
TOTAL # OF OPERANDS	0.0008	0.0003	0.0074	0.0001	0.0010	0.0000	0.0001
EFFORT	0.57460	0.82754	0.84325	0.81631	0.53309	0.89612	1.00000
# OF ELEM. MENTAL DISCRIM.	0.0507	0.0009	0.0006	0.0012	0.0743	0.0001	0.0000

COMSC 4323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 7

	LOC	VG	ETA1	N1	ETA2	N2	EFFORT
LOC	1.00000	0.97537	-0.05779	0.76684	0.12000	0.72245	0.87030
# OF NBNC LINES OF CODE	0.0000	0.0002	0.9021	0.0443	0.7977	0.0667	0.0108
VG	0.97537	1.00000	-0.05645	0.70401	0.04126	0.65348	0.82705
MCCABE'S CYCLOMATIC COMPLEXITY	0.0002	0.0000	0.9043	0.0775	0.9300	0.1114	0.0217
ETA1	-0.05779	-0.05645	1.00000	-0.60596	-0.51657	-0.60733	-0.24896
# OF UNIQUE OPERATORS	0.9021	0.9043	0.0000	0.1492	0.2352	0.1481	0.5903
N1	0.76684	0.70401	-0.60596	1.00000	0.54238	0.96639	0.83008
TOTAL # OF OPERATORS	0.0443	0.0775	0.1492	0.0000	0.2085	0.0004	0.0208
ETA2	0.12000	0.04126	-0.51657	0.54238	1.00000	0.37925	0.02497
# OF UNIQUE OPERANDS	0.7977	0.9300	0.2352	0.2085	0.0000	0.4015	0.9576
N2	0.72245	0.65348	-0.60733	0.96639	0.37925	1.00000	0.89586
TOTAL # OF OPERANDS	0.0667	0.1114	0.1481	0.0004	0.4015	0.0000	0.0064
EFFORT	0.87030	0.82705	-0.24896	0.83008	0.02497	0.89586	1.00000
# OF ELEM. MENTAL DISCRIM.	0.0108	0.0217	0.5903	0.0208	0.9576	0.0064	0.0000

TABLE III

Correlations Between Selected Static Complexity Metrics (cont.)

COMSC 5323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 10

	LOC	VG	ETA1	N1	ETA2	N2	EFFORT
LOC # OF NBNC LINES OF CODE	1.00000 0.0000	0.69927 0.0244	0.85601 0.0016	0.84728 0.0020	0.86882 0.0011	0.47193 0.1685	0.51880 0.1244
VG MCCABE'S CYCLOMATIC COMPLEXITY	0.69927 0.0244	1.00000 0.0000	0.62963 0.0511	0.77014 0.0092	0.66707 0.0351	0.50208 0.1392	0.46156 0.1793
ETA1 # OF UNIQUE OPERATORS	0.85601 0.0016	0.62963 0.0511	1.00000 0.0000	0.73479 0.0155	0.85517 0.0016	0.26764 0.4547	0.49412 0.1466
N1 TOTAL # OF OPERATORS	0.84728 0.0020	0.77014 0.0092	0.73479 0.0155	1.00000 0.0000	0.70359 0.0232	0.80925 0.0046	0.80346 0.0051
ETA2 # OF UNIQUE OPERANDS	0.86882 0.0011	0.66707 0.0351	0.85517 0.0016	0.70359 0.0232	1.00000 0.0000	0.23987 0.5044	0.20964 0.5610
N2 TOTAL # OF OPERANDS	0.47193 0.1685	0.50208 0.1392	0.26764 0.4547	0.80925 0.0046	0.23987 0.5044	1.00000 0.0000	0.84682 0.0020
EFFORT # OF ELEM. MENTAL DISCRIM.	0.51880 0.1244	0.46156 0.1793	0.49412 0.1466	0.80346 0.0051	0.20964 0.5610	0.84682 0.0020	1.00000 0.0000

For Comsc 5323, strong, positive correlations (0.05 or less) exist between LOC, VG, and most of the Software Science metrics. In general, N2 and EFFORT do not correlate well with the other metrics. These are reasonably large programs (mean LOC over 2100, see Table II), and therefore N2 and EFFORT may be misleading.

Basically, strong, positive correlations exist between lines of code, McCabe's cyclomatic complexity, and the Software Science metrics for each class. These correlations are expected [Crawford85 and Woodward79].

Table IV shows the correlations between the dynamic measurements. For all but Comsc 4323, these measurements are strongly correlated with exceptions being between compilation and execution times for Comsc 2123 and 2133 and between system and user times for Comsc 5323. This should be

TABLE IV

Correlations Between Dynamic Measurements

COMSC 2123

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	MTSYSTIM	MTJOBTIM	MCOMPTIM	MEXECTIM
MTSYSTIM	1.00000	0.99962	0.92976	0.78780
MEAN TOTAL SYSTEM TIME (SECS)	0.0000	0.0001	0.0001	0.0023
MTJOBTIM	0.99962	1.00000	0.92917	0.78966
MEAN TOTAL JOB TIME (SECS)	0.0001	0.0000	0.0001	0.0023
MCOMPTIM	0.92976	0.92917	1.00000	0.50694
MEAN COMPILATION TIME (SECS)	0.0001	0.0001	0.0000	0.0925
MEXECTIM	0.78780	0.78966	0.50694	1.00000
MEAN EXECUTION TIME (SECS)	0.0023	0.0023	0.0925	0.0000

COMSC 2133

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	MTSYSTIM	MTJOBTIM	MCOMPTIM	MEXECTIM
MTSYSTIM	1.00000	0.99997	0.82390	0.90264
MEAN TOTAL SYSTEM TIME (SECS)	0.0000	0.0001	0.0010	0.0001
MTJOBTIM	0.99997	1.00000	0.82473	0.90205
MEAN TOTAL JOB TIME (SECS)	0.0001	0.0000	0.0010	0.0001
MCOMPTIM	0.82390	0.82473	1.00000	0.49984
MEAN COMPILATION TIME (SECS)	0.0010	0.0010	0.0000	0.0980
MEXECTIM	0.90264	0.90205	0.49984	1.00000
MEAN EXECUTION TIME (SECS)	0.0001	0.0001	0.0980	0.0000

COMSC 4323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 7

	MUTIME	MSYSTIME	MELAPSED	MEXECTIM
MUTIME	1.00000	-0.05341	0.94621	0.93274
MEAN USER TIME (SECS)	0.0000	0.9095	0.0013	0.0022
MSYSTIME	-0.05341	1.00000	0.22984	0.31022
MEAN SYSTEM TIME (SECS)	0.9095	0.0000	0.6200	0.4983
MELAPSED	0.94621	0.22984	1.00000	0.98380
MEAN ELAPSED TIME (H:MM:SS)	0.0013	0.6200	0.0000	0.0001
MEXECTIM	0.93274	0.31022	0.98380	1.00000
MEAN EXECUTION TIME (SECS)	0.0022	0.4983	0.0001	0.0000

TABLE IV

Correlations Between Dynamic Measurements (cont.)

COMSC 5323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=O / N = 10

	MUTIME	MSYSTIME	MELAPSED	MEXECTIM
MUTIME	1.00000	0.42030	0.61214	0.66835
MEAN USER TIME (SECS)	0.0000	0.2265	0.0600	0.0346
MSYSTIME	0.42030	1.00000	0.97164	0.95586
MEAN SYSTEM TIME (SECS)	0.2265	0.0000	0.0001	0.0001
MELAPSED	0.61214	0.97164	1.00000	0.99473
MEAN ELAPSED TIME (H:MM:SS)	0.0600	0.0001	0.0000	0.0001
MEXECTIM	0.66835	0.95586	0.99473	1.00000
MEAN EXECUTION TIME (SECS)	0.0346	0.0001	0.0001	0.0000

expected; time to compile a program, unless it is highly sequential, will not correlate well with its execution time, and system time can be relatively independent of user time.

Table V shows the correlations between selected static and dynamic measurements (see Appendix L for a complete list of correlations). Significance levels are printed below their respective correlation coefficients.

For Comsc 2123 and 2133, strong, positive correlations (0.05 or less) exist between the static metrics and the mean compilation time, mean total job time (compilation time plus execution time), mean total system time which includes the compilation time, and the mean object code size. These correlations are expected because as the volume of a program increases so should the amount of time to compile it and the size of its object code.

TABLE V

Correlations Between Selected Static and Dynamic Measurements

COMSC 2123

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	LOC	VG	ETA1	N1	ETA2	N2	EFFORT
MEXECTIM	0.22931	0.33592	0.17919	0.25715	0.53439	0.28162	0.14742
MEAN EXECUTION TIME (SECS)	0.4734	0.2857	0.5774	0.4198	0.0735	0.3752	0.6475
MEXECMEM	0.01933	0.15701	0.06285	0.03360	0.29832	0.12514	-0.01380
MEAN EXECUTION MEMORY (BYTES)	0.9525	0.6260	0.8461	0.9174	0.3463	0.6984	0.9660
MOBJSIZE	0.94968	0.95633	0.47397	0.94373	0.81092	0.89397	0.83353
MEAN OBJECT CODE SIZE (BYTES)	0.0001	0.0001	0.1196	0.0001	0.0014	0.0001	0.0008
MTSYSTIM	0.73336	0.79040	0.42811	0.73230	0.85754	0.71243	0.57776
MEAN TOTAL SYSTEM TIME (SECS)	0.0066	0.0022	0.1650	0.0068	0.0004	0.0093	0.0491
MTJOBTIM	0.72733	0.78695	0.41819	0.72676	0.85665	0.71115	0.57369
MEAN TOTAL JOB TIME (SECS)	0.0073	0.0024	0.1761	0.0074	0.0004	0.0095	0.0511
MCOMPTIM	0.88359	0.90309	0.47947	0.86597	0.88162	0.82940	0.71704
MEAN COMPILATION TIME (SECS)	0.0001	0.0001	0.1147	0.0003	0.0001	0.0008	0.0087

COMSC 2133

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	LOC	VG	ETA1	N1	ETA2	N2	EFFORT
MEXECTIM	0.67814	0.70947	0.31091	0.68623	0.50503	0.70343	0.61506
MEAN EXECUTION TIME (SECS)	0.0154	0.0098	0.3253	0.0137	0.0940	0.0107	0.0333
MEXECMEM	0.10453	0.42413	0.47426	0.20668	0.18676	0.15390	0.27993
MEAN EXECUTION MEMORY (BYTES)	0.7465	0.1694	0.1193	0.5192	0.5611	0.6330	0.3782
MOBJSIZE	0.91381	0.89617	0.74231	0.98546	0.90348	0.93814	0.79719
MEAN OBJECT CODE SIZE (BYTES)	0.0001	0.0001	0.0057	0.0001	0.0001	0.0001	0.0019
MTSYSTIM	0.84472	0.88532	0.59445	0.91068	0.76094	0.89505	0.78725
MEAN TOTAL SYSTEM TIME (SECS)	0.0005	0.0001	0.0415	0.0001	0.0040	0.0001	0.0024
MTJOBTIM	0.84304	0.88630	0.59781	0.91025	0.75992	0.89457	0.78811
MEAN TOTAL JOB TIME (SECS)	0.0006	0.0001	0.0401	0.0001	0.0041	0.0001	0.0023
MCOMPTIM	0.80305	0.84890	0.79226	0.92739	0.86311	0.87338	0.77559
MEAN COMPILATION TIME (SECS)	0.0017	0.0005	0.0021	0.0001	0.0003	0.0002	0.0030

For Comsc 4323 and 5323, the compilation times were not collected. However, for Comsc 5323, lines of code and most of the Software Science metrics correlate strongly with the object code size.

TABLE V

Correlations Between Selected Static and Dynamic Measurements (cont.)

COMSC 4323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 7

	LOC	VG	ETA1	N1	ETA2	N2	EFFORT
MEXECTIM	-0.75873	-0.72151	0.10956	-0.59434	0.27268	-0.72264	-0.91113
MEAN EXECUTION TIME (SECS)	0.0480	0.0672	0.8151	0.1593	0.5541	0.0666	0.0043
MEXECMEM	-0.60080	-0.68249	-0.56914	0.01364	0.41535	0.08485	-0.27882
MEAN EXECUTION MEMORY (BYTES)	0.1537	0.0911	0.1824	0.9769	0.3541	0.8565	0.5448
MOBJSIZE	-0.37538	-0.47097	-0.72913	0.24993	0.37236	0.33883	-0.02201
MEAN OBJECT CODE SIZE (BYTES)	0.4067	0.2861	0.0630	0.5888	0.4108	0.4572	0.9626
MUTIME	-0.70998	-0.65766	-0.20527	-0.38267	0.46377	-0.51279	-0.80458
MEAN USER TIME (SECS)	0.0739	0.1084	0.6588	0.3969	0.2945	0.2392	0.0291
MSYSTIME	-0.22935	-0.26423	0.84467	-0.63706	-0.46763	-0.64932	-0.40199
MEAN SYSTEM TIME (SECS)	0.6208	0.5669	0.0168	0.1239	0.2900	0.1145	0.3713
MELAPSED	-0.80762	-0.76006	0.06167	-0.58588	0.25938	-0.68869	-0.88697
MEAN ELAPSED TIME (H:MM:SS)	0.0280	0.0474	0.8955	0.1669	0.5743	0.0871	0.0078

COMSC 5323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 10

	LOC	VG	ETA1	N1	ETA2	N2	EFFORT
MEXECTIM	0.39264	-0.03804	0.32222	0.47060	0.02480	0.49021	0.75197
MEAN EXECUTION TIME (SECS)	0.2617	0.9169	0.3639	0.1698	0.9458	0.1503	0.0121
MEXECMEM	-0.05618	-0.27078	-0.08090	-0.43047	0.03585	-0.60619	-0.57579
MEAN EXECUTION MEMORY (BYTES)	0.8775	0.4492	0.8242	0.2143	0.9217	0.0632	0.0815
MOBJSIZE	0.91191	0.52097	0.85045	0.85040	0.79478	0.50999	0.61760
MEAN OBJECT CODE SIZE (BYTES)	0.0002	0.1226	0.0018	0.0018	0.0060	0.1321	0.0571
MUTIME	0.38112	-0.25198	0.47382	0.12875	0.24250	-0.05501	0.23990
MEAN USER TIME (SECS)	0.2772	0.4825	0.1665	0.7230	0.4996	0.8800	0.5044
MSYSTIME	0.32843	0.05313	0.20590	0.52321	-0.06554	0.61972	0.82254
MEAN SYSTEM TIME (SECS)	0.3542	0.8841	0.5682	0.1207	0.8573	0.0560	0.0035
MELAPSED	0.41740	0.01687	0.31649	0.50821	0.02664	0.54209	0.78503
MEAN ELAPSED TIME (H:MM:SS)	0.2301	0.9631	0.3730	0.1337	0.9418	0.1055	0.0071

For Comsc 2133 and 5323, significant correlations (0.05 or less) exist between some of the static measurements and the mean execution times. For Comsc 2133, lines of code, McCabe's cyclomatic complexity, total number of operators, total number of operands, and effort correlate strongly with mean

execution time (see Figures 5 - 9). For Comsc 5323, effort correlates strongly with mean execution time (see Figure 10).

For Comsc 2123 and 4323, no significant correlations exist between the static measurements and the mean execution times. For 2123, the highest correlation (level of 0.07) is between mean execution time and number of unique operands.

Correlations using natural logarithm transformations of the measurements were computed. No new correlations of significance were revealed.

Other statistical methods such as principal components, factor analysis, and detailed regression analysis [SAS85a,b] were considered for this study. However, it was decided that the results of these methods would not be creditable unless data from larger (30 or more) sample sizes were used.

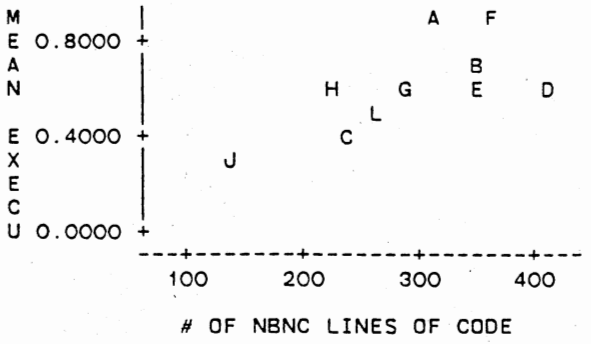

Figure 5. Plot of mean execution time vs. lines of code for Comsc 2133.

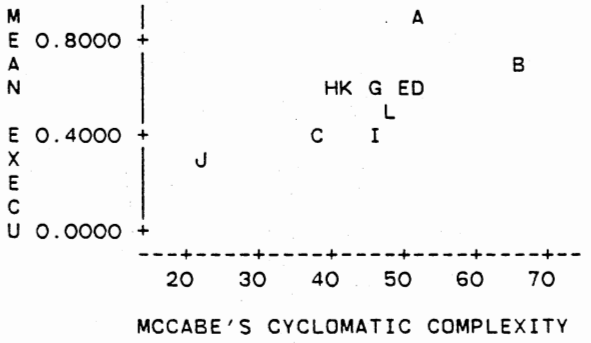

Figure 6. Plot of mean execution time vs. McCabe's cyclomatic complexity for Comsc 2133.

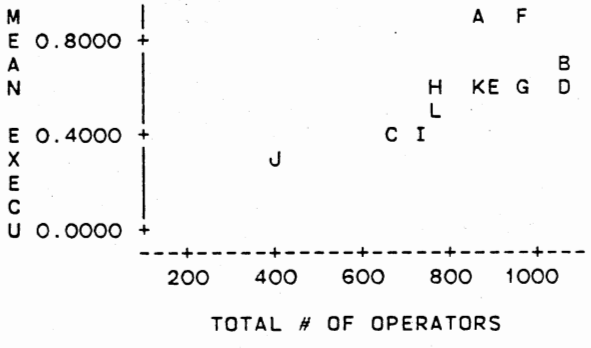

Figure 7. Plot of mean execution time vs. total number of operators for Comsc 2133.

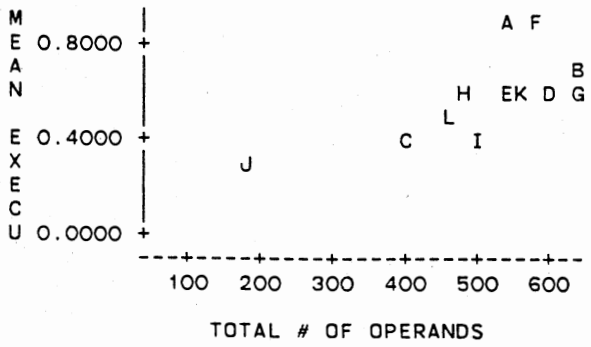

Figure 8. Plot of mean execution time vs. total number of operands for Comsc 2133.

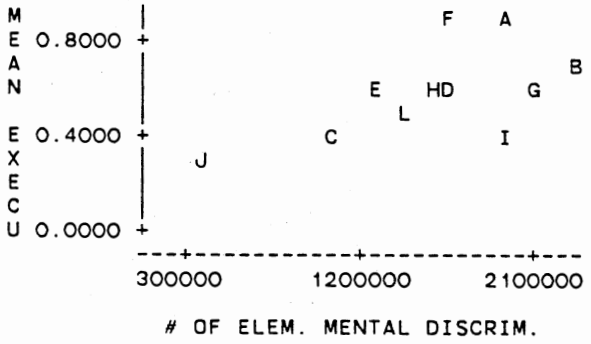

Figure 9. Plot of mean execution time vs. effort for Comsc 2133.

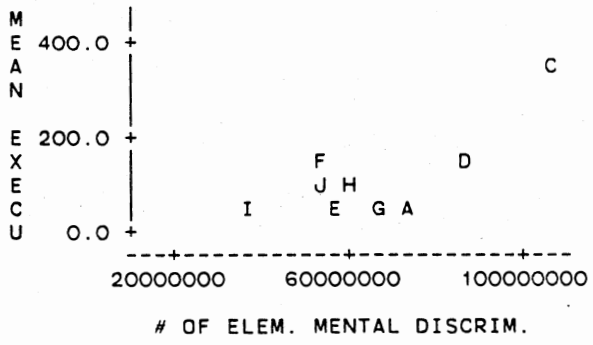

Figure 10. Plot of mean execution time vs. effort for Comsc 5323.

CHAPTER V

SUMMARY AND CONCLUSIONS

In summary, the measurements collected, even though from small sample sizes [Moher81], are reasonably representative of their populations. In addition, strong, positive correlations exist among the static measurements. Also, reasonable correlations exist among the dynamic measurements.

For Comsc 2133, significant correlations exist between the static measurements and mean execution time. For Comsc 5323, significant correlations exist between Halstead's effort metric and mean execution time. Other less significant correlations exist for these classes.

For Comsc 2123, the only significant correlation exists between the number of unique operands and mean execution time. One possible explanation why more correlations do not exist might be that the programming assignment was too simple to show any significant variations in code: the standard deviations of static measurements were not as great as those for 2133 (see Table II).

No significant correlations exist for the measurements of Comsc 4323. It seems that the sample size was too small for this class, thus explaining the unusual negative correlations that exist between these measurements.

In conclusion, there seems to be a relationship between some of the static complexity metrics and the dynamic measurements however slight they may be. Care should be taken not to read too much from the data [Mitchell87 and Weems88] nor to make any irrational generalizations [Brooks80]. As was stated in the introduction, this study is not a controlled experiment. It is an exploratory study carried out to reveal possible relationships between static complexity metrics and dynamic measurements of Pascal and C programs.

CHAPTER VI

FUTURE WORK

Suggestions for future work include conducting a similar, but controlled, experiment designed to test the null hypothesis that there is no variability in execution times for Pascal and/or C programs that can be explained by the static metrics used in this study. Obvious extensions to this include studies of other programming languages (maybe even a superset of C, such as C++ [Stroustrup84 and Rosler84]) and other studies using different metrics including Henry and Kafura's [Henry81] information flow metrics, Oviedo's scope measure [Oviedo80], Woodward *et al.*'s knot count [Woodward79], and the hybrid metric proposed by Li and Cheung [Li87]. These may correlate better with execution times.

In this same context, other studies could be done that examine programs written by the same students taking several courses over a period of time. This might help reduce variations in subjects [Brooks80]. The null hypothesis that no variability in subjects can be observed with respect to static and/or dynamic measurements over the period of time could be tested. Of course, the measurements would have to be standardized so that they could be compared, because the subjects (the student programs) would differ over the time period.

The programming assignments would be different, and other programming languages could be used as well.

Studies using the UNIX [UNIX86] commands **prof** and **gprof** could be done to examine detailed execution times and how they might possibly relate to certain complexity metrics. Possibly better correlations would result from using these more detailed times. The null hypothesis that no variability in execution times can be explained by the amount of CPU time spent at any particular level in a program could be tested.

Other work could be done to develop execution profilers for both the IBM and VAX systems. A good set of software tools would make it easier to conduct more extensive, controlled experiments. Finally, work could be done to develop a program to aid instructors in evaluating student programs by providing quantitative ratings of the programs based on their static and dynamic measurements.

CITED REFERENCES

[Basili83a]
V. R. Basili and D. H. Hutchens, "An Empirical Study of a Syntactic Complexity Family," *IEEE Trans. Software Eng.*, vol. SE-9, pp. 664-672, Nov. 1983.

[Basili86]
V. R. Basili, R. W. Selby, and D. H. Hutchens, "Experimentation in Software Engineering," *IEEE Trans. Software Eng.*, vol. SE-12, pp. 733-743, July 1986.

[Basili83b]
V. R. Basili, R. W. Selby, and T.-Y. Phillips, "Metric Analysis and Data Validation across Fortran Projects," *IEEE Trans. Software Eng.*, vol. SE-9, pp. 652-663, Nov. 1983.

[Basili75]
V. R. Basili and A. J. Turner, "Iterative Enhancement: A Practical Technique for Software Development," *IEEE Trans. Software Eng.*, vol. SE-1, pp. 390-396, Dec. 1975.

[Berge73]
C. Berge, *Graphs and Hypergraphs*, Elsevier North-Holland, Amsterdam, The Netherlands, 1973.

[Bishop87]
M. Bishop, "Profiling under UNIX by Patching," *Software--Practice & Experience*, vol. 17, pp. 729-739, Oct. 1987.

[Bonanni77]
L. E. Bonanni and A. L. Glasser, "SCCS/PWB User's Manual," Bell Telephone Laboratories, Inc., Nov. 1977.

[Boswell87]
F. D. Boswell, T. R. Grove, and E. W. Mackie, *Waterloo Pascal: Primer and Reference*, WATCOM, Ontario, Canada, 1987.

[Bourne78]
S. R. Bourne, "UNIX Time-Sharing System: The UNIX Shell," *Bell Syst. J.*, vol. 57, no. 6, pp. 1971-1989, July-Aug. 1978.

[Brooks80]
R. E. Brooks, "Studying Programmer Behavior Experimentally: The Problems of Proper Methodology," *Comm. ACM*, vol. 23, no. 4, pp. 207-213, 1980.

[Conte86]
S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models*, Benjamin/Cummings, Menlo Park, CA, 1986.

[Crawford85]
S. G. Crawford, A. A. McIntosh, and D. Pregibon, "An Analysis of Static Metrics and Faults in C Software," *J. Syst. Software*, vol. 5, pp. 37-48, 1985.

[da Cruz84]
Frank da Cruz, ed., *Kermit User Guide*, Columbia Univ. Center for Computing Activities, New York, July 21, 1984.

[Curtis84]
B. Curtis, "Fifteen Years of Psychology in Software Engineering: Individual Differences and Cognitive Science," *Proc. 7th Int. Conf. Software Eng.*, pp. 97-106, 1984.

[Davis88]
J. S. Davis and R. J. LeBlanc, "A Study of the Applicability of Complexity Measures," *IEEE Trans. Software Eng.*, vol. SE-14, pp. 1366-1372, Sept. 1988.

[Dolotta78]
T. A. Dolotta, R. C. Haight, and J. R. Mashey, "UNIX Time-Sharing System: The Programmer's Workbench," *Bell Syst. Tech. J.*, vol. 57, no. 6, pp. 2177-2200, July-Aug. 1978.

[Dowdy83]
S. M. Dowdy and S. Wearden, *Statistics for Research*, Wiley, New York, 1983.

[Elshoff76]
J. L. Elshoff, "An Analysis of Some Commercial PL/I Programs," *IEEE Trans. Software Eng.*, vol. SE-2, pp. 113-120, June 1976.

[Evangelist84]
W. M. Evangelist, "Program Complexity and Programming Style," *Proc. Int. Conf. Data Eng.*, Los Angeles, CA, Apr. 24-27, 1984.

[Evangelist83]
W. M. Evangelist, "Software Complexity Metric Sensitivity to Program Structuring Rules," *J. Syst. Software*, vol. 3, pp. 231-243, 1983.

[Feder84]
J. Feder, "The UNIX System: The Evolution of UNIX System Performance," *AT&T Bell Lab. Tech. J.*, vol. 63, no. 8, pp. 1791-1814, Oct. 1984.

[Findlay87]
W. Findlay and D. A. Watt, *Pascal: An Introduction to Methodical Programming*, Computer Science Press, Rockville, MD, 1987.

[de Freitas78]
S. L. de Freitas and P. J. Lavelle, "A Method for the Time Analysis of Programs," *IBM Syst. J.*, vol. 17, no. 1, pp. 26-38, 1978.

[Goldberg86]
R. Goldberg, "Software Engineering: An Emerging Discipline," *IBM Syst. J.*, vol. 25, nos. 3 & 4, pp. 334-353, 1986.

[Gordon79]
R. D. Gordon, "Measuring Improvements in Program Clarity," *IEEE Trans. Software Eng.*, vol. SE-5, pp. 79-90, Mar. 1979.

[Graham83]
S. L. Graham, P. B. Kessler, and M. K. McKusick, "An Execution Profiler for Modular Programs," *Software--Practice & Experience*, vol. 13, pp. 671-685, 1983.

[Halstead79]
M. H. Halstead (Yovits, ed.), "Advances in Software Science," *Advances in Computers*, vol. 18, Academic Press, New York, pp. 119-172, 1979.

[Halstead77a]
M. H. Halstead, *Elements of Software Science*, Elsevier North-Holland, New York, 1977.

[Halstead77b]
M. H. Halstead, "On Lines of Code and Programming Productivity," *IBM Syst. J.*, vol. 16, no. 4, pp. 421-423, 1977.

[Henry81]
S. Henry and D. Kafura, "Software Structure Metrics Based on Information Flow," *IEEE Trans. Software Eng.*, vol. SE-7, pp. 510-518, Sept. 1981.

[Johnson82]
R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[Jones78]
T. C. Jones, "Measuring Programming Quality and Productivity," *IBM Syst. J.*, vol. 17, no. 1, pp. 39-63, 1978.

[Kernighan78]
B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[Lesk75]
M. E. Lesk and E. Schmidt, "LEX--A Lexical Analyzer Generator," Computer Science Technical Report No. 39, Bell Laboratories, Murray Hill, NJ 07974, 1975.

[Lew88]
K. S. Lew, T. S. Dillion, and K. E. Forward, "Software Complexity and Its Impact on Software Reliability," *IEEE Trans. Software Eng.*, vol. SE-14, pp. 1645-1655, Nov. 1988.

[Li87]
H. F. Li, and W. K. Cheung, "An Empirical Study of Software Metrics," *IEEE Trans. Software Eng.*, vol. SE-13, pp. 697-708, June 1987.

[Lind89]
R. K. Lind and K. Vairavan, "An Experimental Investigation of Software Metrics and Their Relationship to Software Development Effort," *IEEE Trans. Software Eng.*, vol. SE-15, pp. 649-653, May 1989.

[McCabe76]
T. J. McCabe, "A Complexity Measure," *IEEE Trans. Software Eng.*, vol. SE-2, pp. 308-320, Dec. 1976.

[Mitchell87]
J. Mitchell, J. E. Urban, and R. McDonald, "The Effect of Abstract Data Types on Program Development," *Computer*, pp. 85-88, Aug. 1987.

[Moher81]
T. Moher and C. M. Schneider, "Methods for Improving Controlled Experimentation in Software Engineering," *Proc. 5th Int. Conf. Software Eng.*, pp. 224-233, 1981.

[MVS87]
MVS/Extended Architecture Message Library: System Messages, Volumes 1 & 2. IBM Corp., Poughkeepsie, NY 12602, June 1987.

[OS78]
OS/VS Linkage Editor and Loader, VS1 Rel. 7, VS2 Rel. 3.8, IBM Corp., San Jose, CA 95150, Aug. 1978.

[Oviedo80]
E. I. Oviedo, "Control Flow, Data Flow and Program Complexity," *Proceedings of IEEE COMPSAC*, Chicago, IL, pp. 146-152, Nov. 1980.

[Plattner81]
B. Plattner and J. Nievergelt, "Monitoring Program Execution: A Survey," *Computer*, vol. 14, no. 11, pp. 76-93, Nov. 1981.

[Power83]
L. R. Power, "Design and Use of a Program Execution Analyzer," *IBM Syst. J.*, vol. 22, no. 3, 1983.

[Ritchie78a]
D. M. Ritchie, S. C. Johnson, M. E. Lesk, and B. W. Kernighan, "UNIX Time-Sharing System: The C Programming Language," *Bell Syst. Tech. J.*, vol. 57, no. 6, pp. 1991-2019, July-Aug. 1978.

[Ritchie78b]
D. M. Ritchie and K. Thompson, "The UNIX Time-Sharing System," *Bell Syst. Tech. J.*, vol. 57, no. 6, pp. 1905-1929, July-Aug. 1978.

[Rosler84]
L. Rosler, "The UNIX System: The Evolution of C--Past and Future," *AT&T Bell Lab. Tech. J.*, vol. 63, no. 8, pp. 1685-1699, Oct. 1984.

[SAS85a]
SAS User's Guide: Basics, Version 5 Edition, SAS Inst. Inc., Box 8000, Cary, NC 27511, 1985.

[SAS85b]
SAS User's Guide: Statistics, Version 5 Edition, SAS Inst. Inc., Box 8000, Cary, NC 27511, 1985.

[Shaw89]
W. H. Shaw, Jr., J. W. Howatt, R. S. Maness, and D. M. Miller, "A Software Science Model of Compile Time," *IEEE Trans. Software Eng.*, vol. SE-15, pp. 543-549, May 1989.

[Shen83]
V. Y. Shen, S. D. Conte, and H. E. Dunsmore, "Software Science Revisited: A Critical Analysis of the Theory and Its Empirical Support," *IEEE Trans. Software Eng.*, vol. SE-9, no. 2, pp. 155-165, 1983.

[Snedecor80]
G. W. Snedecor and W. G. Cochran, *Statistical Methods*, Iowa State University Press, Ames, Iowa, 1980.

[Sobell85]
M. G. Sobell, *A Practical Guide to UNIX System V*, Benjamin/Cummings, Menlo Park, CA, 1985.

[Stroustrup84]
B. Stroustrup, "The UNIX System: Data Abstraction in C," *AT&T Bell Lab. Tech. J.*, vol. 63, no. 8, pp. 1701-1732, Oct. 1984.

[Thompson78]
K. Thompson, "UNIX Time-Sharing System: UNIX Implementation," *Bell Syst. Tech. J.*, vol. 57, no. 6, pp. 1931-1945, July-Aug. 1978.

[Triola83]
M. F. Triola, *Elementary Statistics*, Benjamin/Cummings, Menlo Park, CA, 1983.

[TSO86]
TSO/E Clists: Implementation and Reference, IBM Corp., Poughkeepsie, NY 12602, Sept. 1986.

[Tucker84]
A. Tucker, *Applied Combinatorics*, Wiley, New York, 1984.

[ULTRIX87]
ULTRIX-32 Programmer's Manual, sections 1-7, Digital Equipment Corp., Merrimack, NH 03054, 1987.

[UNIX86]
The UNIX System User's Manual, AT&T, Prentice-Hall, Englewood Cliffs, NJ, 1986.

[VSPascal87a]
VS Pascal Application Programming Guide, IBM Corp., San Jose, CA 95150, June 1987.

[VSPascal87b]
VS Pascal Diagnosis Guide and Reference, IBM Corp., San Jose, CA 95150, June 1987.

[VSPascal87c]
VS Pascal General Information, rel. 2, IBM Corp., San Jose, CA 95150, September 1988.

[VSPascal87d]
VS Pascal Language Reference, IBM Corp., San Jose, CA 95150, June 1987.

[Walston77]
C. E. Walston and C. P. Felix, "A Method of Programming Measurement and Estimation," *IBM Syst. J.*, vol. 16, no. 1, pp. 54-73, 1977.

[Weems88]
C. Weems, "Does Method Invalidate Results?," *Computer*, pp. 6-8, Apr. 1988.

[Weinberger84]
P. J. Weinberger, "The UNIX System: Cheap Dynamic Instruction Counting," *AT&T Bell Lab. Tech. J.*, vol. 63, no. 8, pp. 1815-1825, Oct. 1984.

[Weiser84]
M. Weiser, "Program Slicing," *IEEE Trans. Software Eng.*, vol. SE-10, pp. 352-357, July 1984.

[Weyuker88]
E. J. Weyuker, "Evaluating Software Complexity Measures," *IEEE Trans. Software Eng.*, vol. SE-14, pp. 1357-1365, Sept. 1988.

[Wiedenbeck85]
S. Wiedenbeck, "Novice/Expert Differences in Programming Skill," *Int. J. Man-Machine Studies*, vol. 23, pp. 383-390, 1985.

[Woodward79]
M. R. Woodward, M. A. Hennell, and D. Hedley, "A Measure of Control Flow Complexity in Program Text," *IEEE Trans. Software Eng.*, vol. 5, pp. 45-50, Jan. 1979.

GENERAL REFERENCES

[Aho86]
A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers, Principles, Techniques, and Tools*, Prentice-Hall, Reading, MA, 1986.

[Deitel84]
H. M. Deitel, *An Introduction to Operating Systems*, Addison-Wesley, Reading, MA, 1984.

[Dyer80]
M. Dyer, "The Management of Software Engineering, Part IV: Software Development Practices," *IBM Syst. J.*, vol. 19, no. 4, pp. 451-465, 1980.

[Gries71]
D. Gries, *Compiler Construction For Digital Computers*, Wiley, New York, 1971.

[Johnson84]
S. C. Johnson and D. M. Ritchie, "UNIX Time-Sharing System: Portability of C Programs and the UNIX System," *Bell Syst. Tech. J.*, vol. 57, no. 6, pp. 2021-2048, July-Aug. 1984.

[Mills80]
H. D. Mills, "The Management of Software Engineering, Part I: Principles of Software Engineering," *IBM Syst. J.*, vol. 19, no. 4, pp. 26-38, 1980.

[Ostle63]
B. Ostle, *Statistics in Research; Basic Concepts and Techniques for Research Workers*, Iowa State University Press, Ames, Iowa, 1963.

[Peterson85]
J. L. Peterson and A. Silberschatz, *Operating System Concepts*, Addison-Wesley, Reading, MA, 1985.

[Pike84]
R. Pike and B. W. Kernighan, "The UNIX System: Program Design in the UNIX Environment," *AT&T Bell Lab. Tech. J.*, vol. 63, no. 8, pp. 1595-1605, Oct. 1984.

[Ritchie84a]
D. M. Ritchie, "The UNIX System: The Evolution of the UNIX Time-Sharing System," *AT&T Bell Lab. Tech. J.*, vol. 63, no. 8, pp. 1577-1593, Oct. 1984.

[Ritchie84b]
D. M. Ritchie, "The UNIX System: A Stream Input-Output System," *AT&T Bell Lab. Tech. J.*, vol. 63, no. 8, pp. 1897-1910, Oct. 1984.

[Ritchie78]
D. M. Ritchie, "UNIX Time-Sharing System: A Retrospective," *Bell Syst. Tech. J.*, vol. 57, no. 6, pp. 1947-1969, July-Aug. 1978.

[Sahni85]
S. Sahni, *Concepts in Discrete Mathematics*, Camelot, Fridley, MN, 1985.

[Zolnowski81]
J. C. Zolnowski and D. B. Simmons, "Taking the measure of program complexity," *Proc. of the Nat'l. Computer Conf.*, pp. 329-336, 1981.

APPENDIXES

APPENDIX A

PASCAL COUNTING STRATEGY

Pascal Counting Strategy[1]

1. All of the program including statement parts, program heading, and declaration parts should be considered. The only statements ignored should be comments.

2. Variables, constants (including the standard constants **FALSE**, **TRUE**, and **MAXINT**), user-defined types, literals, file names, and the reserved word **NIL** are counted as operands. All operands are counted as if they were global in scope. In other words, local variables with the same name in different procedures are counted as multiple occurrences of the same operand.

3. The following entities are always counted as single operators (* is not differentiated between set and arithmetic use):

*	/	DIV	MOD	<	<=
;	<>	>=	>	:=	^
,	..	NOT	AND	OR	IN
PACKED	TO	DOWNTO	INTEGER	REAL	TEXT
CHAR	BOOLEAN	LABEL	PROGRAM	FUNCTION	
FORWARD				PROCEDURE	
EXTERNAL_CW		EXTERNAL_PVS		EXTERNAL_OS	

4. The following multiple entities are counted as single operators.

SET OF	FILE OF	ARRAY OF	RECORD END
BEGIN END	CASE END	WHILE DO	FOR DO
REPEAT UNTIL	WITH DO	IF THEN	IF THEN ELSE

---

[1] This counting strategy is adopted from Conte *et al.* [Conte86] with modifications to make it applicable to WPascal [Boswell87].

5. The following entities or pairs of entities are counted as single operators subject to the accompanying conditions:

VAR	is counted as an operator in parameter lists and is not counted as a section label.
=	is counted as either a relational operator in expressions or a definition operator in non-executable sections of the program.
+	is counted as either a unary + or binary + depending on its function. The binary + is not differentiated between arithmetic and set usage.
-	is counted as either a unary - or binary - depending on its function. The binary - is not differentiated between arithmetic and set usage.
.	is counted as either a record component selector symbol or a program terminator depending on its function.
:	is a definition operator in the **VAR** section and parameter lists. It is a separation operator following **CASE** or **GOTO** labels.
( )	is counted as either an argument list operator or expression operator depending on the function.
[ ]	is counted as either a subscript operator or set operator depending on the function. The use of (. .) is equivalent.

6. Procedure and function calls are counted as operators. The subprogram name following **FUNCTION** or **PROCEDURE** is not counted, though it actually is the operand for the **FUNCTION** or **PROCEDURE** operator.

7. Goto statements (i.e., **GOTO** and an accompanying label) are counted as the operator **GOTO** and the operand *label.*

8. Declarations of labels are not enumerated--all tokens after the **LABEL** operator through the next semicolon (inclusive) are ignored.

9. The following are syntactic devices and are not counted:

CONST **TYPE** **VAR** (for variable sections)

10. (not applicable to WPascal)

11. McCabe's VG metric is counted as follows:

increment on keywords:

WHILE **FOR** **REPEAT** **IF**
AND **OR** **PROCEDURE** **FUNCTION**
PROGRAM

AND and **OR** include loops/branches controlled by Boolean variables. Conditionals couldn't be counted directly as Boolean functions and variables would not be counted correctly. (I interpreted this to mean that VG is to be incremented on only **AND** and **OR** operators in conditionals and not those in assignment statements such as **A := B AND C OR D;**.)

also count **CASE** labels by incrementing on:

colons in the executable part of the program but outside a **WRITE(LN)** parameter list.

commas in a **CASE** label list (on scanning a colon, add the number of commas since the last semicolon).

and decrementing on:

LABEL keyword.
commas in a **LABEL** statement.

The decrement is necessary to remove the **GOTO** labels, if any.

12. The number of variables is computed by subtracting the number of numeric constants and character strings in the program from the eta2 count. (I used a slightly different technique for counting the number of variables: the number of unique operands that occur between a semicolon or left parenthesis and a colon in a **VAR** section or in a formal parameter list.)

APPENDIX B

C COUNTING STRATEGY

C Counting Strategy[1]

1. All of the program including statement parts and preprocessor directives should be considered. The only statements ignored should be comments.

2. Variables, constants (including the standard constants **EOF** and **NULL**), user-defined types, literals, and file names are counted as operands. All operands are counted as if they were global in scope. In other words, local variables with the same name in different procedures are counted as multiple occurrences of the same operand.

3. The following entities are counted as single operators:

+	-	*	/	%	~
!	&	|	^	++	--
+=	-=	*=	/=	%=	&=
|=	^=	=	<<	>>	<<=
>>=	==	!=	<	>	<=
>=	&&	||	,	->	.
?	:	**main**	**if**	**else**	**for**
while	**do**	**switch**	**case**	**break**	**goto**
default	**return**	**continue**	**sizeof**	**typedef**	**int**
float	**char**	**short**	**long**	**double**	**auto**
unsigned	**struct**	**union**	**static**	**extern**	**FILE**
register	;	**#define**	**#include**	**#ifdef**	**#if**
#ifndef	**#else**	**#endif**	**#line**		

4. The following pairs of entities are counted as single operators:

{ } ( ) [ ]

---

[1] Modelled after strategy presented in [Conte86].

5. Function calls are counted as operators. The function name appearing on the definition of a function is neither counted as an operator nor an operand.

6. Goto statements (i.e., **goto** and an accompanying label) are counted as the operator **goto** and the operand *label.*

7. Goto labels, *label***:**, are counted as the operand *label* and the operator **:**.

8. McCabe's VG metric is counted as follows:

   increment on keywords and symbols:

   while **case** **default** **for** **main** **if**
   && **||**

   && and **||** cause VG to be incremented only if they are part of a loop or branch condition. This means that VG would not be incremented on **&&** and **||** operators in assignment statements such as **a = b && c || d;**.

   Also increment VG on the definition of each function.

9. The number of constants is computed by counting the number of **#define**s.

APPENDIX C

VS PASCAL PROGRAM USED TO COMPUTE THE STATIC SOFTWARE COMPLEXITY METRICS OF THE WPASCAL PROGRAMS

```
PROGRAM METRICS (INPUT,OUTPUT,OUTFILE);                                  0001
(*=========================================================*)           0002
(*             S O F T W A R E   M E T R I C S             *)           0003
(*=========================================================*)           0004
(*                                                         *)           0005
(*   SOFTWARE METRICS PROGRAM DESIGNED AND WRITTEN BY      *)           0006
(*   KEITH E. MOLL                                         *)           0007
(*   MAY 1, 1989                                           *)           0008
(*   COMSC 5000 - THESIS                                   *)           0009
(*   ADVISER: DR. MANSUR SAMADZADEH                        *)           0010
(*                                                         *)           0011
(*=========================================================*)           0012
(*                                                         *)           0013
(*   PROGRAM DESCRIPTION:                                  *)           0014
(*                                                         *)           0015
(*   THIS PROGRAM WILL COMPUTE LINES OF CODE (LOC),        *)           0016
(*   MCCABE'S CYCLOMATIC NUMBER, AND HALSTEAD'S            *)           0017
(*   SOFTWARE SCIENCE METRICS ON PASCAL PROGRAMS           *)           0018
(*   WRITTEN IN WPASCAL.                                   *)           0019
(*                                                         *)           0020
(*   LIMITATIONS:                                          *)           0021
(*                                                         *)           0022
(*   THIS PROGRAM WILL ONLY RECOGNIZE THOSE KEYWORDS       *)           0023
(*   AND SPECIAL SYMBOLS DEFINED BELOW AS OPERATORS.       *)           0024
(*   ALL OTHER TOKENS, UNLESS IMMEDIATELY FOLLOWED         *)           0025
(*   BY A LEFT PARENTHESIS IN WHICH CASE IT IS AN          *)           0026
(*   OPERATOR, ARE CONSIDERED OPERANDS.                    *)           0027
(*                                                         *)           0028
(*=========================================================*)           0029
                                                                         0030
CONST                                                                    0031
   (* WPASCAL KEYWORDS *)                                               0032
   TAND = 'AND';                                                         0033
   TARRAY = 'ARRAY';                                                     0034
   TBEGIN = 'BEGIN';                                                     0035
   TCASE = 'CASE';                                                       0036
   TCONST = 'CONST';                                                     0037
   TDIV = 'DIV';                                                         0038
   TDO = 'DO';                                                           0039
   TDOWNTO = 'DOWNTO';                                                   0040
   TELSE = 'ELSE';                                                       0041
   TEND = 'END';                                                         0042
   TFILE = 'FILE';                                                       0043
   TFOR = 'FOR';                                                         0044
   TFUNCTION = 'FUNCTION';                                               0045
   TGOTO = 'GOTO';                                                       0046
   TIF = 'IF';                                                           0047
   TIN = 'IN';                                                           0048
   TLABEL = 'LABEL';                                                     0049
   TMOD = 'MOD';                                                         0050
   TNIL = 'NIL';                                                         0051
   TNOT = 'NOT';                                                         0052
   TOF = 'OF';                                                           0053
   TOR = 'OR';                                                           0054
   TPACKED = 'PACKED';                                                   0055
```

```
TPROCEDURE = 'PROCEDURE';                                                    0056
TPROGRAM = 'PROGRAM';                                                        0057
TRECORD = 'RECORD';                                                          0058
TREPEAT = 'REPEAT';                                                          0059
TSET = 'SET';                                                                0060
TTHEN = 'THEN';                                                              0061
TTO = 'TO';                                                                  0062
TTYPE = 'TYPE';                                                              0063
TUNTIL = 'UNTIL';                                                            0064
TVAR = 'VAR';                                                                0065
TWHILE = 'WHILE';                                                            0066
TWITH = 'WITH';                                                              0067
                                                                             0068
(* SPECIAL EXTENSIONS TO WPASCAL *)                                          0069
TFORWARD = 'FORWARD';   (* FOR FORWARD REFERENCES *)                         0070
TEXTERNAL_CW = 'EXTERNAL_CW';   (* WATERLOO C EXTERNAL SUBPROGRAM *)         0071
TEXTERNAL_PVS = 'EXTERNAL_PVS'; (* IBM PASCAL/VS EXT. SUBPROGRAM *)          0072
TEXTERNAL_OS = 'EXTERNAL_OS';   (* IBM 370 ASSEMBLER LANGUAGE *)             0073
                                                                             0074
(* PREDEFINED PASCAL DATA TYPES *)                                           0075
TINTEGER = 'INTEGER';                                                        0076
TREAL = 'REAL';                                                              0077
TCHAR = 'CHAR';                                                              0078
TBOOLEAN = 'BOOLEAN';                                                        0079
TTEXT = 'TEXT';                                                              0080
TMAXINT = 'MAXINT';                                                          0081
                                                                             0082
(* SOME OF THE STANDARD PASCAL FUNCTIONS AND PROCEDURES *)                   0083
TEOF = 'EOF';                                                                0084
TEOLN = 'EOLN';                                                              0085
TREADLN = 'READLN';                                                          0086
TWRITELN = 'WRITELN';                                                        0087
                                                                             0088
(* OPERATOR PAIRS *)                                                         0089
TWHILEDO = 'WHILE DO';                                                       0090
TIFTHEN = 'IF THEN';                                                         0091
TIFTHENELSE = 'IF THEN ELSE';                                                0092
TBEGINEND = 'BEGIN END';                                                     0093
TFORDO = 'FOR DO';                                                           0094
TREPEATUNTIL = 'REPEAT UNTIL';                                               0095
TCASEEND = 'CASE END';                                                       0096
TRECORDEND = 'RECORD END';                                                   0097
TSETOF = 'SET OF';                                                           0098
TFILEOF = 'FILE OF';                                                         0099
TARRAYOF = 'ARRAY OF';                                                       0100
TWITHDO = 'WITH DO';                                                         0101
TPARENS = '( )';                                                             0102
TBRACKETS = '(. .)';                                                         0103
                                                                             0104
(* DELIMITERS AND PUNCTUATION MARKS *)                                       0105
BLANK = ' ';                                                                 0106
COMMA = ',';                                                                 0107
PERIOD = '.';                                                                0108
DECIMALPT = '.';                                                             0109
SEMICOLON = ';';                                                             0110
```

```
COLON = ':';                                                              0111
APOSTROPHE = '''';   SQUOTE = APOSTROPHE;   (* SINGLE QUOTE *)          0112
DQUOTE = '"';    (* DOUBLE QUOTE *)                                      0113
LPAREN = '(';    (* LEFT PARENTHESIS *)                                  0114
RPAREN = ')';    (* RIGHT PARENTHESIS *)                                 0115
LBRACKET = '¢'; (* LEFT BRACKET *)                                       0116
RBRACKET = '!'; (* RIGHT BRACKET *)                                      0117
LARRAY = '(.';  (* LEFT ARRAY SUBSCRIPT BRACKET *)                       0118
RARRAY = '.)';  (* RIGHT ARRAY SUBSCRIPT BRACKET *)                      0119
BCOMI = '{';     (* BEGIN COMMENT TYPE I *)                              0120
ECOMI = '}';     (* END COMMENT TYPE I *)                                0121
BCOMII = '(*';  (* BEGIN COMMENT TYPE II *)                              0122
ECOMII = '*)';  (* END COMMENT TYPE II *)                               0123
DOTDOT = '..';  (* SUBRANGE SPECIFIER *)                                0124
PTR = '¬';       (* POINTER *)                                           0125
                                                                          0126
(* ARITHMETIC AND SET OPERATORS *)                                        0127
ASSIGNMENT = ':=';                                                        0128
ASTERISK = '*';    (* MULTIPLICATION AND SET INTERSECTION *)             0129
TIMES = '*';                                                              0130
SLASH = '/';       (* REAL DIVISION *)                                   0131
PLUS = '+';     (* BOTH UNARY AND BINARY ADDITION AND SET UNION *)       0132
MINUS = '-';    (* BOTH UNARY AND BINARY SUBTRACTION AND                 0133
                   SET DIFFERENCE *)                                      0134
                                                                          0135
(* RELATIONAL OPERATORS *)                                                0136
EQ = '=';     (* EQUAL TO *)                                             0137
LT = '<';     (* LESS THAN *)                                            0138
LE = '<=';    (* LESS THAN OR EQUAL TO *)                                0139
NE = '<>';    (* NOT EQUAL *)                                            0140
GT = '>';     (* GREATER THAN *)                                         0141
GE = '>=';    (* GREATER THAN OR EQUAL TO *)                             0142
                                                                          0143
(* TOKENS MARKING START AND END OF PASCAL PROGRAM *)                      0144
T$JOB = '$JOB';                                                           0145
T$ENTRY = '$ENTRY';                                                       0146
                                                                          0147
MAXLINELEN = 72; (* MAX. LINE LENGTH: THERE SHOULD NOT BE                0148
                    ANY LINES IN THE PROGRAM BEING ANALYZED               0149
                    LONGER THAN THIS *)                                   0150
MAXTOKLEN = 20;  (* MAX. TOKEN LENGTH: ALL TOKENS MUST BE SHORTER        0151
                    THAN OR EQUAL TO THIS; MUST LESS                      0152
                    THAN OR EQUAL TO MAXLINELEN *)                        0153
MAXOPR = 100;     (* MAXIMUM NUMBER OF OPERATORS THAT CAN BE            0154
                    HANDLED BY THIS PROGRAM *)                            0155
MAXOPD = 400;     (* MAXIMUM NUMBER OF OPERANDS THAT CAN BE             0156
                    HANDLED BY THIS PROGRAM *)                            0157
MAXSUB = 50;      (* MAXIMUM NUMBER OF SUBPROGRAMS THAT CAN             0158
                    BE HANDLED BY THIS PROGRAM *)                         0159
YES = TRUE;                                                               0160
NO = FALSE;                                                               0161
DEBUG = TRUE;                                                             0162
                                                                          0163
NOTFOUND = -1;    (* NOT FOUND INDEX FLAG VALUE *)                       0164
                                                                          0165
```

```
   (* WARNING NUMBERS *)                                                    0166
   OPRFULL = 1;    (* OPERATOR ARRAYS FULL *)                              0167
   OPDFULL = 2;    (* OPERAND ARRAYS FULL *)                               0168
   SUBPGMFULL = 3;    (* SUBPROGRAM ARRAY FULL *)                          0169
                                                                           0170
TYPE                                                                       0171
   NONNEG = 0..MAXINT;                                                     0172
                                                                           0173
VAR                                                                        0174
   OUTFILE : TEXT;    (* OUTPUT FILE TO WHICH METRICS ARE WRITTEN *)       0175
   LINE : STRING(MAXLINELEN); (* PHYSICAL PROGRAM LINE *)                 0176
   TOK,                       (* CURRENT TOKEN *)                         0177
   PREVTOK,                   (* PREVIOUS TOKEN *)                        0178
   TEMPTOK                    (* TEMPORARY TOKEN *)                       0179
      : STRING(MAXTOKLEN);                                                 0180
   OPR : ARRAY (.1..MAXOPR.) OF STRING(MAXTOKLEN); (* OPERATORS *)        0181
   NOPR : ARRAY (.1..MAXOPR.) OF INTEGER;  (* OPERATOR FREQUENCIES *)     0182
   OPD : ARRAY (.1..MAXOPD.) OF STRING(MAXTOKLEN); (* OPERANDS *)         0183
   NOPD : ARRAY (.1..MAXOPD.) OF INTEGER;  (* OPERAND FREQUENCIES *)      0184
   SUBPGM : ARRAY (.1..MAXSUB.) OF STRING(MAXTOKLEN); (* USER-DEFINED *)0185
                                                 (* SUBPROGRAM NAMES *) 0186
   NSUB : INTEGER;    (* NUMBER OF SUBPROGRAMS *)                         0187
   (* SOFTWARE SCIENCE METRICS *)                                         0188
   PLEN,    (* PROGRAM LENGTH *)                                         0189
   ETA1,    (* NUMBER OF UNIQUE OPERATORS *)                             0190
   N1,      (* TOTAL NUMBER OF OPERATORS *)                              0191
   ETA2,    (* NUMBER OF UNIQUE OPERANDS *)                              0192
   N2       (* TOTAL NUMBER OF OPERANDS *)                               0193
      : NONNEG;                                                          0194
   EPLEN,   (* ESTIMATED PROGRAM LENGTH *)                               0195
   PVOL,    (* PROGRAM VOLUME *)                                         0196
   EPLEV,   (* ESTIMATED PROGRAM LEVEL *)                                0197
   EPDIF,   (* ESTIMATED PROGRAM DIFFICULTY *)                           0198
   EPPVOL,  (* ESTIMATED POTENTIAL PROGRAM VOLUME *)                     0199
   EFFORT,  (* NUMBER OF ELEMENTARY MENTAL DISCRIMINATIONS *)            0200
   TIME,    (* IMPLEMENTATION TIME *)                                    0201
   LAMBDA   (* LANGUAGE LEVEL *)                                         0202
      : REAL;                                                            0203
   (* FLAGS *)                                                           0204
   FCONST,     (* SET IF IN CONST SECTION *)                            0205
   FCONSTANT,  (* SET IF TOKEN IS A CONSTANT: THIS IS USED TO *)         0206
               (* DISTINGUISH BETWEEN IDENTIFIERS ON THE LEFTHAND *)     0207
               (* SIDE OF THE EQUAL SIGN AND THEIR VALUES ON THE *)      0208
               (* RIGHTHAND SIDE IN THE CONST SECTION SO THAT ONLY *)    0209
               (* THE ACTUAL CONSTANTS (THE IDENTIFIERS) WILL BE *)      0210
               (* COUNTED AS CONSTANTS (NCONST). *)                      0211
   FTYPE,      (* SET IF IN TYPE SECTION *)                             0212
   FTTYPE,     (* SET IF TOKEN IS A TYPE: THIS IS USED TO *)            0213
               (* DISTINGUISH BETWEEN IDENTIFIERS ON THE LEFTHAND *)     0214
               (* SIDE OF THE EQUAL SIGN AND ASSIGNED TYPES ON THE *)    0215
               (* RIGHTHAND SIDE IN THE TYPE SECTION SO THAT ONLY *)     0216
               (* THE ACTUAL TYPES (THE IDENTIFIERS) WILL BE *)          0217
               (* COUNTED AS TYPES (NTYPE). *)                           0218
   FVAR,       (* SET IF IN VAR SECTION *)                              0219
   FVARIABLE,  (* SET IF TOKEN IS A VARIABLE: THIS IS USED TO *)         0220
```

```
                (* DISTINGUISH BETWEEN IDENTIFIERS ON THE LEFTHAND *)      0221
                (* SIDE OF THE COLON AND THEIR DATA TYPES ON THE *)         0222
                (* RIGHTHAND SIDE IN THE VAR SECTION SO THAT ONLY *)        0223
                (* THE ACTUAL VARIABLES (THE IDENTIFIERS) WILL BE *)        0224
                (* COUNTED AS VARIABLES (NVAR). *)                         0225
    FFUNCTION,  (* SET IF FUNCTION KEYWORD FOUND SO THAT FUNCTION *)        0226
                (* NAME WILL NOT BE COUNTED AS AN OPERAND *)                0227
    FPROCEDURE, (* SET IF PROCEDURE KEYWORD FOUND SO THAT PROCEDURE *)     0228
                (* NAME WILL NOT BE COUNTED AS AN OPERAND *)                0229
    FPROGRAM,   (* SET IF PROGRAM KEYWORD FOUND SO THAT PROGRAM NAME *)     0230
                (* WILL NOT BE COUNTED AS AN OPERAND *)                     0231
    FREAL,      (* SET IF REAL NUMBER PROCESSED *)                         0232
    FCOND,      (* SET IF PROCESSING CONDITIONAL *)                        0233
    FLABEL,     (* SET IF PROCESSING LABEL SECTION *)                      0234
    FSQUOTE,    (* SET IF PROCESSING A STRING DELIMITED BY                 0235
                   SINGLE QUOTES *)                                        0236
    FDQUOTE,    (* SET IF PROCESSING A STRING DELIMITED BY                 0237
                   DOUBLE QUOTES *)                                        0238
    FCOM        (* SET IF PROCESSING A COMMENT *)                          0239
       : BOOLEAN;                                                         0240
    FCASE,      (* INCR. ON CASE; DECR. ON END *)                          0241
    FPAREN      (* INCR. ON LPAREN; DECR. ON RPAREN *)                     0242
       : NONNEG;                                                          0243
    LOC,        (* LOC METRIC *)                                           0244
    POS,        (* POSITION IN LINE STRING WHERE SEARCH FOR THE            0245
                   NEXT TOKEN BEGINS *)                                    0246
    SAVEPOS     (* TO SAVE POSITION VALUE FOR BACKTRACKING *)              0247
       : NONNEG;                                                          0248
    INC         (* INCREMENTING VALUE: 0 FOR BLANK OR COMMENT LINES,       0249
                                       1 OTHERWISE *)                     0250
       : 0..1;                                                            0251
    VG : INTEGER;  (* MCCABE'S CYCLOMATIC NUMBER *)                       0252
    NCONST,    (* NUMBER OF CONSTANSTS *)                                 0253
    NTYPE,     (* NUMBER OF TYPES *)                                      0254
    NVAR       (* NUMBER OF VARIABLES *)                                  0255
       : NONNEG;                                                          0256
    I, J, K    (* GENERIC INDEX VARIABLES *)                              0257
       : INTEGER;                                                         0258
                                                                          0259
                                                                          0260
(*================================================================*)      0261
(*                                                                *)      0262
(*   WARNING:                                                     *)      0263
(*                                                                *)      0264
(*   THIS PROCEDURE REPORTS WARNING MESSAGES GIVEN WARNING NUMBER *)      0265
(*   WARNUM.                                                      *)      0266
(*                                                                *)      0267
(*================================================================*)      0268
                                                                          0269
PROCEDURE WARNING (WARNUM : INTEGER);                                     0270
   BEGIN                                                                  0271
      CASE WARNUM OF                                                      0272
      OPRFULL: WRITELN('WARNING: OPERATOR ARRAYS ARE FULL');              0273
      OPDFULL: WRITELN('WARNING: OPERAND ARRAYS ARE FULL');               0274
      SUBPGMFULL: WRITELN('WARNING: SUBPROGRAM ARRAY IS FULL');           0275
```

```
      END;                                                                  0276
   END;                                                                     0277
                                                                            0278
                                                                            0279
(*=========================================================================*)  0280
(*                                                                       *)  0281
(*   TOUPPER:                                                            *)  0282
(*                                                                       *)  0283
(*   THIS PROCEDURE CONVERTS ALL LETTERS IN THE GIVEN STRING TO          *)  0284
(*   UPPERCASE.  NUMBERS AND SPECIAL SYMBOLS ARE UNAFFECTED.             *)  0285
(*                                                                       *)  0286
(*=========================================================================*)  0287
                                                                            0288
PROCEDURE TOUPPER (VAR TOK : STRING(MAXTOKLEN));                            0289
   VAR                                                                      0290
      I : INTEGER;                                                          0291
   BEGIN                                                                    0292
      FOR I := 1 TO LENGTH(TOK) DO BEGIN                                    0293
         (* IF THIS WERE TO BE RUN ON A SYSTEM USING THE ASCII             0294
            CHARACTER SET, IT SHOULD WORK WITHOUT ALTERATION.  *)           0295
         IF (('a' <= TOK(.I.)) AND (TOK(.I.) <= 'i'))                       0296
            OR (('j' <= TOK(.I.)) AND (TOK(.I.) <= 'r'))                    0297
            OR (('s' <= TOK(.I.)) AND (TOK(.I.) <= 'z')) then               0298
            TOK(.I.) := CHR(ORD(TOK(.I.)) + ORD('A') - ORD('a'));           0299
      END;                                                                  0300
   END;                                                                     0301
                                                                            0302
                                                                            0303
(*=========================================================================*)  0304
(*                                                                       *)  0305
(*   ISNUMBER:                                                           *)  0306
(*                                                                       *)  0307
(*   THIS FUNCTION CHECKS TO SEE IF THE GIVEN STRING IS A NUMBER.        *)  0308
(*   A NUMBER IS ANY STRING OF DIGITS.                                   *)  0309
(*                                                                       *)  0310
(*=========================================================================*)  0311
                                                                            0312
FUNCTION ISNUMBER (VAR TOK : STRING(MAXTOKLEN)) : BOOLEAN;                  0313
   VAR                                                                      0314
      I : INTEGER;                                                          0315
      RESULT : BOOLEAN;                                                     0316
   BEGIN                                                                    0317
      RESULT := TRUE;   (* ASSUME IT IS A NUMBER *)                        0318
      FOR I := 1 TO LENGTH(TOK) DO                                          0319
         IF (TOK(.I.) < '0') OR ('9' < TOK(.I.)) THEN                       0320
            RESULT := FALSE;                                                0321
      ISNUMBER := RESULT;                                                   0322
   END;                                                                     0323
                                                                            0324
                                                                            0325
(*=========================================================================*)  0326
(*                                                                       *)  0327
(*   OPRSEARCH:                                                          *)  0328
(*                                                                       *)  0329
(*   THIS FUNCTION SEARCHES FOR THE GIVEN TOKEN WHICH IS AN              *)  0330
```

```
(*   OPERATOR AND RETURNS THE INDEX IF IT IS IN THE LIST.          *)  0331
(*                                                                 *)  0332
(*=================================================================*)  0333
                                                                       0334
FUNCTION OPRSEARCH (TOK : STRING(MAXTOKLEN); MAX : INTEGER) : INTEGER;  0335
   VAR                                                                 0336
      I : INTEGER;                                                     0337
      FOUND : BOOLEAN;                                                 0338
   BEGIN                                                               0339
      I := 0;                                                          0340
      FOUND := NO;                                                     0341
      WHILE (I < MAX) AND (NOT FOUND) DO BEGIN                         0342
         I := I + 1;                                                   0343
         FOUND := TOK = OPR(.I.);                                      0344
      END;                                                             0345
      IF (FOUND) THEN                                                  0346
         OPRSEARCH := I                                                0347
      ELSE                                                             0348
         OPRSEARCH := NOTFOUND;                                        0349
   END;                                                                0350
                                                                       0351
                                                                       0352
(*=================================================================*)  0353
(*                                                                 *)  0354
(*   PROCOPR:                                                      *)  0355
(*                                                                 *)  0356
(*   THIS PROCEDURE PROCESSES AN OPERATOR BY ADDING IT TO THE LIST  *)  0357
(*   OF OPERATORS IF NOT ALREADY IN THE LIST AND BY ADDING ONE     *)  0358
(*   TO ITS COUNT.                                                 *)  0359
(*                                                                 *)  0360
(*=================================================================*)  0361
                                                                       0362
PROCEDURE PROCOPR (TOK : STRING(MAXTOKLEN));                           0363
   VAR                                                                 0364
      I : INTEGER;                                                     0365
   BEGIN                                                               0366
      I := OPRSEARCH(TOK,ETA1);                                        0367
      IF (I = NOTFOUND) THEN BEGIN   (* ADD OPERATOR TO LIST *)        0368
         IF (ETA1 < MAXOPR) THEN BEGIN   (* IF LIST IS NOT FULL *)     0369
            ETA1 := ETA1 + 1;                                          0370
            I := ETA1;                                                 0371
            OPR(.I.) := TOK;                                           0372
            NOPR(.I.) := 1;                                            0373
            END                                                        0374
         ELSE   (* ELSE OPERATOR ARRAYS ARE FULL *)                    0375
            WARNING(OPRFULL);                                          0376
         END                                                           0377
      ELSE   (* ELSE OPERATOR ALREADY IN LIST; JUST COUNT *)           0378
         NOPR(.I.) := NOPR(.I.) + 1;   (* ANOTHER OCCURRENCE *)       0379
   END;                                                                0380
                                                                       0381
                                                                       0382
(*=================================================================*)  0383
(*                                                                 *)  0384
(*   OPDSEARCH:                                                    *)  0385
```

```
(*                                                                    *)  0386
(*   THIS FUNCTION SEARCHES FOR THE GIVEN TOKEN WHICH IS AN           *)  0387
(*   OPERAND AND RETURNS THE INDEX IF IT IS IN THE LIST.              *)  0388
(*                                                                    *)  0389
(*====================================================================*)  0390
                                                                          0391
FUNCTION OPDSEARCH (TOK : STRING(MAXTOKLEN); MAX : INTEGER) : INTEGER;    0392
   VAR                                                                    0393
      I : INTEGER;                                                        0394
      FOUND : BOOLEAN;                                                    0395
   BEGIN                                                                  0396
      I := 0;                                                             0397
      FOUND := NO;                                                        0398
      WHILE (I < MAX) AND (NOT FOUND) DO BEGIN                            0399
         I := I + 1;                                                      0400
         FOUND := TOK = OPD(.I.);                                         0401
      END;                                                                0402
      IF (FOUND) THEN                                                     0403
         OPDSEARCH := I                                                   0404
      ELSE                                                                0405
         OPDSEARCH := NOTFOUND;                                           0406
   END;                                                                   0407
                                                                          0408
                                                                          0409
(*====================================================================*)  0410
(*                                                                    *)  0411
(*   PROCOPD:                                                         *)  0412
(*                                                                    *)  0413
(*   THIS PROCEDURE PROCESSES AN OPERAND BY ADDING IT TO THE LIST     *)  0414
(*   OF OPERANDS IF NOT ALREADY IN THE LIST AND BY ADDING ONE TO      *)  0415
(*   ITS COUNT.                                                       *)  0416
(*                                                                    *)  0417
(*====================================================================*)  0418
                                                                          0419
PROCEDURE PROCOPD (TOK : STRING(MAXTOKLEN));                              0420
   VAR                                                                    0421
      I : INTEGER;                                                        0422
   BEGIN                                                                  0423
      I := OPDSEARCH(TOK,ETA2);                                           0424
      IF (I = NOTFOUND) THEN BEGIN    (* ADD OPERAND TO LIST *)           0425
         IF (ETA2 < MAXOPD) THEN BEGIN    (* IF LIST IS NOT FULL *)       0426
            ETA2 := ETA2 + 1;                                             0427
            I := ETA2;                                                    0428
            OPD(.I.) := TOK;                                              0429
            NOPD(.I.) := 1;                                               0430
            END                                                           0431
         ELSE    (* ELSE OPERAND ARRAYS ARE FULL *)                      0432
            WARNING(OPDFULL);                                             0433
         END                                                              0434
      ELSE    (* ELSE OPERAND ALREADY IN LIST; JUST COUNT *)             0435
         NOPD(.I.) := NOPD(.I.) + 1;    (* ANOTHER OCCURRENCE *)         0436
   END;                                                                   0437
                                                                          0438
                                                                          0439
(*====================================================================*)  0440
```

```
(*                                                                    *)  0441
(*   PROCSUBPGM:                                                      *)  0442
(*                                                                    *)  0443
(*   THIS PROCEDURE PROCESSES A SUBPROGRAM NAME.  IT ADDS THE NAME    *)  0444
(*   TO THE SUBPGM LIST IF NOT ALREADY IN IT.   TOK IS THE NAME.      *)  0445
(*                                                                    *)  0446
(*====================================================================*)  0447
                                                                           0448
PROCEDURE PROCSUBPGM (TOK : STRING(MAXTOKLEN));                            0449
   VAR                                                                     0450
      I : INTEGER;                                                         0451
      FSUBPGM : BOOLEAN;   (* SET IF NAME IS IN LIST *)                    0452
   BEGIN                                                                   0453
      FSUBPGM := FALSE;   (* ASSUME TOKEN IS NOT IN LIST *)                0454
      I := 0;                                                              0455
      WHILE (I < NSUB) DO BEGIN   (* LOOK FOR TOKEN IN *)                  0456
         I := I + 1;              (* SUBPROGRAM NAME LIST *)               0457
         IF (TOK = SUBPGM(.I.)) THEN BEGIN                                 0458
            I := NSUB;   (* EARLY EXIT *)                                 0459
            FSUBPGM := TRUE;                                               0460
         END;                                                              0461
      END;                                                                 0462
      IF (NOT FSUBPGM) THEN BEGIN   (* IF NAME NOT IN LIST & *)            0463
         IF (NSUB < MAXSUB) THEN BEGIN   (* LIST IS NOT FULL *)            0464
            NSUB := NSUB + 1;   (* THEN ADD SUBPGM NAME TO LIST *)          0465
            SUBPGM(.NSUB.) := TOK;                                         0466
            IF DEBUG THEN                                                  0467
               WRITELN('SUBPGM(.NSUB.)=',SUBPGM(.NSUB.));                  0468
            END                                                            0469
         ELSE   (* ELSE REPORT THAT SUBPGM ARRAY IS FULL *)                0470
            WARNING(SUBPGMFULL);                                           0471
      END;                                                                 0472
   END;                                                                    0473
                                                                           0474
                                                                           0475
(*====================================================================*)  0476
(*                                                                    *)  0477
(*   HALSTEAD'S OPERATOR/OPERAND TOKEN COUNT:                         *)  0478
(*                                                                    *)  0479
(*   THIS PROCEDURE COUNTS THE NUMBER OF OPERATORS AND OPERANDS       *)  0480
(*   IN THE PROGRAM.                                                  *)  0481
(*                                                                    *)  0482
(*====================================================================*)  0483
                                                                           0484
PROCEDURE TOKCOUNT (TOK : STRING(MAXTOKLEN));                              0485
   VAR                                                                     0486
      I,                                                                   0487
      NOP   (* NO OPERATION: THIS VARIABLE IS SET TO ZERO *)              0488
         : INTEGER;                                                        0489
   BEGIN                                                                   0490
      (* CHECK FOR OPERATORS AND OPERANDS *)                              0491
      IF (TOK = SEMICOLON) THEN BEGIN                                      0492
         PROCOPR(TOK);                                                     0493
         FVARIABLE := TRUE;   (* LOOK FOR VARIABLES, *)                   0494
         FCONSTANT := TRUE;   (* CONSTANTS, AND TYPES AFTER *)             0495
```

```
   FTTYPE := TRUE;       (* AFTER A SEMICOLON *)                    0496
   END                                                               0497
ELSE IF (TOK = ASSIGNMENT) THEN                                      0498
   PROCOPR(TOK)                                                      0499
ELSE IF (TOK = TBEGIN) THEN BEGIN                                    0500
   PROCOPR(TBEGINEND);                                               0501
   FLABEL := NO;                                                     0502
   FCONST := NO;                                                     0503
   FTYPE := NO;                                                      0504
   FVAR := NO;                                                       0505
   END                                                               0506
ELSE IF (TOK = EQ) THEN BEGIN   (* NO DISTINCTION BETWEEN *)         0507
   PROCOPR(TOK);          (* RELATIONAL OP. & DEFINITION OP. *)      0508
   FCONSTANT := FALSE;   (* ALL CONSTANTS IN CONST SECTION *)        0509
   FTTYPE := FALSE;      (* AND TYPES IN TYPE SECTION MUST *)        0510
   END                   (* APPEAR ON LEFTHAND SIDE OF EQUAL *)      0511
ELSE IF (TOK = NE) THEN                                              0512
   PROCOPR(TOK)                                                      0513
ELSE IF (TOK = LT) THEN                                              0514
   PROCOPR(TOK)                                                      0515
ELSE IF (TOK = LE) THEN                                              0516
   PROCOPR(TOK)                                                      0517
ELSE IF (TOK = GT) THEN                                              0518
   PROCOPR(TOK)                                                      0519
ELSE IF (TOK = GE) THEN                                              0520
   PROCOPR(TOK)                                                      0521
ELSE IF (TOK = PLUS) THEN   (* NO DISTINCTION BETWEEN UNARY +, *)    0522
   PROCOPR(TOK)             (* BINARY +, AND SET UNION *)           0523
ELSE IF (TOK = MINUS) THEN  (* NO DISTINCTION BETWEEN UNARY -, *)   0524
   PROCOPR(TOK)             (* BINARY -, AND SET DIFFERENCE *)      0525
ELSE IF (TOK = TIMES) THEN  (* NO DISTINCTION BETWEEN SET AND *)    0526
   PROCOPR(TOK)             (* ARITHMETIC USE *)                    0527
ELSE IF (TOK = TDIV) THEN                                            0528
   PROCOPR(TOK)                                                      0529
ELSE IF (TOK = TMOD) THEN                                            0530
   PROCOPR(TOK)                                                      0531
ELSE IF (TOK = SLASH) THEN                                           0532
   PROCOPR(TOK)                                                      0533
ELSE IF (TOK = TIF) THEN                                             0534
   PROCOPR(TIFTHEN)   (* ASSUME IT IS AN 'IF THEN' *)               0535
ELSE IF (TOK = TELSE) THEN BEGIN                                     0536
   PROCOPR(TIFTHENELSE);   (* COUNT AS AN 'IF THEN ELSE' *)         0537
   I := OPRSEARCH(TIFTHEN,ETA1);   (* FIND 'IF THEN' COUNT *)       0538
   NOPR(.I.) := NOPR(.I.) - 1;   (* AND ADJUST BY 1 *)              0539
   END                                                               0540
ELSE IF (TOK = TWHILE) THEN                                          0541
   PROCOPR(TWHILEDO)                                                 0542
ELSE IF (TOK = TFOR) THEN                                            0543
   PROCOPR(TFORDO)                                                   0544
ELSE IF (TOK = TTO) THEN                                             0545
   PROCOPR(TOK)                                                      0546
ELSE IF (TOK = TDOWNTO) THEN                                         0547
   PROCOPR(TOK)                                                      0548
ELSE IF (TOK = TREPEAT) THEN                                         0549
   PROCOPR(TREPEATUNTIL)                                             0550
```

```
ELSE IF (TOK = TCASE) THEN                                              0551
   PROCOPR(TCASEEND)                                                    0552
ELSE IF (TOK = TRECORD) THEN BEGIN                                      0553
   PROCOPR(TRECORDEND);                                                 0554
   FTTYPE := TRUE;    (* FIRST DATA TYPE OF A RECORD APPEARS *)        0555
   END                (* RIGHT AFTER 'RECORD' KEYWORD *)               0556
ELSE IF (TOK = TSET) THEN                                               0557
   PROCOPR(TSETOF)                                                      0558
ELSE IF (TOK = TFILE) THEN                                              0559
   PROCOPR(TFILEOF)                                                     0560
ELSE IF (TOK = TARRAY) THEN                                             0561
   PROCOPR(TARRAYOF)                                                    0562
ELSE IF (TOK = TWITH) THEN                                              0563
   PROCOPR(TWITHDO)                                                     0564
ELSE IF (TOK = TAND) THEN                                               0565
   PROCOPR(TOK)                                                         0566
ELSE IF (TOK = TOR) THEN                                                0567
   PROCOPR(TOK)                                                         0568
ELSE IF (TOK = TNOT) THEN                                               0569
   PROCOPR(TOK)                                                         0570
ELSE IF (TOK = TINTEGER) THEN                                           0571
   PROCOPR(TOK)                                                         0572
ELSE IF (TOK = TREAL) THEN                                              0573
   PROCOPR(TOK)                                                         0574
ELSE IF (TOK = TCHAR) THEN                                              0575
   PROCOPR(TOK)                                                         0576
ELSE IF (TOK = TBOOLEAN) THEN                                           0577
   PROCOPR(TOK)                                                         0578
ELSE IF (TOK = TTEXT) THEN                                              0579
   PROCOPR(TOK)                                                         0580
ELSE IF (TOK = TFUNCTION) THEN BEGIN                                    0581
   PROCOPR(TOK);                                                        0582
   FFUNCTION := YES;                                                    0583
   FLABEL := NO;                                                        0584
   FCONST := NO;                                                        0585
   FTYPE := NO;                                                         0586
   FVARIABLE := NO;    (* DON'T COUNT FUNCTION NAME AS VARIABLE *)     0587
   FVAR := YES;    (* VARIABLES MAY BE DECLARED ON FUNCTION *)         0588
   END    (* DECLARATION LINE *)                                       0589
ELSE IF (TOK = TPROCEDURE) THEN BEGIN                                   0590
   PROCOPR(TOK);                                                        0591
   FPROCEDURE := YES;                                                   0592
   FLABEL := NO;                                                        0593
   FCONST := NO;                                                        0594
   FTYPE := NO;                                                         0595
   FVARIABLE := NO;  (* DON'T COUNT PROCEDURE NAME AS VARIABLE *)      0596
   FVAR := YES;    (* VARIABLES MAY BE DECLARED ON PROCEDURE *)        0597
   END    (* DECLARATION LINE *)                                       0598
ELSE IF (TOK = TOF) OR (TOK = TEND) OR (TOK = TDO) OR                   0599
        (TOK = TUNTIL) OR (TOK = TTHEN) OR                              0600
        (TOK = RARRAY) OR (TOK = RBRACKET) THEN                         0601
   NOP := 0                                                             0602
ELSE IF (TOK = RPAREN) THEN                                             0603
   FVARIABLE := NO    (* OUTSIDE OF FUNCTION OF PROCEDURE DECL. *)     0604
ELSE IF (TOK = TPACKED) THEN                                            0605
```

```
   PROCOPR(TOK)                                                          0606
ELSE IF (TOK = COMMA) THEN                                               0607
   PROCOPR(TOK)                                                          0608
ELSE IF (TOK = PERIOD) THEN    (* NO DISTINCTION BETWEEN RECORD *)       0609
   PROCOPR(TOK)  (* COMPONENT SELECTOR & PROGRAM TERMINATOR *)           0610
ELSE IF (TOK = COLON) THEN BEGIN    (* NO DISTINCTION BETWEEN *)         0611
   PROCOPR(TOK);  (* VARIABLE DEFINITION OPERATOR & SEPARATION *)        0612
                  (* OPERATOR FOR CASE AND GOTO LABELS *)                0613
   FTTYPE := FALSE;    (* FOR RECORD DEFINITIONS, ALL RECORD            0614
                          COMPONENTS ARE ON THE LEFTHAND SIDE            0615
                          OF THE COLON; THIS IS USED TO COUNT            0616
                          RECORD COMPONENTS AS TYPES (NTYPE) *)          0617
   FVARIABLE := FALSE;  (* ALL VARIABLES IN VAR SECTION MUST *)          0618
   END                  (* APPEAR ON RIGHTHAND SIDE OF COLON *)          0619
ELSE IF (TOK = LPAREN) THEN BEGIN                                        0620
   PROCOPR(TPARENS);                                                     0621
   FVARIABLE := YES;    (* VARIABLES DECLARATIONS MAY APPEAR *)          0622
   END    (* AFTER LEFT PARENTHESIS *)                                   0623
ELSE IF (TOK = LARRAY) OR (TOK = LBRACKET) THEN                          0624
   PROCOPR(TBRACKETS)                                                    0625
ELSE IF (TOK = DOTDOT) THEN                                              0626
   PROCOPR(TOK)                                                          0627
ELSE IF (TOK = PTR) THEN                                                 0628
   PROCOPR(TOK)                                                          0629
ELSE IF (TOK = TIN) THEN                                                 0630
   PROCOPR(TOK)                                                          0631
ELSE IF (TOK = TGOTO) THEN                                               0632
   PROCOPR(TOK)                                                          0633
ELSE IF (TOK = TVAR) AND (FPAREN > 0) THEN BEGIN    (* VAR *)           0634
   FVAR := YES;    (* IN PARAMETER LIST COUNT AS OPERATORS *)           0635
   PROCOPR(TOK);                                                         0636
   END                                                                   0637
ELSE IF (TOK = TVAR) AND (FPAREN = 0) THEN BEGIN                         0638
   FLABEL := NO;                                                         0639
   FCONST := NO;                                                         0640
   FTYPE := NO;                                                          0641
   FVAR := YES;                                                          0642
   END                                                                   0643
ELSE IF (TOK = TCONST) THEN BEGIN                                        0644
   FLABEL := NO;                                                         0645
   FCONST := YES;                                                        0646
   FVAR := NO;                                                           0647
   END                                                                   0648
ELSE IF (TOK = TTYPE) THEN BEGIN                                         0649
   FLABEL := NO;                                                         0650
   FCONST := NO;                                                         0651
   FTYPE := YES;                                                         0652
   FVAR := NO;                                                           0653
   END                                                                   0654
ELSE IF (TOK = TLABEL) THEN BEGIN                                        0655
   PROCOPR(TOK);                                                         0656
   FLABEL := YES;                                                        0657
   FVAR := NO;                                                           0658
   END                                                                   0659
ELSE IF (TOK = TFORWARD) THEN                                            0660
```

```
            PROCOPR(TOK)                                                    0661
         ELSE IF (TOK = TEXTERNAL_CW) THEN                                  0662
            PROCOPR(TOK)                                                    0663
         ELSE IF (TOK = TEXTERNAL_PVS) THEN                                 0664
            PROCOPR(TOK)                                                    0665
         ELSE IF (TOK = TEXTERNAL_OS) THEN                                  0666
            PROCOPR(TOK)                                                    0667
         ELSE IF (TOK = TPROGRAM) THEN BEGIN                                0668
            PROCOPR(TOK);                                                   0669
            FPROGRAM := YES;                                                0670
            END                                                             0671
         (* WRITELN, READLN, EOF, AND EOLN DO NOT HAVE TO HAVE             0672
            FOLLOWING PARENTHESIS (SPECIAL CASE): COMMON USAGE              0673
            IS "WRITELN;", "READLN;", EOF, AND EOLN; OTHER                  0674
            PROCEDURES AND FUNCTIONS MAY ALSO NOT REQUIRE A                 0675
            PARAMETER LIST, BUT IT IS ASSUMED THAT THEY WILL *)             0676
         ELSE IF (TOK = TWRITELN) OR (TOK = TREADLN) OR                     0677
            (TOK = TEOF) OR (TOK = TEOLN) THEN BEGIN                        0678
            PROCOPR(TOK);                                                   0679
            PROCSUBPGM(TOK);                                                0680
            END                                                             0681
         ELSE IF (NOT FLABEL) THEN BEGIN                                    0682
            (* ALL LABEL DECLARATIONS SHOULD BE IGNORED; ALL               0683
               OTHER TOKENS ARE OPERANDS IF A LEFT PARENTHESIS              0684
               DOES NOT IMMEDIATELY FOLLOW THE TOKEN *)                     0685
            SAVEPOS := POS;    (* SAVE BUFFER POINTER *)                   0686
            LTOKEN(POS,LINE,TEMPTOK);    (* GET NEXT TOKEN *)              0687
            POS := SAVEPOS;    (* RESTORE BUFFER POINTER *)                 0688
            IF (TEMPTOK <> LPAREN) THEN BEGIN    (* OPERAND ? *)           0689
               PROCOPD(TOK);                                                0690
               END                                                          0691
            ELSE BEGIN                                                      0692
               (* IT MUST BE AN OPERATOR: SUBPROGRAM NAME OR CALL *)        0693
               (* DO NOT COUNT NAME ON FUNCTION OR PROCEDURE                0694
                  DECLARATION AS AN OPERATOR *)                             0695
               IF (NOT FFUNCTION) AND (NOT FPROCEDURE) THEN                 0696
                  PROCOPR(TOK);                                             0697
                                                                            0698
               (* ADD TOK TO SUBPROGRAM NAME LIST *)                        0699
               PROCSUBPGM(TOK);                                             0700
            END;                                                            0701
            FFUNCTION := NO;                                                0702
            FPROCEDURE := NO;                                               0703
            FPROGRAM := NO;                                                 0704
            IF (FCONST) AND (FCONSTANT) THEN    (* IF IN CONST SECTION *)  0705
               NCONST := NCONST + 1    (* AND A CONSTANT, THEN COUNT IT *)  0706
            ELSE IF (FTYPE) AND (FTTYPE) THEN    (* IF IN TYPE SECTION *)  0707
               NTYPE := NTYPE + 1    (* AND A TYPE, THEN COUNT IT *)         0708
            ELSE IF (FVAR) AND (FVARIABLE) THEN    (* IF IN VAR *)         0709
               NVAR := NVAR + 1;    (* SECTION AND A VARIABLE, *)           0710
         END                        (* THEN COUNT IT *)                     0711
      END;                                                                  0712
                                                                            0713
                                                                            0714
(*==================================================================*)     0715
```

```
(*                                                                    *)  0716
(*   MCCABE'S CYCLOMATIC NUMBER:                                      *)  0717
(*                                                                    *)  0718
(*   THIS METRIC IS COMPUTED USING THE NUMBER OF DECISION POINTS      *)  0719
(*   PLUS ONE METHOD.  THE ALGORITHM IS OUTLINED IN CONTE.            *)  0720
(*                                                                    *)  0721
(*====================================================================*)  0722
                                                                          0723
PROCEDURE MCCABE (TOK : STRING(MAXTOKLEN));                               0724
   BEGIN                                                                  0725
      IF (TOK = TWHILE) OR (TOK = TIF) OR (TOK = TUNTIL) THEN             0726
         BEGIN                                                            0727
            FCOND := YES;    (* START OF CONDITIONAL *)                  0728
            VG := VG + 1;                                                 0729
         END                                                              0730
      ELSE IF (TOK = TFOR) OR (TOK = TPROCEDURE) OR                       0731
              (TOK = TFUNCTION) OR (TOK = TPROGRAM) THEN                  0732
         BEGIN                                                            0733
            VG := VG + 1;                                                 0734
         END                                                              0735
      ELSE IF (FCOND) AND ((TOK = TAND) OR (TOK = TOR)) THEN              0736
         BEGIN                                                            0737
            VG := VG + 1;                                                 0738
         END                                                              0739
      ELSE IF (TOK = TCASE) THEN    (* KEEP TRACK OF CASE *)             0740
         BEGIN                                                            0741
            FCASE := FCASE + 1;                                           0742
         END                                                              0743
      ELSE IF (FCASE >= 1) AND (FPAREN = 0) AND ((TOK = COLON) OR         0744
              (TOK = COMMA)) THEN                                         0745
         BEGIN                                                            0746
            VG := VG + 1;                                                 0747
         END                                                              0748
      ELSE IF (TOK = TBEGIN) AND (FCASE >= 1) THEN                        0749
         BEGIN                                                            0750
            FCASE := FCASE + 1;                                           0751
         END                                                              0752
      ELSE IF (TOK = TEND) AND (FCASE >= 1) THEN                          0753
         BEGIN                                                            0754
            FCASE := FCASE - 1;                                           0755
         END                                                              0756
      ELSE IF (FCOND) AND ((TOK = TDO) OR (TOK = TTHEN) OR                0757
              (TOK = SEMICOLON)) THEN                                     0758
         BEGIN                                                            0759
            FCOND := NO;    (* END OF CONDITIONAL *)                     0760
         END                                                              0761
      ELSE IF (TOK = TLABEL) THEN                                         0762
         BEGIN                                                            0763
            FLABEL := YES;                                                0764
            VG := VG - 1;                                                 0765
         END                                                              0766
      ELSE IF (TOK = COMMA) AND (FLABEL) THEN                             0767
         BEGIN                                                            0768
            VG := VG - 1;                                                 0769
         END                                                              0770
```

```
      ELSE IF (TOK = SEMICOLON) AND (FLABEL) THEN                          0771
         BEGIN                                                             0772
            FLABEL := NO;                                                  0773
         END;                                                              0774
   END;                                                                    0775
                                                                           0776
                                                                           0777
(*=================================================================*)     0778
(*                                                                 *)     0779
(*   HALSTEAD:                                                     *)     0780
(*                                                                 *)     0781
(*   THIS PROCEDURE COMPUTES HALSTEAD'S SOFTWARE METRICS.          *)     0782
(*   THE METRICS ARE ...                                           *)     0783
(*                                                                 *)     0784
(*        N1 ..... TOTAL NUMBER OF OPERATORS                       *)     0785
(*        N2 ..... TOTAL NUMBER OF OPERANDS                        *)     0786
(*        PLEN ..... PROGRAM LENGTH                                *)     0787
(*        EPLEN .... ESTIMATED PROGRAM LENGTH                      *)     0788
(*        PVOL ..... PROGRAM VOLUME                                *)     0789
(*        EPLEV .... ESTIMATED PROGRAM LEVEL                       *)     0790
(*        EPDIF .... ESTIMATED PROGRAM DIFFICULTY                  *)     0791
(*        EPPVOL ... ESTIMATED POTENTIAL PROGRAM VOLUME            *)     0792
(*        EFFORT ... NUMBER OF ELEMENTARY MENTAL DISCRIMINATIONS   *)     0793
(*        TIME ..... IMPLEMENTATION TIME IN SECONDS                *)     0794
(*        LAMBDA ... LANGUAGE LEVEL                                *)     0795
(*                                                                 *)     0796
(*=================================================================*)     0797
                                                                           0798
PROCEDURE HALSTEAD (ETA1, ETA2 : NONNEG; VAR N1, N2, PLEN : NONNEG;        0799
                    VAR EPLEN, PVOL, EPLEV, EPDIF, EPPVOL, EFFORT,         0800
                        TIME, LAMBDA : REAL);                              0801
   CONST                                                                   0802
      BETA = 18;    (* STROUD NUMBER: NORMALLY SET TO 18 SINCE            0803
                       THIS GAVE THE BEST RESULTS IN HALSTEAD'S            0804
                       EARLIEST EXPERIMENTS. *)                            0805
   VAR                                                                     0806
      I : INTEGER;                                                         0807
   BEGIN                                                                   0808
      FOR I := 1 TO ETA1 DO BEGIN                                          0809
         N1 := N1 + NOPR(.I.);                                             0810
      END;                                                                 0811
      FOR I := 1 TO ETA2 DO BEGIN                                          0812
         N2 := N2 + NOPD(.I.);                                             0813
      END;                                                                 0814
      PLEN := N1 + N2;                                                     0815
      IF (ETA1 > 0) AND (ETA2 > 0) THEN                                    0816
         EPLEN := ETA1 * LN(ETA1) / LN(2) + ETA2 * LN(ETA2) / LN(2);       0817
      IF ((ETA1+ETA2) > 0) THEN                                            0818
         PVOL := PLEN * LN(ETA1+ETA2) / LN(2);                             0819
      IF (ETA1 <> 0) AND (N2 <> 0) THEN BEGIN                              0820
         EPLEV := (2 / ETA1) * (ETA2 / N2);                                0821
         EPDIF := 1 / EPLEV;                                               0822
         EPPVOL := PVOL * EPLEV;                                           0823
      END;                                                                 0824
      IF (ETA2 > 0) THEN                                                   0825
```

```
      EFFORT := ETA1 * N2 * PLEN * (LN(ETA1+ETA2) / LN(2)) / (2 * ETA2);0826
      TIME := EFFORT / BETA;                                            0827
      LAMBDA := SQR(EPLEV) * PVOL;   (* USING EST. PROG. LEVEL *)       0828
   END;                                                                 0829
                                                                        0830
                                                                        0831
(*====================================================================*) 0832
(*                                                                    *) 0833
(*   INITIALIZE:                                                      *) 0834
(*                                                                    *) 0835
(*   THIS PROCEDURE INITIALIZES ALL ARRAYS, FLAGS, AND COUNTERS       *) 0836
(*   USED IN THE PROGRAM.                                             *) 0837
(*                                                                    *) 0838
(*====================================================================*) 0839
                                                                        0840
PROCEDURE INITIALIZE;                                                   0841
   VAR                                                                  0842
      I : INTEGER;                                                      0843
   BEGIN                                                                0844
   (* INITIALIZATION *)                                                 0845
      IF DEBUG THEN WRITELN('START OF INITIALIZATION');                 0846
      (* MCCABE'S METRIC *)                                            0847
      VG := 0;    (* # OF DECISION POINTS PLUS ONE *)                  0848
                  (* THE 'PLUS ONE' IS ACCOUNTED FOR BY COUNTING       0849
                     THE 'PROGRAM' KEYWORD *)                           0850
      (* HALSTEAD'S BASIC METRICS *)                                   0851
      ETA1 := 0;    N1 := 0;    ETA2 := 0;    N2 := 0;                0852
      (* INITIALIZE OPERAND ARRAYS *)                                  0853
      FOR I := 1 TO MAXOPD DO BEGIN                                     0854
         OPD(.I.) := BLANK;                                             0855
         NOPD(.I.) := 0;                                                0856
      END;                                                              0857
      (* INITIALIZE OPERATOR ARRAYS *)                                 0858
      FOR I := 1 TO MAXOPR DO BEGIN                                     0859
         OPR(.I.) := BLANK;                                             0860
         NOPR(.I.) := 0;                                                0861
      END;                                                              0862
      (* INITIALIZE USER-DEFINED SUBPROGRAMS ARRAY *)                  0863
      FOR I := 1 TO MAXSUB DO                                           0864
         SUBPGM(.I.) := BLANK;                                          0865
      NSUB := 0;                                                        0866
                                                                        0867
      (* LOC METRIC *)                                                 0868
      LOC := 0;                                                         0869
      (* FLAGS *)                                                      0870
      FCOND := FALSE;                                                   0871
      FLABEL := FALSE;                                                  0872
      FSQUOTE := FALSE;                                                 0873
      FDQUOTE := FALSE;                                                 0874
      FCOM := FALSE;                                                    0875
      FPAREN := 0;                                                      0876
      FCASE := 0;                                                       0877
      FPROCEDURE := FALSE;                                              0878
      FFUNCTION := FALSE;                                               0879
      FVARIABLE := TRUE;                                                0880
```

```
      FCONSTANT := FALSE;                                                    0881
      FTTYPE := FALSE;                                                       0882
                                                                             0883
      IF DEBUG THEN WRITELN('END OF INITIALIZATION');                        0884
   END;                                                                      0885
                                                                             0886
                                                                             0887
BEGIN                                                                        0888
   INITIALIZE;   (* INITIALIZE PROGRAM VARIABLES *)                         0889
                                                                             0890
   (* SKIP TO START OF PROGRAM ($JOB) *)                                    0891
   POS := MAXLINELEN + 1;                                                    0892
   TOK := BLANK;                                                             0893
   WHILE (NOT EOF) AND (TOK <> T$JOB) DO BEGIN                               0894
      IF (POS = (MAXLINELEN+1)) THEN (* IF CURRENT LINE *)                   0895
         BEGIN                       (* PARSED, THEN    *)                  0896
            READLN(LINE);            (* READ NEXT LINE  *)                  0897
            IF DEBUG THEN                                                    0898
               WRITELN('ECHO: ',LINE);                                       0899
            POS := 0;                                                        0900
         END;                                                                0901
      LTOKEN(POS,LINE,TOK);   (* PARSE NEXT TOKEN FROM LINE *)               0902
      TOUPPER(TOK);                                                          0903
      (* IF USER ID, THEN WRITE TO OUTPUT FILE *)                            0904
      IF (TOK(.1.) = 'U') THEN WRITELN(OUTFILE,'UID=',TOK);                  0905
      IF DEBUG THEN                                                          0906
         WRITELN(POS:5,'*',TOK,'*');                                         0907
   END;                                                                      0908
                                                                             0909
   (* PROCESS ACTUAL PASCAL PROGRAM *)                                      0910
   WHILE (NOT EOF) AND (TOK <> T$ENTRY) DO BEGIN                             0911
      READLN(LINE);   (* READ NEXT PHYSICAL PROGRAM LINE *)                 0912
      IF DEBUG THEN WRITELN('ECHO: ',LINE);                                  0913
      IF DEBUG THEN WRITELN('      VG = ',VG:1);                             0914
      POS := 0;   (* RESET POSITION TO START AT BEGINNING OF NEW LINE *)    0915
      INC := 0;   (* ASSUME NEXT LINE IS A BLANK OR COMMENT LINE *)          0916
      WHILE (POS <= MAXLINELEN) DO BEGIN (* PARSE ENTIRE LINE *)             0917
         LTOKEN(POS,LINE,TOK);   (* PARSE NEXT TOKEN FROM LINE *)           0918
         TOUPPER(TOK);           (* INSURE TOKENS ARE IN UPPERCASE *)       0919
         IF (TOK = BCOMI) OR (TOK = BCOMII) THEN   (* BEGINNING OF A *)     0920
            BEGIN                                  (* COMMENT? *)           0921
               FCOM := YES;                                                  0922
            END                                                              0923
         ELSE IF (TOK = SQUOTE) AND NOT FCOM THEN  (* BEGINNING OF A *)     0924
            BEGIN              (* STRING DELIMITED BY SINGLE QUOTES? *)     0925
               FSQUOTE := TRUE;                                              0926
               IF (NOT FCOM) THEN BEGIN   (* NOT WITHIN A COMMENT? *)       0927
                  TEMPTOK := TOK;                                            0928
                  REPEAT                  (* THEN BUILD STRING *)           0929
                     PREVTOK := TEMPTOK;                                     0930
                     TOK(.1.) := LINE(.POS.);                                0931
                     POS := POS + 1;                                         0932
                     IF (LENGTH(PREVTOK)+LENGTH(TOK)) <= MAXTOKLEN THEN      0933
                        WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),            0934
                                         TOK:LENGTH(TOK));                   0935
```

```
      UNTIL (TOK = SQUOTE);                                              0936
      TOK := TEMPTOK;                                                    0937
   END;                                                                  0938
END                                                                      0939
ELSE IF (TOK = DQUOTE) AND NOT FCOM THEN  (* BEGINNING OF A *)          0940
   BEGIN              (* STRING DELIMITED BY DOUBLE QUOTES? *)          0941
      FDQUOTE := TRUE;                                                   0942
      IF (NOT FCOM) THEN BEGIN   (* NOT WITHIN A COMMENT? *)            0943
         TEMPTOK := TOK;                                                 0944
         REPEAT                     (* THEN BUILD STRING *)             0945
            PREVTOK := TEMPTOK;                                          0946
            TOK(.1.) := LINE(.POS.);                                     0947
            POS := POS + 1;                                              0948
            IF (LENGTH(PREVTOK)+LENGTH(TOK)) <= MAXTOKLEN THEN           0949
               WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),                 0950
                                TOK:LENGTH(TOK));                        0951
         UNTIL (TOK = DQUOTE);                                           0952
         TOK := TEMPTOK;                                                 0953
      END;                                                               0954
   END                                                                   0955
ELSE IF (TOK = LPAREN) AND NOT FCOM THEN (* KEEP TRACK OF *)             0956
   BEGIN                                  (* PARENTHESES *)             0957
      FPAREN := FPAREN + 1;    (* MATCHING LEFT AND *)                   0958
   END                                                                   0959
ELSE IF (TOK = RPAREN) AND NOT FCOM THEN                                 0960
   BEGIN                                                                 0961
      FPAREN := FPAREN - 1;    (* RIGHT PARENTHESIS *)                   0962
   END                                                                   0963
ELSE IF ISNUMBER(TOK) AND NOT FCOM THEN                                  0964
   BEGIN  (* CHECK FOR NUMBERS *)                                       0965
   FREAL := FALSE;    (* SET REAL NUMBER FLAG *)                        0966
   SAVEPOS := POS;    (* SAVE POSITION FOR BACKTRACKING *)              0967
   PREVTOK := TOK;                                                       0968
   LTOKEN(POS,LINE,TOK);    (* GET NEXT TOKEN AND... *)                 0969
   TOUPPER(TOK);                                                         0970
   IF (TOK = DECIMALPT) THEN BEGIN   (* DECIMAL PT.? *)                  0971
      WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),                          0972
                       TOK:LENGTH(TOK));                                 0973
      PREVTOK := TEMPTOK;    (* CONCATE DECIMAL POINT *)                 0974
      FREAL := TRUE;    (* IT IS A REAL NUMBER *)                       0975
      LTOKEN(POS,LINE,TOK);                                              0976
      TOUPPER(TOK);                                                      0977
      IF ISNUMBER(TOK) THEN BEGIN                                        0978
         WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),                       0979
                          TOK:LENGTH(TOK));                              0980
         PREVTOK := TEMPTOK;    (* CONCATE DECIMAL DIGITS *)             0981
         LTOKEN(POS,LINE,TOK);                                           0982
         TOUPPER(TOK);                                                   0983
      END;                                                               0984
   END;                                                                  0985
   (* TOK(.1.) IS REQUIRED BECAUSE IF THERE IS NOT A *)                  0986
   (* PLUS OR MINUS SIGN BETWEEN 'E' AND THE EXPONENT, *)                0987
   (* THEN LTOKEN WILL RETURN BOTH AS ONE TOKEN *)                       0988
   IF (TOK(.1.) = 'E') THEN BEGIN                                        0989
      WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),                          0990
```

```
                              TOK:LENGTH(TOK));                              0991
               PREVTOK := TEMPTOK;    (* CONCATE E DESIGNATOR *)            0992
               FREAL := TRUE;    (* IT IS A REAL NUMBER *)                   0993
               (* ONLY GET NEXT TOKEN IF EXPONENT NOT PART OF *)             0994
               (* CURRENT TOKEN *)                                           0995
               IF LENGTH(TOK) < 2 THEN LTOKEN(POS,LINE,TOK);                 0996
               TOUPPER(TOK);                                                 0997
               IF (TOK = PLUS) OR (TOK = MINUS) THEN BEGIN                   0998
                  WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),                  0999
                                   TOK:LENGTH(TOK));                         1000
                  PREVTOK := TEMPTOK;    (* CONCATE EXPONENT SIGN *)         1001
                  LTOKEN(POS,LINE,TOK);                                      1002
                  TOUPPER(TOK);                                              1003
               END;                                                          1004
               IF ISNUMBER(TOK) THEN BEGIN                                   1005
                  WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),                  1006
                                   TOK:LENGTH(TOK));                         1007
                  PREVTOK := TEMPTOK;    (* CONCATE EXPONENT DIGITS *)      1008
               END;                                                          1009
            END;                                                             1010
            (* IF NOT A REAL NUMBER, THEN BACKTRACK *)                       1011
            IF NOT(FREAL) THEN POS := SAVEPOS;                               1012
            TOK := PREVTOK;                                                  1013
         END;                                                                1014
         IF DEBUG THEN WRITELN(POS:5,'*',TOK,'*');                           1015
                                                                             1016
         (* LOC METRIC *)                                                    1017
         IF (TOK <> BLANK) AND (NOT FCOM) THEN  (* COUNT NON-BLANK *)        1018
            INC := 1;                          (* AND NON-COMMENT LINES *)   1019
                                                                             1020
         (* MCCABE'S METRIC: IGNORE COMMENTS, QUOTED STRINGS *)              1021
         (* (BOTH SINGLE AND DOUBLE), AND BLANK LINES *)                     1022
         IF (NOT FCOM) AND (NOT FSQUOTE) AND (NOT FDQUOTE)                   1023
            AND (TOK <> BLANK) THEN MCCABE(TOK);                             1024
                                                                             1025
         (* HALSTEAD'S OPERATOR/OPERAND TOKEN COUNT *)                       1026
         (* IGNORE COMMENTS, THE $ENTRY LINE, AND BLANK LINES *)             1027
         IF (NOT FCOM) AND (TOK <> T$ENTRY) AND (TOK <> BLANK) THEN          1028
            TOKCOUNT(TOK);                                                   1029
                                                                             1030
         (* THE FOLLOWING LINES MUST FOLLOW THE PROCEDURE CALLS              1031
            TO MCCABE AND TOKCOUNT *)                                        1032
         (* RESET FSQUOTE *)                                                1033
         FSQUOTE := FALSE;                                                   1034
         (* RESET FDQUOTE *)                                                1035
         FDQUOTE := FALSE;                                                   1036
                                                                             1037
         IF (TOK = ECOMI) OR (TOK = ECOMII) THEN   (* CHECK FOR *)          1038
            FCOM := NO;                                (* COMMENT END *)     1039
      END;                                                                   1040
      LOC := LOC + INC;           (* INCREMENT LINE COUNT ACCORDINGLY *)     1041
      IF DEBUG THEN WRITELN;                                                 1042
END;                                                                         1043
                                                                             1044
(* OUTPUT METRICS *)                                                        1045
```

```
   IF (LOC > 0) THEN                                                    1046
      LOC := LOC - 1;            (* DON'T COUNT $ENTRY LINE *)         1047
   WRITELN('THERE ARE ',LOC:1,' NON-BLANK, NON-COMMENT LINES.');        1048
   WRITELN(OUTFILE,'LOC=',LOC:1);                                       1049
   WRITELN('MCCABE''S: VG = ',VG:1);                                    1050
   WRITELN(OUTFILE,'VG=',VG:1);                                         1051
   WRITELN('NVAR=',NVAR:1,' NCONST=',NCONST:1,' NTYPE=',NTYPE:1,        1052
           ' NSUB=',NSUB:1);                                            1053
   WRITELN(OUTFILE,                                                     1054
           'NVAR=',NVAR:1,' NCONST=',NCONST:1,' NTYPE=',NTYPE:1,        1055
           ' NSUB=',NSUB:1);                                            1056
   HALSTEAD(ETA1,ETA2,N1,N2,PLEN,EPLEN,PVOL,EPLEV,EPDIF,EPPVOL,         1057
            EFFORT,TIME,LAMBDA);                                        1058
   IF ((ETA1+1) > MAXOPR) OR ((ETA2+1) > MAXOPD) OR                     1059
      ((NSUB+1) > MAXSUB) THEN BEGIN                                    1060
      WRITELN('WARNING: FOLLOWING METRICS WILL NOT BE ACCURATE');       1061
      WRITELN('         BECAUSE NOT ALL TOKENS COULD BE COUNTED');      1062
      WRITELN(OUTFILE,'WARNING: FOLLOWING METRICS WILL NOT BE',         1063
                      ' ACCURATE');                                     1064
      WRITELN(OUTFILE,'         BECAUSE NOT ALL TOKENS COULD',          1065
                      ' BE COUNTED');                                   1066
   END;                                                                 1067
   WRITELN('HALSTEAD''S:  ETA1 = ',ETA1:1,'  ETA2 = ',ETA2:1);          1068
   WRITELN('OPERATORS:');                                               1069
   FOR I := 1 TO ETA1 DO BEGIN                                          1070
      WRITELN(OPR(.I.),' ',NOPR(.I.):3);                                1071
   END;                                                                 1072
   WRITELN;                                                             1073
   WRITELN('TOTAL # OF OPERATORS = ',N1:3);                             1074
   WRITELN;                                                             1075
   WRITELN('OPERANDS:');                                                1076
   FOR I := 1 TO ETA2 DO BEGIN                                          1077
      WRITELN(OPD(.I.),' ',NOPD(.I.):3);                                1078
   END;                                                                 1079
   WRITELN;                                                             1080
   WRITELN('TOTAL # OF OPERANDS = ',N2:3);                              1081
   WRITELN;                                                             1082
   WRITELN('PLEN=',PLEN:1,' EPLEN=',EPLEN:4:2);                         1083
   WRITELN('PVOL=',PVOL:4:2,' EPPVOL=',EPPVOL:4:2);                     1084
   WRITELN('EPLEV=',EPLEV:6:4,' EPDIF=',EPDIF:5:3);                     1085
   WRITELN('EFFORT=',EFFORT:2:0,' TIME=',TIME:4:2,' (SEC)  ',           1086
           (TIME/60):4:2,' (MINUTES)');                                 1087
   WRITELN('LAMBDA=',LAMBDA:4:2);                                       1088
   WRITELN(OUTFILE,'ETA1=',ETA1:1,' N1=',N1:1,' ETA2=',ETA2:1,          1089
           ' N2=',N2:1);                                                1090
   WRITELN(OUTFILE,'PLEN=',PLEN:1,' EPLEN=',EPLEN:4:2,                  1091
           ' PVOL=',PVOL:4:2,' EPPVOL=',EPPVOL:4:2);                    1092
   WRITELN(OUTFILE,'EPLEV=',                                            1093
           EPLEV:6:4,' EPDIF=',EPDIF:5:3);                              1094
   WRITELN(OUTFILE,                                                     1095
           'EFFORT=',EFFORT:2:0,                                        1096
           ' TIME=',TIME:4:2,' LAMBDA=',LAMBDA:4:2);                    1097
END.                                                                    1098
```

APPENDIX D

SAMPLE WPASCAL PROGRAM USED TO TEST VS PASCAL METRICS PROGRAM; INCLUDES SAMPLE INPUT, OUTPUT, DETAILED METRICS, AND SUMMARY METRICS

```
1  | PROGRAM SAMPLE (INPUT,OUTPUT);
2  |
3  | (*  THIS IS A SAMPLE PASCAL PROGRAM.     *)
4  | (*  IT IS USED TO ILLUSTRATE HOW THE     *)
5  | (*  FOLLOWING METRICS ARE COMPUTED:      *)
6  | (*     - NUMBER OF LINES OF CODE (LOC)   *)
7  | (*     - MCCABE'S CYCLOMATIC NUMBER      *)
8  | (*     - HALSTEAD'S SOFTWARE SCIENCE     *)
9  |
10 | CONST
11 |    MAXELE = 20;  (* MAXIMUM ELEMENTS *)
12 |
13 | TYPE
14 |    MEASUREMENTS = ARRAY (.1..MAXELE.) OF REAL;
15 |
16 | VAR
17 |    A : MEASUREMENTS;
18 |    SUM,       (* SUM OF MEASUREMENTS *)
19 |    MIN,       (* MINIMUM MEASUREMENT *)
20 |    MAX        (* MAXIMUM MEASUREMENT *)
21 |       : REAL;
22 |    CNT,       (* NUMBER OF MEASUREMENTS *)
23 |    I
24 |       : INTEGER;
25 |
26 | PROCEDURE MAXMIN (VAR M : MEASUREMENTS; VAR MAX, MIN : REAL;
27 |                   N : INTEGER);
28 |    VAR
29 |       I : INTEGER;
30 |    BEGIN
31 |       (* FIND MINIMUM AND MAXIMUM *)
32 |       MIN := M(.1.);
33 |       MAX := M(.1.);
34 |       FOR I := 2 TO N DO BEGIN
35 |          IF M(.I.) < MIN THEN MIN := M(.I.);
36 |          IF M(.I.) > MAX THEN MAX := M(.I.);
37 |       END;
38 |    END;
39 |
40 | FUNCTION MEAN (SUM : REAL; N : INTEGER) : REAL;   (* COMPUTE MEAN *)
41 |    BEGIN
42 |       MEAN := (SUM/N);
43 |    END;
44 |
45 | BEGIN
46 |    (* INITIALIZATION *)
47 |    SUM := +0.0E0;
48 |    CNT := 0;
49 |    A(.1.) := 0.0;   (* IN CASE NO INPUT DATA *)
50 |
51 |    (* READ, COUNT, AND SUM MEASUREMENTS *)
52 |    WHILE (CNT < MAXELE) AND NOT EOF DO BEGIN
53 |       CNT := CNT + 1;
54 |       READLN(A(.CNT.));     (* READ MEASUREMENT *)
55 |       WRITELN(A(.CNT.));    (* ECHO PRINT *)
```

```
56  |          SUM := SUM + A(.CNT.);    (* ACCUMULATE MEASUREMENTS *)
57  |      END;
58  |
59  |      MAXMIN(A, MAX, MIN, CNT);    (* CALL PROCEDURE TO FIND
60  |                                       MAXIMUM AND MINIMUM MEASUREMENTS *)
61  |
62  |      (* REPORT COUNT, SUM, AVERAGE, MINIMUM, AND MAXIMUM MEASUREMENTS *)
63  |      WRITELN;
64  |      WRITELN("COUNT = ",CNT:1);
65  |      WRITELN('SUM = ',SUM:6:4);
66  |      WRITELN('MEAN = ',MEAN(SUM,CNT):6:4);
67  |      WRITELN('MINIMUM = ',MIN:6:4);
68  |      WRITELN('MAXIMUM = ',MAX:6:4);
69  |  END.
```

Sample Input:

```
  6.43
  0
 -2.34
  8.75
 -5.25
  9
  3
 -1.75
```

Sample Output:

```
  6.430000000E+00
  0.000000000E+00
 -2.340000000E+00
  8.750000000E+00
 -5.250000000E+00
  9.000000000E+00
  3.000000000E+00
 -1.750000000E+00

 COUNT = 8
 SUM = 17.8400
 MEAN = 2.2300
 MINIMUM = -5.2500
 MAXIMUM = 9.0000
```

```
DETAILED METRICS:


THERE ARE 48 NON-BLANK, NON-COMMENT LINES.
MCCABE'S: VG = 8
NVAR=13 NCONST=1 NTYPE=1 NSUB=6
HALSTEAD'S:  ETA1 = 33  ETA2 = 26


OPERATORS:

PROGRAM    1
SAMPLE    1
( )   14
,   14
;   34
=    2
ARRAY OF    1
(. .)   11
..    1
REAL    5
:   19
INTEGER    4
PROCEDURE    1
VAR    2
BEGIN END    5
:=   11
FOR DO    1
TO    1
IF THEN    2
<    2
>    1
FUNCTION    1
/    1
+    3
WHILE DO    1
AND    1
NOT    1
EOF    1
READLN    1
WRITELN    7
MAXMIN    1
MEAN    1
.    1

TOTAL # OF OPERATORS = 153


OPERANDS:

INPUT    1
OUTPUT    1
MAXELE    3
2O    1
MEASUREMENTS    3
```

```
1    6
A    6
SUM    8
MIN    7
MAX    7
CNT   11
I    7
M    7
N    4
2    1
MEAN    1
0.0E0    1
0    1
0.0    1
"COUNT = "    1
'SUM = '    1
6    4
4    4
'MEAN = '    1
'MINIMUM = '    1
'MAXIMUM = '    1

TOTAL # OF OPERANDS =  90


PLEN=243 EPLEN=288.68
PVOL=1429.48 EPPVOL=25.03
EPLEV=0.0175 EPDIF=57.115
EFFORT=81645. TIME=4535.86 (SEC)  75.60 (MINUTES)
LAMBDA=0.44
```

SUMMARY METRICS:

UID=SAMPLE
LOC=48
VG=8
NVAR=13 NCONST=1 NTYPE=1 NSUB=6
ETA1=33 N1=153 ETA2=26 N2=90
PLEN=243 EPLEN=288.68 PVOL=1429.48 EPPVOL=25.03
EPLEV=0.0175 EPDIF=57.115
EFFORT=81645. TIME=4535.86 LAMBDA=0.44

APPENDIX E

VS PASCAL PROGRAM USED TO EXTRACT ACCOUNTING DATA ON THE IBM

```
PROGRAM EXTRACTDATA (INPUT,OUTPUT,OUTFILE);                                001
(* THIS IS THE PRODUCTION CODE WITHOUT THE DEBUG PRINT SWITCH *)           002
CONST                                                                      003
   BLANK = ' ';                                                            004
                                                                           005
   LINELEN = 80;     (* LENGTH OF PHYSICAL LINE READ *)                    006
   MAXTOKLEN = 20;   (* MAXIMUM LENGTH OF TOKEN *)                         007
   UIDSIZE = 7;      (* LENGTH OF USER ID STRING *)                        008
   MAXTRANS = 100;   (* MAXIMUM NUMBER OF TRANSACTIONS TO READ *)          009
                                                                           010
   (* TOKEN FIELD NAMES IN THE RAW DATA *)                                 011
   TTRANCODE = 'TRANCODE';                                                 012
   TTRANDATE = 'TRANDATE';                                                 013
   TTRANPROJ = 'TRANPROJ';                                                 014
   TTRANDSCR = 'TRANDSCR';                                                 015
   TCHRGFLAG = 'CHRGFLAG';                                                 016
   TTRANAMT = 'TRANAMT';                                                   017
   TKEYDATE = 'KEYDATE';                                                   018
   TKEYTIME = 'KEYTIME';                                                   019
   TSECONDID = 'SECONDID';                                                 020
   TSTRTTIME = 'STRTTIME';                                                 021
   TSTOPTIME = 'STOPTIME';                                                 022
   TTYPETASK = 'TYPETASK';                                                 023
   TJESNR = 'JESNR';                                                       024
   TJOBCLASS = 'JOBCLASS';                                                 025
   TPGMRNAME = 'PGMRNAME';                                                 026
   TRACFGRUP = 'RACFGRUP';                                                 027
   TRACFUSER = 'RACFUSER';                                                 028
   TCONDCODE = 'CONDCODE';                                                 029
   TSTEPNR = 'STEPNR';                                                     030
   TSHIFT = 'SHIFT';                                                       031
   TCPUTIME = 'CPUTIME';                                                   032
   TCPUCOST = 'CPUCOST';                                                   033
   TCNCTTIME = 'CNCTTIME';                                                 034
   TCNCTCOST = 'CNCTCOST';                                                 035
   TDISKEXCP = 'DISKEXCP';                                                 036
   TDISKCOST = 'DISKCOST';                                                 037
   TTAPEEXCP = 'TAPEEXCP';                                                 038
   TTAPECOST = 'TAPECOST';                                                 039
   TTRANSCHG = 'TRANSCHG';                                                 040
   TBATCH = 'BATCH';                                                       041
   TJOB = 'JOB';                                                           042
                                                                           043
   (* DATA CONSTANTS *)                                                    044
   JOB = 100;    (* BATCH PROCESSING *)                                   045
   WYLBURCMD = 'WYL-BR-CMDS';    (* INDICATES WYLBUR COMMANDS *)           046
   EOR = '*';    (* END-OF-RECORD MARKER: ACTUALLY '***' BUT              047
                    LTOKEN WILL NOT RECOGNIZE IT *)                        048
                                                                           049
   (* DATE RANGE OF DESIRED DATA VALUES *)                                 050
   MINDATE = 890303;   MAXDATE = 890324;    (* YYMMDD FORMAT *)           051
                                                                           052
VAR                                                                        053
   POS : INTEGER;    (* POSITION IN LINE WHERE NEXT TOKEN APPEARS *)       054
   LINE : STRING(LINELEN);     (* PHYSICAL LINE *)                        055
```

```
   TOK,  (* CURRENT TOKEN *)                                                  056
   PREVTOK,   (* PREVIOUS TOKEN *)                                           057
   TEMPTOK   (* TEMPORARY TOKEN *)                                           058
      : STRING(MAXTOKLEN);                                                   059
   RACFUSER : STRING(UIDSIZE);   (* RACF USER ID *)                          060
   TRANDATE : INTEGER;   (* DATE OF TRANSACTION *)                          061
   TRANCODE,   (* TRANSACTION CODE *)                                        062
   CONDCODE    (* CONDITION CODE *)                                          063
      : INTEGER;                                                             064
   CPUTIME,   (* CPU TIME OF JOB *)                                         065
   CNCTTIME   (* CONNECT TIME FOR JOB *)                                     066
      : REAL;                                                                067
   OUTFILE    (* OUTPUT FILE OF SELECTED DATA *)                            068
      : TEXT;                                                                069
                                                                             070
(*************************************************************************)  071
(*                                                                       *)  072
(*   THIS PROCEDURE SKIPS DATA UNTIL THE SEARCH TOKEN IS FOUND.          *)  073
(*                                                                       *)  074
(*************************************************************************)  075
PROCEDURE SKIPTO (STOK : STRING(MAXTOKLEN));                                 076
   BEGIN                                                                     077
      REPEAT                                                                 078
         POS := 0;                                                           079
         READLN(LINE);                                                       080
         LTOKEN(POS,LINE,TOK);   (* GET TOKEN FIELD NAME *)                  081
      UNTIL (TOK = STOK);   (* UNTIL SEARCH TOKEN FOUND *)                   082
   END;                                                                      083
                                                                             084
BEGIN                                                                        085
   REWRITE(OUTFILE);                                                         086
   (* WRITE HEADER INFORMATION TO OUTPUT FILE *)                            087
   WRITELN(OUTFILE,                                                          088
           'TRANDATE   RACFUSER   CONDCODE    CPUTIME   CNCTTIME');         089
   WRITELN(OUTFILE,                                                          090
           '--------   --------   --------    -------   --------');         091
   WHILE NOT EOF DO BEGIN                                                    092
      SKIPTO(TTRANCODE);                                                     093
      POS := POS + 2;   (* SKIP OVER EQUAL SIGN *)                          094
      LTOKEN(POS,LINE,TOK);   (* GET TRANSACTION CODE *)                     095
      READSTR(TOK,TRANCODE);                                                 096
      IF (TRANCODE = JOB) THEN BEGIN                                         097
         SKIPTO(TTRANDATE);                                                  098
         POS := POS + 2;   (* SKIP OVER EQUAL SIGN *)                       099
         LTOKEN(POS,LINE,TOK);           (* GET TRANSACTION DATE *)         100
         READSTR(TOK,TRANDATE);  (* PROCESS ONLY DESIRED TRANSACTIONS *)     101
         IF (MINDATE <= TRANDATE) AND (TRANDATE <= MAXDATE) THEN BEGIN       102
            SKIPTO(TSECONDID);                                               103
            POS := POS + 2;   (* SKIP OVER EQUAL SIGN *)                     104
            (* GET SECOND ID *)                                              105
            LTOKEN(POS,LINE,PREVTOK);   (* ### *)                           106
            LTOKEN(POS,LINE,TOK);       (* - *)                             107
            WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),TOK:LENGTH(TOK));       108
            PREVTOK := TEMPTOK;         (* ###- *)                          109
            LTOKEN(POS,LINE,TOK);       (* ## *)                            110
```

```
WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),TOK:LENGTH(TOK));   111
PREVTOK := TEMPTOK;          (* ###-## *)                    112
LTOKEN(POS,LINE,TOK);        (* - *)                         113
WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),TOK:LENGTH(TOK));   114
PREVTOK := TEMPTOK;          (* ###-##- *)                   115
LTOKEN(POS,LINE,TOK);        (* #### *)                      116
WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),TOK:LENGTH(TOK));   117
TOK := TEMPTOK;              (* ###-##-#### *)               118
(* VERIFY THAT SECOND ID IS NOT 'LIST OFFLINE' COMMAND *)    119
IF (TOK <> WYLBURCMD) THEN BEGIN                             120
   SKIPTO(TTYPETASK);                                        121
   POS := POS + 2;   (* SKIP OVER EQUAL SIGN *)              122
   LTOKEN(POS,LINE,TOK);   (* GET TYPE OF TASK CODE *)       123
   IF (TOK = TJOB) THEN BEGIN   (* BATCH JOB? *)            124
      SKIPTO(TRACFUSER);                                     125
      POS := POS + 2;   (* SKIP OVER EQUAL SIGN *)           126
      LTOKEN(POS,LINE,TOK);   (* GET USER ID *)             127
      RACFUSER := TOK;                                       128
      IF (RACFUSER = 'U11816T') OR (RACFUSER = 'U11813N')    129
      OR (RACFUSER = 'U11813S') OR (RACFUSER = 'U11814A')    130
      OR (RACFUSER = 'U11814D') OR (RACFUSER = 'U11814P')    131
      OR (RACFUSER = 'U11814S') OR (RACFUSER = 'U11816N')    132
      OR (RACFUSER = 'U11815G') OR (RACFUSER = 'U11815L')    133
      OR (RACFUSER = 'U11816G') OR (RACFUSER = 'U11815M')    134
      THEN BEGIN                                             135
         SKIPTO(TCONDCODE);                                  136
         POS := POS + 2;   (* SKIP OVER EQUAL SIGN *)        137
         LTOKEN(POS,LINE,TOK);   (* GET CONDITION CODE *)    138
         READSTR(TOK,CONDCODE);                              139
         SKIPTO(TCPUTIME);                                   140
         POS := POS + 2;   (* SKIP OVER EQUAL SIGN *)        141
         (* GET CPU TIME *)                                  142
         LTOKEN(POS,LINE,PREVTOK);   (* WHOLE DIGITS *)     143
         LTOKEN(POS,LINE,TOK);       (* DECIMAL POINT *)     144
         WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),           145
                          TOK:LENGTH(TOK));                  146
         PREVTOK := TEMPTOK;                                 147
         LTOKEN(POS,LINE,TOK);   (* DECIMAL DIGITS *)       148
         WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),           149
                          TOK:LENGTH(TOK));                  150
         TOK := TEMPTOK;                                     151
         READSTR(TOK,CPUTIME);                               152
         SKIPTO(TCNCTTIME);                                  153
         POS := POS + 2;   (* SKIP OVER EQUAL SIGN *)        154
         (* GET CONNECT TIME *)                              155
         LTOKEN(POS,LINE,PREVTOK);   (* WHOLE DIGITS *)     156
         LTOKEN(POS,LINE,TOK);       (* DECIMAL POINT *)     157
         WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),           158
                          TOK:LENGTH(TOK));                  159
         PREVTOK := TEMPTOK;                                 160
         LTOKEN(POS,LINE,TOK);   (* DECIMAL DIGITS *)       161
         WRITESTR(TEMPTOK,PREVTOK:LENGTH(PREVTOK),           162
                          TOK:LENGTH(TOK));                  163
         TOK := TEMPTOK;                                     164
         READSTR(TOK,CNCTTIME);                              165
```

```
                    WRITELN(OUTFILE,TRANDATE:8,BLANK:4,RACFUSER,          166
                            BLANK:3,CONDCODE:8,BLANK:3,CPUTIME:8:6,       167
                            BLANK:3,CNCTTIME:8:6);                        168
                    SKIPTO(TBATCH);                                       169
                    END                                                   170
                 ELSE SKIPTO(TBATCH);                                     171
                 END                                                      172
              ELSE SKIPTO(TBATCH);                                        173
              END                                                         174
           ELSE SKIPTO(TBATCH);                                           175
           END                                                            176
        ELSE SKIPTO(TBATCH);                                              177
        END                                                               178
     ELSE SKIPTO(TBATCH);                                                 179
  END;    (* WHILE NOT EOF *)                                            180
END.                                                                      181
```

APPENDIX F

LEX & C METRICS PROGRAM

(Note: ~ is represented by ° and ^ is represented by ¬)

Jun 4 12:42 1989 makefile Page 1

```
 1 #*====================================================================*/
 2 #*                  S o f t w a r e   M e t r i c s                   */
 3 #*====================================================================*/
 4 #*                                                                    */
 5 #*   File: makefile (2.3)                                             */
 6 #*   Author: Keith E. Moll                                            */
 7 #*   Date: 89/06/04                                                   */
 8 #*   Class: COMSC 5000 - Thesis                                       */
 9 #*   Adviser: Dr. Mansur Samadzadeh                                   */
10 #*                                                                    */
11 #*====================================================================*/
12 #*                                                                    */
13 #*   Module Description:                                              */
14 #*                                                                    */
15 #*   This is the makefile that compiles and link edits the C          */
16 #*   source code into a single executable called "metrics".           */
17 #*                                                                    */
18 #*   Usage:                                                           */
19 #*                                                                    */
20 #*   At the unix prompt, type in "make"                               */
21 #*                                                                    */
22 #*   Input Required:                                                  */
23 #*                                                                    */
24 #*   None.                                                            */
25 #*                                                                    */
26 #*   Output Produced                                                  */
27 #*                                                                    */
28 #*   Executable called "metrics".                                     */
29 #*                                                                    */
30 #*====================================================================*/
31
32 CFLAGS=  -O # -g  # -O for optimization, -g for symbolic debugging (dbx)
33 LOADLIBES= -ll -lm   # Load Libraries
34 GET= get           # get current SCCS file
35 GFLAGS= -s         # get flags: silent option
36 SCCSFILES=  SCCS/s.se.c SCCS/s.metrics.l SCCS/s.global.c SCCS/s.defs.h\
37             SCCS/s.global.h SCCS/s.halstead.c
38 OBJS= se.o halstead.o global.o metrics.o
39
40 .c~.c:                                  # SCCS .c files to .c
41       $(GET) $(GFLAGS) -p $< > $*.c     # -p is for pipe option
42 .h~.h:                                  # SCCS .h files to .h
43       $(GET) $(GFLAGS) -p $< > $*.h     # -p is for pipe option
44 .l~.l:                                  # SCCS .l files to .l
45       $(GET) $(GFLAGS) -p $< > $*.l     # -p is for pipe option
46
47 all:  getSCCS se cleanup
48
49 getSCCS:    $(SCCSFILES)
50
```

Jun 4 12:42 1989 makefile Page 2

```
51 defs.h:          SCCS/s.defs.h
52 global.h:        SCCS/s.global.h
53 global.c:        SCCS/s.global.c
54 se.c:            SCCS/s.se.c
55 metrics.l:       SCCS/s.metrics.l
56 halstead.c:      SCCS/s.halstead.c
57
58 se:    $(OBJS)
59              cc $(CFLAGS) $(OBJS) $(LOADLIBES) -o metrics
60
61 $(OBJS):      defs.h   global.h
62
63 cleanup:
64              rm *.o
```

Jun 4 12:04 1989 defs.h Page 1

```
 1 /*====================================================================*/
 2 /*                 S o f t w a r e   M e t r i c s                    */
 3 /*====================================================================*/
 4 /*                                                                    */
 5 /*   File: defs.h (2.1)                                               */
 6 /*   Author: Keith E. Moll                                            */
 7 /*   Date: 89/06/04                                                   */
 8 /*   Class: COMSC 5000 - Thesis                                       */
 9 /*   Adviser: Dr. Mansur Samadzadeh                                   */
10 /*                                                                    */
11 /*====================================================================*/
12 /*                                                                    */
13 /*   Module Description:                                              */
14 /*                                                                    */
15 /*   This module defines all symbolic constants used in the other     */
16 /*   source files.                                                    */
17 /*                                                                    */
18 /*   Usage:                                                           */
19 /*                                                                    */
20 /*   This module should be #include'ed in all other files which       */
21 /*   use any of the symbolic constants.                               */
22 /*                                                                    */
23 /*====================================================================*/
24
25 /* debug macro */
26 #define  DEBUG(level,format,p1)  if (level & VOLUME) \
27                                     fprintf(stderr,format,p1);
28
29 /* debug volume and levels */
30 #define  VOLUME   00    /* debug VOLUME */
31 #define  OPERATOR 01    /* echo print OPERATORs */
32 #define  OPERAND  02    /* echo print OPERANDs */
33 #define  STRLEN   04    /* yytext STRing LENgth of quoted strings */
34 #define  PARENS   08    /* echo print yytext when RE "(" or ")" matches */
35
36 #define NOTFOUND  -1 /* flag that token NOT FOUND in current list */
37 #define MAXTOKLEN 20 /* MAXimum TOKen LENgth of any token */
38 #define MAXOPR 250   /* MAXimum no. of OPeratoRs that can be handled */
39 #define MAXOPD 700   /* MAXimum no. of OPeranDs that can be handled */
40 #define MAXSUB 100   /* MAXimum no. of SUBprograms that can be handled */
41 #define MAXTYPE   20 /* MAXimum no. of user-defined TYPEs allowed */
42 #define NO     0     /* used as boolean value for false */
43 #define FALSE  0     /* used as boolean value for false */
44 #define YES    1     /* used as boolean value for true */
45 #define TRUE   1     /* used as boolean value for true */
46
47 /* warning numbers */
48 #define OPRFULL      1  /* OPeratoR arrays FULL */
49 #define OPDFULL      2  /* OPeranD arrays FULL */
50 #define SUBPGMFULL   3  /* SUBProGraM array FULL */
```

```
 1 /*====================================================================*/
 2 /*                  S o f t w a r e   M e t r i c s                   */
 3 /*====================================================================*/
 4 /*                                                                    */
 5 /*   File: global.c (2.2)                                             */
 6 /*   Author: Keith E. Moll                                            */
 7 /*   Date: 89/06/04                                                   */
 8 /*   Class: COMSC 5000 - Thesis                                       */
 9 /*   Adviser: Dr. Mansur Samadzadeh                                   */
10 /*                                                                    */
11 /*====================================================================*/
12 /*                                                                    */
13 /*   Module Description:                                              */
14 /*                                                                    */
15 /*   This module contains declarations of all global variables.       */
16 /*                                                                    */
17 /*====================================================================*/
18
19 #include "defs.h"
20
21 /* token arrays to store ... */
22 char  opr[MAXOPR][MAXTOKLEN],     /* operators */
23       opd[MAXOPD][MAXTOKLEN],     /* operands */
24       subpgm[MAXSUB][MAXTOKLEN], /* subprogram names: both
25                                     user-defined and system */
26       utype[MAXTYPE][MAXTOKLEN]; /* user-defined types */
27
28 int
29       vgincr[MAXSUB],    /* tells whether or not to incr. vg */
30       nconst = 0,        /* number of #define constants */
31       ntype = 0,         /* number of typedef'initions */
32       nvar = 0,          /* number of variables */
33       nsub = 0;          /* number of subprograms */
34
35 /* arrays to store operator and operand frequencies */
36 int nopr[MAXOPR], nopd[MAXOPD];
37
38 int
39       loc = 0,     /* lines of code */
40       inc = 0,     /* value by which to increment loc by: 0 or 1 */
41       vg = 0;      /* McCabe's cyclomatic number */
42
43 /* software science metrics */
44 int
45       plen,        /* program length */
46       eta1 = 0,    /* number of unique operators */
47       N1 = 0,      /* total number of operators */
48       eta2 = 0,    /* number of unique operands */
49       N2 = 0;      /* total number of operands */
50 double
```

```
51        eplen,   /* estimated program length */
52        pvol,    /* program volume */
53        eplev,   /* estimated program level */
54        epdif,   /* estimated program difficulty */
55        eppvol,  /* estimated potential program volume */
56        effort,  /* number of elementary mental discriminations */
57        time,    /* implementation time */
58        lambda;  /* language level */
59
60 /* flags */
61 int
62        fvar = NO;  /* set if processing a declaration list */
63        ftype = NO; /* set if processing a type definition */
64        fcond = NO, /* set if processing a conditional */
65        fsquote=NO, /* set if processing a str delimited by single quotes */
66        fdquote=NO, /* set if processing a str delimited by double quotes */
67        fparen = 0, /* incr. on left paren, decr. on right paren */
68        fbrace = 0, /* incr. on left brace, decr. on right brace */
69        fbracket = 0;  /* incr. on left bracket, decr. on right bracket */
70
71 /* C tokens */
72 char  tparens[] = { "( )" };
73 char  tbraces[] = { "{ }" };
74 char  tbrackets[] = { "[ ]" };
75 char  tternary[] = { "? :" };
76 char  tif[] = { "if" };
77 char  tifelse[] = { "if else" };
```

```
 1 /*====================================================================*/
 2 /*                  S o f t w a r e   M e t r i c s                   */
 3 /*====================================================================*/
 4 /*                                                                    */
 5 /*   File: global.h (2.2)                                             */
 6 /*   Author: Keith E. Moll                                            */
 7 /*   Date: 89/06/04                                                   */
 8 /*   Class: COMSC 5000 - Thesis                                       */
 9 /*   Adviser: Dr. Mansur Samadzadeh                                   */
10 /*                                                                    */
11 /*====================================================================*/
12 /*                                                                    */
13 /*   Module Description:                                              */
14 /*                                                                    */
15 /*   This module contains descriptions of all external (global)       */
16 /*   variables.                                                       */
17 /*                                                                    */
18 /*   Usage:                                                           */
19 /*                                                                    */
20 /*   This module should be #include'ed in all other files which       */
21 /*   use any of global variables.                                     */
22 /*                                                                    */
23 /*====================================================================*/
24
25 #include <stdio.h>
26 #include "defs.h"
27
28 /* external functions */
29 extern halstead();
30
31 /* token arrays to store ... */
32 extern char
33             opr[][MAXTOKLEN],    /* operators */
34             opd[][MAXTOKLEN],    /* operands */
35             subpgm[][MAXTOKLEN], /* subprogram names: both
36                                     user-defined and system */
37             utype[][MAXTOKLEN];  /* user-defined types */
38
39 extern int
40          vgincr[],   /* tells whether or not to incr. vg */
41          nconst,     /* number of #define constants */
42          ntype,      /* number of typedef'initions */
43          nvar,       /* number of variables */
44          nsub;       /* number of subprograms */
45
46 /* arrays to store operator and operand frequencies */
47 extern int nopr[], nopd[];
48
49 extern int
50       loc,     /* lines of code */
```

```
51        inc,      /* value by which to increment loc by: 0 or 1 */
52        vg;       /* McCabe's cyclomatic number */
53
54 /* software science metrics */
55 extern int
56        plen,     /* program length */
57        eta1,     /* number of unique operators */
58        N1,       /* total number of operators */
59        eta2,     /* number of unique operands */
60        N2;       /* total number of operands */
61 extern double
62        eplen,    /* estimated program length */
63        pvol,     /* program volume */
64        eplev,    /* estimated program level */
65        epdif,    /* estimated program difficulty */
66        eppvol,   /* estimated potential program volume */
67        effort,   /* number of elementary mental discriminations */
68        time,     /* implementation time */
69        lambda;   /* language level */
70
71 /* flags */
72 extern   int
73        ftype,    /* set if processing a type definition */
74        fvar,     /* set if processing a variable declaration */
75        fcond,    /* set if processing a conditional */
76        fcom,     /* set if processing a comment */
77        fsquote,  /* set if processing a str delimited by single quotes */
78        fdquote,  /* set if processing a str delimited by double quotes */
79        fparen,   /* incr. on left paren, decr. on right paren */
80        fbrace,   /* incr. on left brace, decr. on right brace */
81        fbracket;    /* incr. on left bracket, decr. on right bracket */
82
83 /* C tokens */
84 extern char tparens[];
85 extern char tbraces[];
86 extern char tbrackets[];
87 extern char tternary[];
88 extern char tif[];
89 extern char tifelse[];
```

```
 1 /*====================================================================*/
 2 /*                 S o f t w a r e   M e t r i c s                    */
 3 /*====================================================================*/
 4 /*                                                                    */
 5 /*   File: halstead.c (2.1)                                           */
 6 /*   Author: Keith E. Moll                                            */
 7 /*   Date: 89/06/04                                                   */
 8 /*   Class: COMSC 5000 - Thesis                                       */
 9 /*   Adviser: Dr. Mansur Samadzadeh                                   */
10 /*                                                                    */
11 /*====================================================================*/
12 /*                                                                    */
13 /*   Module Description:                                              */
14 /*                                                                    */
15 /*   This module computes all of Halstead's Software Science          */
16 /*   metrics given the four basic metrics as input.                   */
17 /*                                                                    */
18 /*   Usage:                                                           */
19 /*                                                                    */
20 /*   To be called from another subprogram.                            */
21 /*                                                                    */
22 /*   Input Required:                                                  */
23 /*                                                                    */
24 /*   Number of unique operators (eta1) and operands (eta2) and        */
25 /*   frequency counts of each operator (nopr) and operand (nopd)      */
26 /*   are required input.                                              */
27 /*                                                                    */
28 /*   Output Produced:                                                 */
29 /*                                                                    */
30 /*   Total number of operators (N1) and operands (N2), program        */
31 /*   length (plen), estimated program length (eplen), program         */
32 /*   volume (pvol), estimated program level (eplev), estimated        */
33 /*   program difficulty (epdif), estimated potential program          */
34 /*   volume (eppvol), number of elementary mental discriminations     */
35 /*   (effort), implementation time in seconds (time), and language    */
36 /*   level (lambda) are all output.                                   */
37 /*                                                                    */
38 /*====================================================================*/
39
40 #include <stdio.h>
41 #include <math.h>
42 #include "global.h"
43 #include "defs.h"
44
45 halstead(eta1, eta2, nopr, nopd, N1, N2, plen, eplen, pvol, eplev, epdif,
46          eppvol, effort, time, lambda)
47 int   eta1, eta2,
48       nopr[], nopd[],
49       *N1, *N2,
50       *plen;
```

```
51 double   *eplen,
52          *pvol, *eplev, *epdif, *eppvol,
53          *effort, *time, *lambda;
54 {
55    int i;
56    double beta = 18;     /* Stroud number: normally set to 18 since
57                             this gave the best results in Halstead's
58                             earliest experiments */
59
60    for (i = 0; i < eta1; i++)     /* compute total number of operators */
61       *N1 += nopr[i];
62    for (i = 0; i < eta2; i++)     /* compute total number of operands */
63       *N2 += nopd[i];
64
65    *plen = *N1 + *N2;
66
67    if ((eta1 > 0) && (eta2 > 0))
68       *eplen = eta1 * log((double) eta1) / log((double) 2) +
69                eta2 * log((double) eta2) / log((double) 2);
70
71    if ((eta1+eta2) > 0)
72       *pvol = *plen * log((double) (eta1+eta2)) / log((double) 2);
73
74    if ((eta1 != 0) && (*N2 != 0)) {
75       *eplev = ((double) 2 / eta1) * ((double) eta2 / *N2);
76       *epdif = 1 / *eplev;
77       *eppvol = *pvol * *eplev;
78    }
79
80    if (eta2 > 0)  /* formula rewritten to prevent overflow errors */
81       *effort = (log((double) (eta1+eta2)) / log((double) 2)) /
82                 (2 * eta2) * eta1 * *N2 * *plen;
83
84    *time = *effort / beta;
85    *lambda = pow((double) *eplev,(double) 2) * *pvol;
86
87    return (((eta1 == 0) || (eta2 == 0)) ? 0 : 1);
88 }
```

```
 1 %{
 2 /*=====================================================================*/
 3 /*                  S o f t w a r e   M e t r i c s                    */
 4 /*=====================================================================*/
 5 /*                                                                     */
 6 /*   File: metrics.l (2.2)                                             */
 7 /*   Author: Keith E. Moll                                             */
 8 /*   Date: 89/06/04                                                    */
 9 /*   Class: COMSC 5000 - Thesis                                        */
10 /*   Adviser: Dr. Mansur Samadzadeh                                    */
11 /*                                                                     */
12 /*=====================================================================*/
13 /*                                                                     */
14 /*   Module Description:                                               */
15 /*                                                                     */
16 /*   This module contains the lex specifications and subprograms       */
17 /*   for finding operators and operands and counting them as such.     */
18 /*                                                                     */
19 /*   Usage:                                                            */
20 /*                                                                     */
21 /*   This module is called yylex().  It extracts tokens from a         */
22 /*   C program and computes lines of code, McCabe's cyclomatic         */
23 /*   number, and Halstead's Software Science metrics.                  */
24 /*                                                                     */
25 /*=====================================================================*/
26
27 #include <stdio.h>
28 #include "global.h"
29 #include "defs.h"
30
31 #define  LEXDEBUG YES   /* turn on lex debug output */
32
33 int   i,
34       save,    /* save variable for index into an array */
35       len;     /* length of yytext */
36 char  c;
37
38 /*   No distinction is made between unary and binary minus.            */
39 /*   Furthermore, bitwise "&" and address operator "&" are not         */
40 /*   differentiated.  The same is true for arithmetic "*" and pointer  */
41 /*   operator "*".  Also, while loops and do-while loops are not       */
42 /*   differentiated.  In general, when a token has multiple meanings   */
43 /*   its separate usages are not distinguished.                        */
44
45 %}
46    /* OCTal DIGits */
47 OCTDIG   [0-7]
48    /* LETTERs used in identifiers */
49 LETTER   [A-Za-z]
50    /* decimal DIGITs */
```

```
 51 DIGIT    [0-9]
 52    /* IDENTifiers */
 53 IDENT    {LETTER}({LETTER}|{DIGIT}|"_"|"$")*
 54    /* INTeger NUMbers */
 55 INTNUM   {DIGIT}+
 56    /* FLOAT no. type 1 */
 57 FLOAT1   {DIGIT}*"."{DIGIT}+([Ee][+-]?{DIGIT}+)?
 58    /* FLOAT no. type 2 */
 59 FLOAT2   {DIGIT}+"."{DIGIT}*([Ee][+-]?{DIGIT}+)?
 60    /* FLOAT no. type 3 */
 61 FLOAT3   {DIGIT}+[Ee][+-]?{DIGIT}+
 62    /* HEXadecimal CHAR. */
 63 HEXCHAR  [A-Fa-f]
 64    /* HEXadecimal NUM. */
 65 HEXNUM   0[Xx][{DIGIT}{HEXCHAR}]+
 66 %%
 67 while          {
 68                    inc = 1;
 69                    fcond = TRUE;
 70                    vg++;
 71                    procopr(yytext);
 72                 }
 73 case |
 74 default        {
 75                    inc = 1;
 76                    vg++;
 77                    procopr(yytext);
 78                 }
 79 for            {
 80                    inc = 1;
 81                    vg++;
 82                    procopr(yytext);
 83                 }
 84 main           {
 85                    inc = 1;
 86                    vg++;
 87                    procopr(yytext);
 88                 }
 89 if             {
 90                    inc = 1;
 91                    fcond = TRUE;
 92                    vg++;
 93                    procopr(tif);
 94                 }
 95 else           {
 96                    inc = 1;
 97                    procopr(tifelse);
 98                    i = lsearch(tif,opr,eta1); /* remove previously */
 99                    nopr[i] -= 1;              /* counted if token */
100                 }
```

```
"&&"            {
                   inc = 1;
                   if (fcond) vg++;
                   procopr(yytext);
                }
"||"            {
                   inc = 1;
                   if (fcond) vg++;
                   procopr(yytext);
                }
"{"             {
                   inc = 1;
                   fbrace++;
                   procopr(tbraces);
                }
"}"             {
                   inc = 1;
                   fbrace--;
                }
"("             {
                   inc = 1;
                   fparen++;
                   procopr(tparens);
                   DEBUG(PARENS,"yytext=%s\n",yytext);
                }
")"             {
                   inc = 1;
                   fparen--;
                   if (fparen == 0) fcond = FALSE;
                   DEBUG(PARENS,"yytext=%s\n",yytext);
                }
"["             {
                   inc = 1;
                   fbracket++;
                   procopr(tbrackets);
                }
"]"             {
                   inc = 1;
                   fbracket--;
                }
"\\"            {  /* line continuation character: ignore it */
                }
\#define        {  /* #define defines a string substitution which is */
                   nconst++;   /* counted as a constant */
                   inc = 1;
                   procopr(yytext);
                }
typedef         {
                   inc = 1;
                   ftype = TRUE;  /* types follow typedef */
```

```
151                     procopr(yytext);
152                 }
153 int |
154 float |
155 char |
156 short |
157 long |
158 unsigned |
159 double |
160 struct |
161 union |
162 auto |
163 static |
164 extern |
165 FILE |   /* include so that file descriptors will be counted as variables */

166 register        {
167                     inc = 1;
168                     if (!ftype)     /* if not within type definition, then */
169                        fvar = TRUE;    /* variable declarations follow */
170                     procopr(yytext);
171                 }
172 ";"             {
173                     inc = 1;
174                     ftype = FALSE; /* semicolon ends type definitions */
175                     fvar = FALSE;  /* and variable declarations */
176                     procopr(yytext);
177                 }
178 "+" |
179 "-" |
180 "*" |
181 "/" |
182 "%" |
183 "~" |
184 "!" |
185 "&" |
186 "|" |
187 "¬" |
188 "++" |
189 "--" |
190 "+=" |
191 "-=" |
192 "*=" |
193 "/=" |
194 "%=" |
195 "&=" |
196 "|=" |
197 "¬=" |
198 "=" |
199 "<<" |
200 ">>". |
```

```
"<<=" |
">>=" |
"==" |
"!=" |
"<" |
">" |
"<=" |
">=" |
"," |
"->" |
"." |
"?" |
":" |
\\[(\\)Ontb({OCTDIG}{OCTDIG}{OCTDIG})] |  /* escape characters */
switch |
do |
return |
goto |
break |
continue |
sizeof |
\#include |
\#if |
\#ifdef |
\#ifndef |
\#else |
\#endif |
\#line          {
                   inc = 1;    /* all other operators */
                   procopr(yytext);
                }
"/*"            {  /* skip all comments */
                   loop:
                   while (input() != '*');
                   switch (input())  {
                      case '/' : break;
                      case '*' : unput('*');
                      default : goto loop;
                   }
                }
\"              {  /* character string: everything inside the double
                      quotes is counted as one operand */
                   inc = 1;
                   i = strlen(yytext);
                   DEBUG(STRLEN,"before: length of yytext=%d\n",i);
                   DEBUG(STRLEN,"yytext=%s\n",yytext);
                   do {  /* add characters to operand until next " read */
                      c = input();
                      if (i < MAXTOKLEN) yytext[i++] = c;
                   } while (c != '"');
```

```
251                     /* make sure that operand terminates with
252                        end-of-string marker */
253                     if (i < MAXTOKLEN)
254                        yytext[i] = '\0';
255                     else
256                        yytext[MAXTOKLEN-1] = '\0';
257                     DEBUG(STRLEN,"after: length of yytext=%d\n",i);
258                     DEBUG(STRLEN,"yytext=%s\n",yytext);
259                     procopd(yytext);
260                  }
261 \'               {  /* character constant: everything inside the single
262                        quotes is counted as one operand; this should be
263                        one character unless it is an escape sequence */
264                     inc = 1;
265                     i = strlen(yytext);
266                     DEBUG(STRLEN,"before: length of yytext=%d\n",i);
267                     DEBUG(STRLEN,"yytext=%s\n",yytext);
268                     do {  /* add chars to operand until next ' read */
269                        c = input();
270                        if (i < MAXTOKLEN) yytext[i++] = c;
271                     } while (c != '\'');
272                     /* make sure that operand terminates with
273                        end-of-string marker */
274                     if (i < MAXTOKLEN)
275                        yytext[i] = '\0';
276                     else
277                        yytext[MAXTOKLEN-1] = '\0';
278                     DEBUG(STRLEN,"after: length of yytext=%d\n",i);
279                     DEBUG(STRLEN,"yytext=%s\n",yytext);
280                     procopd(yytext);
281                  }
282 " "              {  /* no operation: blanks are ignored */
283                  }
284 \<{IDENT}(\/?{IDENT})*\.{LETTER}\> | /* sys include filenames are opds */
285 {IDENT}          {  /* identifiers */
286                     inc = 1;
287                     do {   /* skip whitespace to next token */
288                        c = input();
289                     } while ((c == ' ') || (c == '\t'));
290                     unput(c);
291                     if (c == '(') {   /* if left paren, then subpgm name */
292                        i = lsearch(yytext,subpgm,nsub);   /* look for it */
293                        if (i == NOTFOUND) /* if not in list, then add it */
294                           {
295                           i = nsub;
296                           vgincr[i] = NO;   /* set so that vg is incre-
297                                                mented properly below */
298                           /* copy subprogram name into list */
299                           strcpy(subpgm[nsub++],yytext);
300                           }
```

```
301                     save = i;   /* save index into subpgm for use below */
302                     /* look for subpgm name as an operator */
303                     i = lsearch(yytext,opr,eta1);
304                     if (i == NOTFOUND)   /* if not in list, then add it */
305                        {
306                        if (eta1 < MAXOPR)   /* if list is not full */
307                           {
308                           i = eta1;
309                           strcpy(opr[i],yytext);
310                           nopr[i] = 0;
311                           DEBUG(OPERATOR,"opr[eta1]: %s",opr[eta1]);
312                           DEBUG(OPERATOR,", yytext=%s\n",yytext);
313                           eta1++;
314                           }
315                        else
316                           {  /* report that operator list is full */
317                           warning(OPRFULL);
318                           }
319                        }
320                     /* count calls to subpgms as operators; calls are
321                        inside braces */
322                     if (fbrace != 0)
323                        nopr[i]++;
324                     else  /* incr. vg on each subprogram def'n. */
325                        {
326                        if (!vgincr[save]) vg++;
327                        vgincr[save] = YES;
328                        }
329                     }
330                  else
331                     {  /* else must be an operand...
332                           if it has not been predefined as a macro
333                           function: macro functions are #define strings
334                           that contain a left parenthesis */
335                     i = lsearch(yytext,opr,eta1);   /* look for macro */
336                     if (i != NOTFOUND)
337                        {
338                        procopr(yytext);
339                        }
340                     else
341                        {  /* variable or user-defined operand */
342                        /* if not defining a type and operand in the
343                           user-defined type list, then variable
344                           declaration follows this operand */
345                        if ((!ftype) &&
346                           (lsearch(yytext,utype,ntype) != NOTFOUND))
347                           fvar = TRUE;
348                        else
349                           {  /* operand is a variable or user-defined
350                                 type name */
```

```
351                             i = lsearch(yytext,opd,eta2);
352                             /* if operand is part of a variable declar-
353                                ation statement and is not already in
354                                operand list, then count it as an
355                                occurrence of a unique variable */
356                             if ((fvar) && (i == NOTFOUND)) nvar++;
357                             i = lsearch(yytext,utype,ntype);
358                             /* if operand is part of a type definition
359                                statement and is not already in user-defined
360                                type list, then count it as an occurrence of
361                                a unique user-defined type */
362                             if ((ftype) && (i == NOTFOUND))
363                                {
364                                /* add user-defined type to list
365                                   and count it */
366                                strcpy(utype[ntype],yytext);
367                                ntype++;
368                                }
369                             }
370                          /* count variables and user-defined
371                             types as operands */
372                          procopd(yytext);
373                          }
374                       }
375                 }
376 {FLOAT1} |
377 {FLOAT2} |
378 {FLOAT3} |
379 {HEXNUM} |
380 {INTNUM}        {  /* count floating point, hexidecimal, and integer
381                       numbers as operands */
382                    inc = 1;
383                    procopd(yytext);
384                 }
385 \t              {  /* skip tabs */
386                 }
387 \n              {  /* at the end of each physical line increment loc
388                       metric by inc: inc is set above if at least one
389                       token appears outside a comment */
390                    loc += inc;
391                    inc = 0;    /* reset inc */
392                 }
393 %%
394
395 warning(warnum)
396
397 /*================================================================*/
398 /*                                                                */
399 /*   warning:                                                     */
400 /*                                                                */
```

```
401 /*    This function prints warning messages given a warning number    */
402 /*    (warnum).                                                       */
403 /*                                                                    */
404 /*====================================================================*/
405
406 int warnum;
407 {
408     switch (warnum) {
409     case OPRFULL: printf("warning: operator arrays are full\n"); break;
410     case OPDFULL: printf("warning: operand arrays are full\n"); break;
411     case SUBPGMFULL: printf("warning: subprogram array is full\n"); break;
412     }
413 }
414
415
416 /*====================================================================*/
417 /*                                                                    */
418 /*    lsearch:                                                        */
419 /*                                                                    */
420 /*    This function does a linear search for tok in list with max     */
421 /*    elements.                                                       */
422 /*                                                                    */
423 /*====================================================================*/
424
425 lsearch(tok,list,max)
426 char tok[], list[][MAXTOKLEN];
427 int max;
428 {
429     int i, found;
430
431     i = 0;
432     found = NO;
433     while ((i < max) && !found)   /* search entire list */
434        found = !(strcmp(list[i++],tok));
435     return((found) ? (i-1) : NOTFOUND);
436 }
437
438
439 /*====================================================================*/
440 /*                                                                    */
441 /*    procopd:                                                        */
442 /*                                                                    */
443 /*    This function processes tok as an operand by adding it to the   */
444 /*    operand list if it is not already in it and by counting an      */
445 /*    occurrence of it.                                               */
446 /*                                                                    */
447 /*====================================================================*/
448
449 procopd(tok)
450 char tok[];
```

```
451 {
452    int i;
453
454    i = lsearch(tok,opd,eta2);     /* search for token */
455
456    /* if not in list and list not full, then add operand to list */
457    if (i == NOTFOUND) {
458       if (eta2 < MAXOPD) {
459          strcpy(opd[eta2],tok);
460          DEBUG(OPERAND,"opd[eta2]: %s",opd[eta2]);
461          DEBUG(OPERAND,", tok=%s\n",tok);
462          nopd[eta2] = 1;
463          eta2++;
464       } else {
465          warning(OPDFULL);
466       }
467    } else {   /* else operator already in list, just count */
468       nopd[i]++;   /* another occurrence of it */
469    }
470 }
471
472
473 /*====================================================================*/
474 /*                                                                    */
475 /*   procopr:                                                         */
476 /*                                                                    */
477 /*   This function processes tok as an operator by adding it to the */
478 /*   operator list if it is not already in it and by counting an    */
479 /*   occurrence of it.                                                */
480 /*                                                                    */
481 /*====================================================================*/
482
483 procopr(tok)
484 char tok[];
485 {
486    int i;
487
488    i = lsearch(tok,opr,eta1);     /* search for token */
489
490    /* if not in list and list not full, then add operator to list */
491    if (i == NOTFOUND) {
492       if (eta1 < MAXOPR) {
493          strcpy(opr[eta1],tok);
494          DEBUG(OPERATOR,"opr[eta1]: %s",opr[eta1]);
495          DEBUG(OPERATOR,", tok=%s\n",tok);
496          nopr[eta1] = 1;
497          eta1++;
498       } else {
499          warning(OPRFULL);
500       }
```

```
501     } else {   /* else operator already in list, just count */
502        nopr[i]++;   /* another occurrence of it */
503     }
504 }
```

```
 1 /*====================================================================*/
 2 /*                  S o f t w a r e   M e t r i c s                   */
 3 /*====================================================================*/
 4 /*                                                                    */
 5 /*   File: se.c (2.2)                                                 */
 6 /*   Author: Keith E. Moll                                            */
 7 /*   Date: 89/06/04                                                   */
 8 /*   Class: COMSC 5000 - Thesis                                       */
 9 /*   Adviser: Dr. Mansur Samadzadeh                                   */
10 /*                                                                    */
11 /*====================================================================*/
12 /*                                                                    */
13 /*   Module Description:                                              */
14 /*                                                                    */
15 /*   This is the main module.  It calls the yylex() lexical           */
16 /*   analyzer which tokenizes the input file, a C program.            */
17 /*                                                                    */
18 /*   Usage:                                                           */
19 /*                                                                    */
20 /*   At the unix prompt, type in "metrics < file", where file         */
21 /*   is a file containing C source code.  Or, if there is a need      */
22 /*   to compute metrics on a stream of C programs, then use           */
23 /*   "cat file1 [file2 ...] | metrics".                               */
24 /*      Optional usage is "metrics id < file", where id is the        */
25 /*   id of the program for which the metrics are being computed.      */
26 /*   Of course, "cat file1 [file2 ...] | metrics id" is valid.        */
27 /*                                                                    */
28 /*   Input Required:                                                  */
29 /*                                                                    */
30 /*   This module requires a C source file as input.  The file         */
31 /*   must contain C source code which can include preprocessor        */
32 /*   directives.  As mentioned above, more than one program can be    */
33 /*   input to this program.  It is assumed that all programs are      */
34 /*   syntactically correct: this program does not check syntax.       */
35 /*                                                                    */
36 /*   Output Produced:                                                 */
37 /*                                                                    */
38 /*   This module prints the following metrics:                        */
39 /*                                                                    */
40 /*      1.  Lines of code (LOC)                                       */
41 /*      2.  McCabe's cyclomatic number (vg)                           */
42 /*      3.  Halstead's Software Science metrics                       */
43 /*                                                                    */
44 /*   These metrics are printed to stdout.  Use redirection ">" to     */
45 /*   capture them in a file.  A list of all operators and operands    */
46 /*   is printed to stderr.  Use redirection "2>" to capture this      */
47 /*   in a file.                                                       */
48 /*                                                                    */
49 /*====================================================================*/
50
```

```
#include <stdio.h>
#include "global.h"
#include "defs.h"

main(argc, argv)
int argc;
char *argv[];
{
    int i, j;
    int   N1,      /* total number of operators */
          N2,      /* total number of operands */
          plen;    /* program length */
    double eplen,  /* estimated program length */
          pvol,    /* program volume */
          eplev,   /* estimated program level */
          epdif,   /* estimated program difficulty */
          eppvol,  /* estimated potential program volume */
          effort,  /* number of elementary mental discriminations */
          time,    /* implementation time in seconds */
          lambda;  /* language level */

    while (yylex() != 0);   /* process all tokens until end-of-file */

    /* report metrics */
    fprintf(stderr,"\n\nSoftware Metrics:\n");
    fprintf(stderr,"loc=%d\n",loc);
    printf("UID=%s", strcmp(argv[1],"<") ? argv[1] : " ");
    printf("\nLOC=%1d\n",loc);
    fprintf(stderr,"vg=%d\n",vg);
    printf("VG=%1d\n",vg);
    printf("NVAR=%1d NCONST=%1d NTYPE=%1d NSUB=%1d\n",
            nvar,nconst,ntype,nsub);

    /* compute Halstead's Software Science Metrics */
    halstead(eta1, eta2, nopr, nopd, &N1, &N2, &plen, &eplen, &pvol,
            &eplev, &epdif, &eppvol, &effort, &time, &lambda);

    if (((eta1+1)>MAXOPR) || ((eta2+1)>MAXOPD) || ((nsub+1)>MAXSUB)) {
       fprintf(stderr,"Warning: following metrics will not be accurate\n");
       fprintf(stderr,"         because not all tokens could be counted.\n");

       printf("Warning: following metrics will not be accurate\n");
       printf("         because not all tokens could be counted.\n");
    }

    fprintf(stderr,"Halstead's metrics:\n");
    fprintf(stderr,"eta1=%d\n",eta1);
    fprintf(stderr,"eta2=%d\n",eta2);
    fprintf(stderr,"\noperators:\n");
    for (i = 0; i < eta1; i++) {
       fprintf(stderr,"%s %d\n",opr[i],nopr[i]);
```

```
101     }
102     fprintf(stderr,"\nTotal # of operators = %d\n",N1);
103     fprintf(stderr,"\n\noperands:\n");
104     for (i = 0; i < eta2; i++) {
105        fprintf(stderr,"%s %d\n",opd[i],nopd[i]);
106     }
107     fprintf(stderr,"\nTotal # of operands = %d\n\n",N2);
108     printf("ETA1=%1d N1=%1d ETA2=%1d N2=%1d\n",eta1,N1,eta2,N2);
109     printf("PLEN=%d EPLEN=%4.2f",plen,eplen);
110     printf(" PVOL=%4.2f EPPVOL=%4.2f\n",pvol,eppvol);
111     printf("EPLEV=%6.4f EPDIF=%5.3f\n",eplev,epdif);
112     printf("EFFORT=%2.0f TIME=%4.2f LAMBDA=%4.2f\n",effort,time,lambda);
113 }
```

APPENDIX G

SAMPLE C PROGRAM USED TO TEST LEX & C METRICS PROGRAM; INCLUDES SAMPLE INPUT, OUTPUT, DETAILED METRICS, AND SUMMARY METRICS

May 29 13:38 1989 sample.c Page 1

```
 1 /*====================================================================*/
 2 /*                  S o f t w a r e   M e t r i c s                   */
 3 /*====================================================================*/
 4 /*                                                                    */
 5 /*   File: sample.c (2.2)                                             */
 6 /*   Author: Keith E. Moll                                            */
 7 /*   Date: 89/05/29                                                   */
 8 /*   Class: COMSC 5000 - Thesis                                       */
 9 /*   Adviser: Dr. Mansur Samadzadeh                                   */
10 /*                                                                    */
11 /*====================================================================*/
12 /*                                                                    */
13 /*   Module Description:                                              */
14 /*                                                                    */
15 /*   This module is a sample C program used to illustrate how the     */
16 /*   lines of code, McCabe's cyclomatic number, and Halstead's        */
17 /*   Software Science metrics are computed.                           */
18 /*                                                                    */
19 /*   This sample program reads in a list of up to MAXELE real         */
20 /*   numbers and counts, sums, finds the minimum and maximum, and.    */
21 /*   mean of the list.                                                */
22 /*                                                                    */
23 /*   Usage:                                                           */
24 /*                                                                    */
25 /*   At the unix prompt, type in "sample".                            */
26 /*                                                                    */
27 /*   Input Required:                                                  */
28 /*                                                                    */
29 /*   A file, called sample.data, of numbers is required for input.    */
30 /*                                                                    */
31 /*   Output Produced:                                                 */
32 /*                                                                    */
33 /*   This module prints the number of numbers in the list (count),    */
34 /*   the sum, the minimum number, the maximum, and the mean.          */
35 /*                                                                    */
36 /*====================================================================*/
37
38 #include <stdio.h>
39 #define  MAXELE   20
40 #define  INFILE   "sample.data"
41
42 double mean();
43 typedef float real;
44
45 main()
46 {
47    real a[MAXELE],       /* measurements */
48         meas,            /* temporary variable into which measurements are
49                             read */
50         sum,             /* sum of measurements */
```

```
51          min,            /* minimum measurement */
52          max;            /* maximum measurement */
53      int   cnt,      /* number of measurements */
54          i;
55      FILE *fp;
56      int   fs;       /* file status */
57
58      /* initialization */
59      sum = 0.0e0;
60      cnt = 0;
61      a[0] = 0.0;     /* in case no input data */
62
63      /* read, count, and sum measurements */
64      fp = fopen(INFILE,"r");
65      if (fp == NULL) {
66         printf("error: can't open file\n");
67      } else {
68         fs = fscanf(fp,"%f",&meas);       /* read first measurement */
69         while ((fs != EOF) && (cnt < MAXELE)) {
70            a[cnt] = meas;
71            printf("%f\n",a[cnt]);  /* echo print */
72            sum += a[cnt];          /* accumulate measurements */
73            cnt++;
74            fs = fscanf(fp,"%f",&meas);       /* read next measurement */
75         }
76
77         maxmin (a, &max, &min, cnt);      /* find minimum and maximum
78                                                 measurements */
79
80         /* report count, sum, mean, minimum, and maximum measurements */
81         printf("\ncount = %d",cnt);
82         printf("\nsum = %f",sum);
83         printf("\nmean = %f",mean(sum, cnt));
84         printf("\nminimum = %f",min);
85         printf("\nmaximum = %f\n",max);
86      }
87  }
88
89  /* procedure to find minimum and maximum measurements */
90  maxmin (m, max, min, n)
91  real *m, *max, *min;
92  int n;
93  {
94      int   i;
95
96      /* find minimum and maximum */
97      *min = *m;
98      *max = *m;
99      for (i = 1; i < n; i++) {
100        if (*(m+i) < *min) *min = *(m+i);
```

```
101         if (*(m+i) > *max) *max = *(m+i);
102     }
103 }
104
105 /* compute mean given sum and number of measurements */
106 double mean(sum, n)
107 double sum;
108 int n;
109 {
110     return ((double) (sum/((double) n)));
111 }
```

Apr 30 12:01 1989 sample.data Page 1

6.43
0
-2.34
8.75
-5.25
9
3
-1.75

```
Jun  4 12:08 1989  sample.out Page 1


6.430000
0.000000
-2.340000
8.750000
-5.250000
9.000000
3.000000
-1.750000

count = 8
sum = 17.840000
mean = 2.230000
minimum = -5.250000
maximum = 9.000000
```

```
Jun  4 12:07 1989  detailed.metrics Page 1


Software Metrics:

loc=57
vg=9

Halstead's metrics:
eta1=35
eta2=32

operators:

#include 1
#define 2
double 5
mean 1
( ) 32
; 35
typedef 1
float 1
main 1
{ } 7
[ ] 5
, 26
int 5
FILE 1
* 16
= 12
fopen 1
if 2
== 1
printf 7
if else 1
fscanf 2
& 4
while 1
!= 1
&& 1
< 3
+= 1
++ 2
maxmin 1
for 1
+ 4
> 1
return 1
/ 1

Total # of operators = 187
```

operands:

```
<stdio.h> 1
MAXELE 3
20 1
INFILE 2
"sample.data" 1
real 3
a 6
meas 4
sum 8
min 8
max 8
cnt 10
i 9
fp 5
fs 4
0.0e0 1
0 2
0.0 1
"r" 1
NULL 1
"error: can't open  1
"%f" 2
EOF 1
"%f\n" 1
"\ncount = %d" 1
"\nsum = %f" 1
"\nmean = %f" 1
"\nminimum = %f" 1
"\nmaximum = %f\n" 1
m 8
n 6
1 1
```

Total # of operands = 104

```
Jun  4 12:07 1989  summary.metrics Page 1


UID=sample
LOC=57
VG=9
NVAR=9 NCONST=2 NTYPE=1 NSUB=5
ETA1=35 N1=187 ETA2=32 N2=104
PLEN=291 EPLEN=339.52 PVOL=1765.23 EPPVOL=31.04
EPLEV=0.0176 EPDIF=56.875
EFFORT=100398 TIME=5577.64 LAMBDA=0.55
```

APPENDIX H

DETAILED DESCRIPTIONS OF PROGRAMMING ASSIGNMENTS

Comsc 2123

Programming Assignment #4

Students were required to design and write a Pascal program to sort a list of 40 random numbers using a surrogate array. They were to fill the data array with 40 random numbers and print them using the surrogate array which would be initialized with a list of sequential numbers from 1 to 40. Using the insertion sort method, they were to sort the random numbers by changing the "pointers" of the surrogate array. They were to output the contents of the surrogate and data arrays after the sort and print the data in sorted order by using the surrogate array.

Comsc 2133

Programming Assignment #4

Students were required to design and write a Pascal program to store, maintain, and retrieve an array of records using a linear hashing function. They were to implement five commands (insert, delete, retrieve, print all, and list size) that could process any one of the four main fields of any record. The hashing function used was **HASH := (SUM MOD 19) + 1;**, where **SUM** is the sum of all ordinal values of each character in the key field. The students' programs were required to handle erroneous input gracefully.

Comsc 4323

Phase II

Students were required to modify the design and implementation of their Phase I multiprogramming operating system with variable memory partitions and round robin scheduling. They were to change the CPU scheduling algorithm to a system of multilevel feedback queues. The queues were to be given priorities, and processes were to migrate from higher-priority queues to lower-priority ones based on the number of times a process executed within a queue and the frequency of its I/O. The execution of processes was only simulated. Once the simulation completed, the students were required to write out summary statistics of the processes executed on their simulator.

Comsc 5323

Phase II

Students were required to design and implement a multiprogramming operating system with a specified maximum degree of multiprogramming, variable-sized memory segments, single job stream with spooling of the input and output, and a round robin scheduler. Their programs were to read and execute a stream of batch jobs. The jobs contained specified job control records and programs written in hexadecimal from a limited instruction set. The students were to simulate an architectural machine of their own. Their programs were expected to handle errors in the job control records and in the hexadecimal programs. At the completion of simulation, their programs were to write out statistics of the jobs that were processed. They were also required to produce trace output for each job executed.

APPENDIX I

DYNAMIC MEASUREMENTS

DYNAMIC MEASUREMENTS

COMSC 2123

IDNO	TOTAL SYSTEM TIME (SECS)	# OF SOURCE STMTS. EXEC.	TOTAL JOB TIME (SECS)	COMPILATION TIME (SECS)	EXECUTION TIME (SECS)	OBJECT CODE SIZE (BYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
A	0.690000	5885	0.587700	0.333000	0.254700	10200	10176
	0.690000	5885	0.590700	0.335700	0.255100	10200	10176
B	0.400000	2098	0.305800	0.151800	0.154000	5963	10152
	0.400000	2098	0.304800	0.150600	0.154200	5963	10152
C	0.500000	2687	0.402200	0.185200	0.217000	6323	10176
	0.500000	2687	0.399800	0.184000	0.215800	6323	10176
D	0.450000	2225	0.346800	0.162700	0.184100	5614	10176
	0.450000	2225	0.347800	0.163200	0.184600	5614	10176
E	0.410000	2452	0.308600	0.107400	0.201200	4316	10152
	0.410000	2452	0.307800	0.105800	0.201900	4316	10152
F	0.350000	2284	0.248600	0.110400	0.138200	4177	10080
	0.350000	2284	0.248200	0.110100	0.138100	4177	10080
G	0.400000	2651	0.298500	0.163200	0.135300	6599	10080
	0.400000	2651	0.300600	0.165200	0.135400	6599	10080
H	0.410000	2549	0.313900	0.191200	0.122700	7535	10080
	0.420000	2549	0.315700	0.193100	0.122600	7535	10080
I	0.370000	2081	0.269700	0.127800	0.142000	4277	10080
	0.370000	2081	0.270400	0.127600	0.142800	4277	10080
J	0.370000	2934	0.268600	0.0819000	0.186700	3045	10152
	0.360000	2934	0.268700	0.0811000	0.187600	3045	10152
K	0.410000	2290	0.313700	0.133600	0.180100	4782	10152
	0.410000	2290	0.312100	0.133400	0.178800	4782	10152
L	0.400000	2462	0.304200	0.147300	0.156900	5072	10152
	0.400000	2462	0.303400	0.146900	0.156500	5072	10152

DYNAMIC MEASUREMENTS

COMSC 2133

IDNO	TOTAL SYSTEM TIME (SECS)	# OF SOURCE STMTS. EXEC.	TOTAL JOB TIME (SECS)	COMPILATION TIME (SECS)	EXECUTION TIME (SECS)	OBJECT CODE SIZE (BYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
A	1.43000	14172	1.32500	0.446600	0.878400	16963	10176
	1.42000	14172	1.32050	0.446500	0.874000	16963	10176
B	1.54000	7379	1.44030	0.766300	0.674000	21497	25176
	1.54000	7379	1.44040	0.765900	0.674500	21497	25176
C	0.890000	5491	0.789000	0.393100	0.395900	12736	10176
	0.890000	5491	0.789000	0.394700	0.394200	12736	10176
D	1.40000	8384	1.29800	0.685600	0.612300	22005	10176
	1.40000	8384	1.29670	0.684700	0.611900	22005	10176
E	1.19000	10677	1.08640	0.516800	0.569600	17074	10176
	1.19000	10677	1.08780	0.516500	0.571300	17074	10176
F	1.51000	13549	1.40580	0.505300	0.900500	17345	10176
	1.51000	13549	1.40520	0.505600	0.899600	17345	10176
G	1.24000	4854	1.13530	0.570300	0.565000	17638	10176
	1.24000	4854	1.13490	0.568300	0.566600	17638	10176
H	1.21000	4819	1.11320	0.484500	0.628800	14891	10176
	1.22000	4819	1.12050	0.487700	0.632800	14891	10176
I	0.990000	4883	0.887900	0.442300	0.445600	13881	10176
	0.990000	4883	0.890200	0.444000	0.446200	13881	10176
J	0.580000	4390	0.477200	0.220200	0.257100	6145	13416
	0.580000	4390	0.477200	0.219600	0.257600	6145	13416
K	1.15000	5656	1.04710	0.494800	0.552300	16289	10176
	1.15000	5656	1.04970	0.497000	0.552700	16289	10176
L	1.01000	4312	0.913600	0.412800	0.500900	14298	10176
	1.02000	4312	0.917100	0.416100	0.501000	14298	10176

DYNAMIC MEASUREMENTS

COMSC 4323

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
A	1.50000	1.10000	2.60000	0:00:02	27	60	89088
	1.40000	1.20000	2.60000	0:00:02	27	60	89088
	1.40000	1.20000	2.60000	0:00:02	27	60	89088

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
A	21504	97	0	4	0	0
	21504	96	0	4	0	0
	21504	99	0	4	0	0

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
B	2.70000	0.300000	3.00000	0:00:03	26	98	126976
	2.50000	0.300000	2.80000	0:00:02	26	96	124928
	2.70000	0.300000	3.00000	0:00:03	26	98	126976

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
B	22528	100	0	8	0	0
	22528	98	3	9	0	0
	22528	98	1	8	0	0

DYNAMIC MEASUREMENTS

COMSC 4323

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
C	1.80000	0.200000	2.00000	0:00:02	29	90	121856
	2.00000	0.200000	2.20000	0:00:02	29	90	121856
	2.00000	0.200000	2.20000	0:00:02	29	90	121856

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
C	23552	100	0	2	0	0
	23552	100	0	2	0	0
	23552	96	1	7	0	0

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
D	1.90000	0.200000	2.10000	0:00:02	29	89	120832
	1.60000	0.200000	1.80000	0:00:01	29	88	119808
	1.70000	0.200000	1.90000	0:00:01	29	89	120832

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
D	23552	103	0	5	0	0
	23552	107	0	5	0	0
	23552	106	0	5	0	0

DYNAMIC MEASUREMENTS

COMSC 4323

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
E	3.10000	0.400000	3.50000	0:00:03	27	63	92160
	3.30000	0.300000	3.60000	0:00:03	27	63	92160
	3.20000	0.400000	3.60000	0:00:03	27	63	92160

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
E	21504	98	1	29	0	0
	21504	99	0	24	0	0
	21504	98	1	28	0	0

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
F	0.300000	0.100000	0.400000	0:00:00	26	56	83968
	0.300000	0.100000	0.400000	0:00:00	25	54	80896
	0.300000	0.200000	0.500000	0:00:00	25	53	79872

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
F	21504	110	0	8	0	0
	21504	111	0	3	0	0
	21504	111	0	3	0	0

DYNAMIC MEASUREMENTS

COMSC 4323

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
G	2.70000	0.300000	3.00000	0:00:03	31	107	141312
	2.50000	0.200000	2.70000	0:00:02	31	107	141312
	2.30000	0.200000	2.50000	0:00:02	31	107	141312

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
G	24576	97	1	10	0	0
	24576	101	0	9	0	0
	24576	101	0	4	0	0

DYNAMIC MEASUREMENTS

COMSC 5323 A

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
A	27.4000	3.00000	30.4000	0:00:45	66	134	204800
	26.3000	3.30000	29.6000	0:00:43	66	134	204800
	26.8000	3.00000	29.8000	0:00:43	66	134	204800

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
A	52224	67	5	80	2	0
	52224	67	5	83	2	0
	52224	68	8	86	2	0

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
B	28.2000	2.10000	30.3000	0:00:41	65	123	192512
	28.6000	2.00000	30.6000	0:00:43	65	123	192512
	30.0000	2.00000	32.0000	0:00:45	65	123	192512

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
B	51200	72	4	35	0	0
	51200	71	7	36	0	0
	51200	69	4	39	0	0

DYNAMIC MEASUREMENTS

COMSC 5323 A

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
C	65.9000	150.300	216.200	0:03:48	73	86	162816
	65.2000	150.500	215.700	0:03:40	73	86	162816
	65.7000	155.000	220.700	0:03:50	73	86	162816

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
C	56320	94	6	89	0	0
	56320	97	1	92	0	0
	56320	95	5	99	0	0

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
D	53.0000	35.1000	88.1000	0:01:40	63	188	257024
	51.9000	34.4000	86.3000	0:01:36	63	188	257024
	52.9000	32.9000	85.8000	0:01:39	63	188	257024

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
D	49152	87	8	112	0	0
	49152	89	2	126	0	0
	49152	86	2	122	0	0

DYNAMIC MEASUREMENTS

COMSC 5323 A

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
E	29.9000	1.30000	31.2000	0:00:34	51	171	227328
	29.0000	1.20000	30.2000	0:00:33	51	171	227328
	28.5000	1.20000	29.7000	0:00:33	51	170	226304

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
E	37888	90	2	54	0	0
	37888	90	2	53	0	0
	37888	90	1	52	0	0

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
F	107.200	1.30000	108.500	0:01:48	65	207	278528
	109.100	1.20000	110.300	0:01:50	65	207	278528
	107.200	1.30000	108.500	0:01:49	65	207	278528

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
F	57344	99	0	74	0	0
	57344	99	1	75	0	0
	57344	99	0	78	0	0

DYNAMIC MEASUREMENTS

COMSC 5323 A

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
G	33.6000	2.00000	35.6000	0:00:43	55	119	178176
	34.1000	1.80000	35.9000	0:00:48	55	119	178176
	33.9000	1.80000	35.7000	0:00:48	55	119	178176

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
G	36864	81	2	86	0	0
	36864	74	4	89	0	0
	36864	73	2	86	0	0

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
H	44.9000	2.60000	47.5000	0:01:00	63	203	272384
	48.0000	2.70000	50.7000	0:01:01	63	204	273408
	44.2000	2.60000	46.8000	0:00:57	63	204	273408

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
H	47104	78	1	90	0	0
	47104	83	2	90	0	0
	47104	81	2	96	0	0

DYNAMIC MEASUREMENTS

COMSC 5323 A

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
I	29.4000	1.30000	30.7000	0:00:40	49	243	299008
	34.2000	1.40000	35.6000	0:00:51	49	245	301056
	33.5000	1.40000	34.9000	0:00:50	49	243	299008

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
I	35840	75	6	81	0	0
	35840	69	4	85	0	0
	35840	69	1	81	0	0

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
J	86.8000	0.700000	87.5000	0:01:31	47	134	185344
	89.1000	0.600000	89.7000	0:01:33	47	134	185344
	89.2000	0.800000	90.0000	0:01:34	47	134	185344

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
J	35840	95	6	45	0	0
	35840	96	1	46	0	0
	35840	95	2	47	0	0

DYNAMIC MEASUREMENTS

COMSC 5323 B

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
A	15.0000	3.00000	18.0000	0:00:32	66	128	198656
	15.3000	2.90000	18.2000	0:00:33	66	129	199680
	15.1000	2.80000	17.9000	0:00:32	66	128	198656

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
A	52224	55	6	71	2	0
	52224	54	6	74	2	0
	52224	55	6	70	2	0

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
B	12.7000	1.70000	14.4000	0:00:26	64	120	188416
	13.0000	1.50000	14.5000	0:00:26	65	120	189440
	13.3000	1.60000	14.9000	0:00:27	64	120	188416

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
B	51200	53	11	31	0	0
	51200	55	1	25	0	0
	51200	53	2	26	0	0

DYNAMIC MEASUREMENTS

COMSC 5323 B

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
C	37.1000	81.4000	118.500	0:02:05	73	82	158720
	36.8000	82.2000	119.000	0:02:08	73	82	158720
	36.6000	83.3000	119.900	0:02:04	73	82	158720

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
C	56320	94	9	69	0	0
	56320	92	4	65	0	0
	56320	96	2	68	0	0

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
D	31.1000	28.5000	59.6000	0:01:10	63	189	258048
	32.0000	27.9000	59.9000	0:01:12	63	189	258048
	31.7000	27.5000	59.2000	0:01:08	63	190	259072

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
D	49152	84	2	101	0	0
	49152	82	0	103	0	0
	49152	86	1	106	0	0

DYNAMIC MEASUREMENTS

COMSC 5323 B

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
E	17.4000	1.00000	18.4000	0:00:22	51	159	215040
	17.3000	0.90000	18.2000	0:00:23	51	160	216064
	17.4000	0.90000	18.3000	0:00:22	51	160	216064

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
E	37888	82	3	37	0	0
	37888	78	1	38	0	0
	37888	80	0	35	0	0

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
F	15.2000	1.40000	16.6000	0:00:16	65	183	253952
	16.7000	1.30000	18.0000	0:00:18	65	184	254976
	16.5000	1.50000	18.0000	0:00:18	65	182	252928

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
F	57344	98	1	72	0	0
	57344	98	0	71	0	0
	57344	98	1	73	0	0

DYNAMIC MEASUREMENTS

COMSC 5323 B

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
G	15.5000	1.60000	17.1000	0:00:25	55	104	162816
	17.7000	1.50000	19.2000	0:00:28	55	105	163840
	16.7000	1.40000	18.1000	0:00:27	55	104	162816

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
G	36864	67	2	58	0	0
	36864	68	1	59	0	0
	36864	65	0	59	0	0

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
H	26.9000	2.50000	29.4000	0:00:43	62	206	274432
	26.0000	2.50000	28.5000	0:00:39	62	205	273408
	26.5000	2.60000	29.1000	0:00:41	62	205	273408

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
H	47104	68	4	88	0	0
	47104	71	2	88	0	0
	47104	70	3	92	0	0

DYNAMIC MEASUREMENTS

COMSC 5323 B

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
I	17.7000	1.10000	18.8000	0:00:29	49	190	244736
	18.2000	1.10000	19.3000	0:00:28	49	190	244736
	17.4000	1.00000	18.4000	0:00:30	49	191	245760

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
I	35840	64	2	53	0	0
	35840	67	1	53	0	0
	35840	59	1	54	0	0

IDNO	USER TIME (SECS)	SYSTEM TIME (SECS)	EXECUTION TIME (SECS)	ELAPSED TIME (H:MM:SS)	PROGRAM MEMORY REQ'D. (KILOBYTES)	DATA MEMORY REQ'D. (KILOBYTES)	MEMORY REQUESTED DURING EXEC. (BYTES)
J	3.50000	0.400000	3.90000	0:00:09	47	103	153600
	3.70000	0.400000	4.10000	0:00:08	47	102	152576
	3.70000	0.400000	4.10000	0:00:07	47	103	153600

IDNO	OBJECT CODE SIZE (BYTES)	% CPU UTILIZATION	# OF DISK READS	# OF DISK WRITES	# OF PAGE FAULTS	# OF TIMES PROCESS SWAPPED
J	35840	44	3	21	0	0
	35840	50	0	19	0	0
	35840	53	0	19	0	0

APPENDIX J

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 2123

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=A								
TSYSTIME	2	0.69000	0.69000	0.69000	0.00000	0.00000	.	.
NSTMTS	2	5885.00000	5885.00000	5885.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.58920	0.58770	0.59070	0.00212	0.00150	392.80	0.0016
COMPTIME	2	0.33435	0.33300	0.33570	0.00191	0.00135	247.67	0.0026
EXECTIME	2	0.25490	0.25470	0.25510	0.00028	0.00020	1274.50	0.0005
OBJSIZE	2	10200.00000	10200.00000	10200.00000	0.00000	0.00000	.	.
EXECMEM	2	10176.00000	10176.00000	10176.00000	0.00000	0.00000	.	.
IDNO=B								
TSYSTIME	2	0.40000	0.40000	0.40000	0.00000	0.00000	.	.
NSTMTS	2	2098.00000	2098.00000	2098.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.30530	0.30480	0.30580	0.00071	0.00050	610.60	0.0010
COMPTIME	2	0.15120	0.15060	0.15180	0.00085	0.00060	252.00	0.0025
EXECTIME	2	0.15410	0.15400	0.15420	0.00014	0.00010	1541.00	0.0004
OBJSIZE	2	5963.00000	5963.00000	5963.00000	0.00000	0.00000	.	.
EXECMEM	2	10152.00000	10152.00000	10152.00000	0.00000	0.00000	.	.
IDNO=C								
TSYSTIME	2	0.50000	0.50000	0.50000	0.00000	0.00000	.	.
NSTMTS	2	2687.00000	2687.00000	2687.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.40100	0.39980	0.40220	0.00170	0.00120	334.17	0.0019
COMPTIME	2	0.18460	0.18400	0.18520	0.00085	0.00060	307.67	0.0021
EXECTIME	2	0.21640	0.21580	0.21700	0.00085	0.00060	360.67	0.0018
OBJSIZE	2	6323.00000	6323.00000	6323.00000	0.00000	0.00000	.	.
EXECMEM	2	10176.00000	10176.00000	10176.00000	0.00000	0.00000	.	.
IDNO=D								
TSYSTIME	2	0.45000	0.45000	0.45000	0.00000	0.00000	.	.
NSTMTS	2	2225.00000	2225.00000	2225.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.34730	0.34680	0.34780	0.00071	0.00050	694.60	0.0009
COMPTIME	2	0.16295	0.16270	0.16320	0.00035	0.00025	651.80	0.0010
EXECTIME	2	0.18435	0.18410	0.18460	0.00035	0.00025	737.40	0.0009
OBJSIZE	2	5614.00000	5614.00000	5614.00000	0.00000	0.00000	.	.
EXECMEM	2	10176.00000	10176.00000	10176.00000	0.00000	0.00000	.	.

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 2123

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=E								
TSYSTIME	2	0.41000	0.41000	0.41000	0.00000	0.00000	.	.
NSTMTS	2	2452.00000	2452.00000	2452.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.30820	0.30780	0.30860	0.00057	0.00040	770.50	0.0008
COMPTIME	2	0.10660	0.10580	0.10740	0.00113	0.00080	133.25	0.0048
EXECTIME	2	0.20155	0.20120	0.20190	0.00049	0.00035	575.86	0.0011
OBJSIZE	2	4316.00000	4316.00000	4316.00000	0.00000	0.00000	.	.
EXECMEM	2	10152.00000	10152.00000	10152.00000	0.00000	0.00000	.	.
IDNO=F								
TSYSTIME	2	0.35000	0.35000	0.35000	0.00000	0.00000	.	.
NSTMTS	2	2284.00000	2284.00000	2284.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.24840	0.24820	0.24860	0.00028	0.00020	1242.00	0.0005
COMPTIME	2	0.11025	0.11010	0.11040	0.00021	0.00015	735.00	0.0009
EXECTIME	2	0.13815	0.13810	0.13820	0.00007	0.00005	2763.00	0.0002
OBJSIZE	2	4177.00000	4177.00000	4177.00000	0.00000	0.00000	.	.
EXECMEM	2	10080.00000	10080.00000	10080.00000	0.00000	0.00000	.	.
IDNO=G								
TSYSTIME	2	0.40000	0.40000	0.40000	0.00000	0.00000	.	.
NSTMTS	2	2651.00000	2651.00000	2651.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.29955	0.29850	0.30060	0.00148	0.00105	285.29	0.0022
COMPTIME	2	0.16420	0.16320	0.16520	0.00141	0.00100	164.20	0.0039
EXECTIME	2	0.13535	0.13530	0.13540	0.00007	0.00005	2707.00	0.0002
OBJSIZE	2	6599.00000	6599.00000	6599.00000	0.00000	0.00000	.	.
EXECMEM	2	10080.00000	10080.00000	10080.00000	0.00000	0.00000	.	.
IDNO=H								
TSYSTIME	2	0.41500	0.41000	0.42000	0.00707	0.00500	83.00	0.0077
NSTMTS	2	2549.00000	2549.00000	2549.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.31480	0.31390	0.31570	0.00127	0.00090	349.78	0.0018
COMPTIME	2	0.19215	0.19120	0.19310	0.00134	0.00095	202.26	0.0031
EXECTIME	2	0.12265	0.12260	0.12270	0.00007	0.00005	2453.00	0.0003
OBJSIZE	2	7535.00000	7535.00000	7535.00000	0.00000	0.00000	.	.
EXECMEM	2	10080.00000	10080.00000	10080.00000	0.00000	0.00000	.	.

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 2123

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=I								
TSYSTIME	2	0.37000	0.37000	0.37000	0.00000	0.00000	.	.
NSTMTS	2	2081.00000	2081.00000	2081.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.27005	0.26970	0.27040	0.00049	0.00035	771.57	0.0008
COMPTIME	2	0.12770	0.12760	0.12780	0.00014	0.00010	1277.00	0.0005
EXECTIME	2	0.14240	0.14200	0.14280	0.00057	0.00040	356.00	0.0018
OBJSIZE	2	4277.00000	4277.00000	4277.00000	0.00000	0.00000	.	.
EXECMEM	2	10080.00000	10080.00000	10080.00000	0.00000	0.00000	.	.
IDNO=J								
TSYSTIME	2	0.36500	0.36000	0.37000	0.00707	0.00500	73.00	0.0087
NSTMTS	2	2934.00000	2934.00000	2934.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.26865	0.26860	0.26870	0.00007	0.00005	5373.00	0.0001
COMPTIME	2	0.08150	0.08110	0.08190	0.00057	0.00040	203.75	0.0031
EXECTIME	2	0.18715	0.18670	0.18760	0.00064	0.00045	415.89	0.0015
OBJSIZE	2	3045.00000	3045.00000	3045.00000	0.00000	0.00000	.	.
EXECMEM	2	10152.00000	10152.00000	10152.00000	0.00000	0.00000	.	.
IDNO=K								
TSYSTIME	2	0.41000	0.41000	0.41000	0.00000	0.00000	.	.
NSTMTS	2	2290.00000	2290.00000	2290.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.31290	0.31210	0.31370	0.00113	0.00080	391.13	0.0016
COMPTIME	2	0.13350	0.13340	0.13360	0.00014	0.00010	1335.00	0.0005
EXECTIME	2	0.17945	0.17880	0.18010	0.00092	0.00065	276.08	0.0023
OBJSIZE	2	4782.00000	4782.00000	4782.00000	0.00000	0.00000	.	.
EXECMEM	2	10152.00000	10152.00000	10152.00000	0.00000	0.00000	.	.
IDNO=L								
TSYSTIME	2	0.40000	0.40000	0.40000	0.00000	0.00000	.	.
NSTMTS	2	2462.00000	2462.00000	2462.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.30380	0.30340	0.30420	0.00057	0.00040	759.50	0.0008
COMPTIME	2	0.14710	0.14690	0.14730	0.00028	0.00020	735.50	0.0009
EXECTIME	2	0.15670	0.15650	0.15690	0.00028	0.00020	783.50	0.0008
OBJSIZE	2	5072.00000	5072.00000	5072.00000	0.00000	0.00000	.	.
EXECMEM	2	10152.00000	10152.00000	10152.00000	0.00000	0.00000	.	.

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 2133

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=A								
TSYSTIME	2	1.42500	1.42000	1.43000	0.00707	0.00500	285.00	0.0022
NSTMTS	2	14172.00000	14172.00000	14172.00000	0.00000	0.00000	.	.
TJOBTIME	2	1.32275	1.32050	1.32500	0.00318	0.00225	587.89	0.0011
COMPTIME	2	0.44655	0.44650	0.44660	0.00007	0.00005	8931.00	0.0001
EXECTIME	2	0.87620	0.87400	0.87840	0.00311	0.00220	398.27	0.0016
OBJSIZE	2	16963.00000	16963.00000	16963.00000	0.00000	0.00000	.	.
EXECMEM	2	10176.00000	10176.00000	10176.00000	0.00000	0.00000	.	.
IDNO=B								
TSYSTIME	2	1.54000	1.54000	1.54000	0.00000	0.00000	.	.
NSTMTS	2	7379.00000	7379.00000	7379.00000	0.00000	0.00000	.	.
TJOBTIME	2	1.44035	1.44030	1.44040	0.00007	0.00005	28807.00	0.0001
COMPTIME	2	0.76610	0.76590	0.76630	0.00028	0.00020	3830.50	0.0002
EXECTIME	2	0.67425	0.67400	0.67450	0.00035	0.00025	2697.00	0.0002
OBJSIZE	2	21497.00000	21497.00000	21497.00000	0.00000	0.00000	.	.
EXECMEM	2	25176.00000	25176.00000	25176.00000	0.00000	0.00000	.	.
IDNO=C								
TSYSTIME	2	0.89000	0.89000	0.89000	0.00000	0.00000	.	.
NSTMTS	2	5491.00000	5491.00000	5491.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.78900	0.78900	0.78900	0.00000	0.00000	.	.
COMPTIME	2	0.39390	0.39310	0.39470	0.00113	0.00080	492.37	0.0013
EXECTIME	2	0.39505	0.39420	0.39590	0.00120	0.00085	464.76	0.0014
OBJSIZE	2	12736.00000	12736.00000	12736.00000	0.00000	0.00000	.	.
EXECMEM	2	10176.00000	10176.00000	10176.00000	0.00000	0.00000	.	.
IDNO=D								
TSYSTIME	2	1.40000	1.40000	1.40000	0.00000	0.00000	.	.
NSTMTS	2	8384.00000	8384.00000	8384.00000	0.00000	0.00000	.	.
TJOBTIME	2	1.29735	1.29670	1.29800	0.00092	0.00065	1995.92	0.0003
COMPTIME	2	0.68515	0.68470	0.68560	0.00064	0.00045	1522.56	0.0004
EXECTIME	2	0.61210	0.61190	0.61230	0.00028	0.00020	3060.50	0.0002
OBJSIZE	2	22005.00000	22005.00000	22005.00000	0.00000	0.00000	.	.
EXECMEM	2	10176.00000	10176.00000	10176.00000	0.00000	0.00000	.	.

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 2133

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=E								
TSYSTIME	2	1.19000	1.19000	1.19000	0.00000	0.00000	.	.
NSTMTS	2	10677.00000	10677.00000	10677.00000	0.00000	0.00000	.	.
TJOBTIME	2	1.08710	1.08640	1.08780	0.00099	0.00070	1553.00	0.0004
COMPTIME	2	0.51665	0.51650	0.51680	0.00021	0.00015	3444.33	0.0002
EXECTIME	2	0.57045	0.56960	0.57130	0.00120	0.00085	671.12	0.0009
OBJSIZE	2	17074.00000	17074.00000	17074.00000	0.00000	0.00000	.	.
EXECMEM	2	10176.00000	10176.00000	10176.00000	0.00000	0.00000	.	.
IDNO=F								
TSYSTIME	2	1.51000	1.51000	1.51000	0.00000	0.00000	.	.
NSTMTS	2	13549.00000	13549.00000	13549.00000	0.00000	0.00000	.	.
TJOBTIME	2	1.40550	1.40520	1.40580	0.00042	0.00030	4685.00	0.0001
COMPTIME	2	0.50545	0.50530	0.50560	0.00021	0.00015	3369.67	0.0002
EXECTIME	2	0.90005	0.89960	0.90050	0.00064	0.00045	2000.11	0.0003
OBJSIZE	2	17345.00000	17345.00000	17345.00000	0.00000	0.00000	.	.
EXECMEM	2	10176.00000	10176.00000	10176.00000	0.00000	0.00000	.	.
IDNO=G								
TSYSTIME	2	1.24000	1.24000	1.24000	0.00000	0.00000	.	.
NSTMTS	2	4854.00000	4854.00000	4854.00000	0.00000	0.00000	.	.
TJOBTIME	2	1.13510	1.13490	1.13530	0.00028	0.00020	5675.50	0.0001
COMPTIME	2	0.56930	0.56830	0.57030	0.00141	0.00100	569.30	0.0011
EXECTIME	2	0.56580	0.56500	0.56660	0.00113	0.00080	707.25	0.0009
OBJSIZE	2	17638.00000	17638.00000	17638.00000	0.00000	0.00000	.	.
EXECMEM	2	10176.00000	10176.00000	10176.00000	0.00000	0.00000	.	.
IDNO=H								
TSYSTIME	2	1.21500	1.21000	1.22000	0.00707	0.00500	243.00	0.0026
NSTMTS	2	4819.00000	4819.00000	4819.00000	0.00000	0.00000	.	.
TJOBTIME	2	1.11685	1.11320	1.12050	0.00516	0.00365	305.99	0.0021
COMPTIME	2	0.48610	0.48450	0.48770	0.00226	0.00160	303.81	0.0021
EXECTIME	2	0.63080	0.62880	0.63280	0.00283	0.00200	315.40	0.0020
OBJSIZE	2	14891.00000	14891.00000	14891.00000	0.00000	0.00000	.	.
EXECMEM	2	10176.00000	10176.00000	10176.00000	0.00000	0.00000	.	.

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 2133

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=I								
TSYSTIME	2	0.99000	0.99000	0.99000	0.00000	0.00000	.	.
NSTMTS	2	4883.00000	4883.00000	4883.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.88905	0.88790	0.89020	0.00163	0.00115	773.09	0.0008
COMPTIME	2	0.44315	0.44230	0.44400	0.00120	0.00085	521.35	0.0012
EXECTIME	2	0.44590	0.44560	0.44620	0.00042	0.00030	1486.33	0.0004
OBJSIZE	2	13881.00000	13881.00000	13881.00000	0.00000	0.00000	.	.
EXECMEM	2	10176.00000	10176.00000	10176.00000	0.00000	0.00000	.	.
IDNO=J								
TSYSTIME	2	0.58000	0.58000	0.58000	0.00000	0.00000	.	.
NSTMTS	2	4390.00000	4390.00000	4390.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.47720	0.47720	0.47720	0.00000	0.00000	.	.
COMPTIME	2	0.21990	0.21960	0.22020	0.00042	0.00030	733.00	0.0009
EXECTIME	2	0.25735	0.25710	0.25760	0.00035	0.00025	1029.40	0.0006
OBJSIZE	2	6145.00000	6145.00000	6145.00000	0.00000	0.00000	.	.
EXECMEM	2	13416.00000	13416.00000	13416.00000	0.00000	0.00000	.	.
IDNO=K								
TSYSTIME	2	1.15000	1.15000	1.15000	0.00000	0.00000	.	.
NSTMTS	2	5656.00000	5656.00000	5656.00000	0.00000	0.00000	.	.
TJOBTIME	2	1.04840	1.04710	1.04970	0.00184	0.00130	806.46	0.0008
COMPTIME	2	0.49590	0.49480	0.49700	0.00156	0.00110	450.82	0.0014
EXECTIME	2	0.55250	0.55230	0.55270	0.00028	0.00020	2762.50	0.0002
OBJSIZE	2	16289.00000	16289.00000	16289.00000	0.00000	0.00000	.	.
EXECMEM	2	10176.00000	10176.00000	10176.00000	0.00000	0.00000	.	.
IDNO=L								
TSYSTIME	2	1.01500	1.01000	1.02000	0.00707	0.00500	203.00	0.0031
NSTMTS	2	4312.00000	4312.00000	4312.00000	0.00000	0.00000	.	.
TJOBTIME	2	0.91535	0.91360	0.91710	0.00247	0.00175	523.06	0.0012
COMPTIME	2	0.41445	0.41280	0.41610	0.00233	0.00165	251.18	0.0025
EXECTIME	2	0.50095	0.50090	0.50100	0.00007	0.00005	10019.00	0.0001
OBJSIZE	2	14298.00000	14298.00000	14298.00000	0.00000	0.00000	.	.
EXECMEM	2	10176.00000	10176.00000	10176.00000	0.00000	0.00000	.	.

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 4323

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=A								
USERTIME	3	1.43333	1.40000	1.50000	0.05774	0.03333	43.00	0.0005
SYSTIME	3	1.16667	1.10000	1.20000	0.05774	0.03333	35.00	0.0008
ELAPSED	3	2.00000	2.00000	2.00000	0.00000	0.00000	.	.
PCPU	3	97.33333	96.00000	99.00000	1.52753	0.88192	110.37	0.0001
MEMPROG	3	27.00000	27.00000	27.00000	0.00000	0.00000	.	.
MEMDATA	3	60.00000	60.00000	60.00000	0.00000	0.00000	.	.
NREAD	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NWRITE	3	4.00000	4.00000	4.00000	0.00000	0.00000	.	.
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	21504.00000	21504.00000	21504.00000	0.00000	0.00000	.	.
EXECTIME	3	2.60000	2.60000	2.60000	0.00000	0.00000	.	.
EXECMEM	3	89088.00000	89088.00000	89088.00000	0.00000	0.00000	.	.
IDNO=B								
USERTIME	3	2.63333	2.50000	2.70000	0.11547	0.06667	39.50	0.0006
SYSTIME	3	0.30000	0.30000	0.30000	0.00000	0.00000	.	.
ELAPSED	3	2.66667	2.00000	3.00000	0.57735	0.33333	8.00	0.0153
PCPU	3	98.66667	98.00000	100.00000	1.15470	0.66667	148.00	0.0001
MEMPROG	3	26.00000	26.00000	26.00000	0.00000	0.00000	.	.
MEMDATA	3	97.33333	96.00000	98.00000	1.15470	0.66667	146.00	0.0001
NREAD	3	1.33333	0.00000	3.00000	1.52753	0.88192	1.51	0.2697
NWRITE	3	8.33333	8.00000	9.00000	0.57735	0.33333	25.00	0.0016
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	22528.00000	22528.00000	22528.00000	0.00000	0.00000	.	.
EXECTIME	3	2.93333	2.80000	3.00000	0.11547	0.06667	44.00	0.0005
EXECMEM	3	126293.33333	124928.00000	126976.00000	1182.41335	682.66667	185.00	0.0001

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 4323

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=C								
USERTIME	3	1.93333	1.80000	2.00000	0.11547	0.06667	29.00	0.0012
SYSTIME	3	0.20000	0.20000	0.20000	0.00000	0.00000	.	.
ELAPSED	3	2.00000	2.00000	2.00000	0.00000	0.00000	.	.
PCPU	3	98.66667	96.00000	100.00000	2.30940	1.33333	74.00	0.0002
MEMPROG	3	29.00000	29.00000	29.00000	0.00000	0.00000	.	.
MEMDATA	3	90.00000	90.00000	90.00000	0.00000	0.00000	.	.
NREAD	3	0.33333	0.00000	1.00000	0.57735	0.33333	1.00	0.4226
NWRITE	3	3.66667	2.00000	7.00000	2.88675	1.66667	2.20	0.1588
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	23552.00000	23552.00000	23552.00000	0.00000	0.00000	.	.
EXECTIME	3	2.13333	2.00000	2.20000	0.11547	0.06667	32.00	0.0010
EXECMEM	3	121856.00000	121856.00000	121856.00000	0.00000	0.00000	.	.
IDNO=D								
USERTIME	3	1.73333	1.60000	1.90000	0.15275	0.08819	19.65	0.0026
SYSTIME	3	0.20000	0.20000	0.20000	0.00000	0.00000	.	.
ELAPSED	3	1.33333	1.00000	2.00000	0.57735	0.33333	4.00	0.0572
PCPU	3	105.33333	103.00000	107.00000	2.08167	1.20185	87.64	0.0001
MEMPROG	3	29.00000	29.00000	29.00000	0.00000	0.00000	.	.
MEMDATA	3	88.66667	88.00000	89.00000	0.57735	0.33333	266.00	0.0001
NREAD	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NWRITE	3	5.00000	5.00000	5.00000	0.00000	0.00000	.	.
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	23552.00000	23552.00000	23552.00000	0.00000	0.00000	.	.
EXECTIME	3	1.93333	1.80000	2.10000	0.15275	0.08819	21.92	0.0021
EXECMEM	3	120490.66667	119808.00000	120832.00000	591.20668	341.33333	353.00	0.0001

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 4323

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=E								
USERTIME	3	3.20000	3.10000	3.30000	0.10000	0.05774	55.43	0.0003
SYSTIME	3	0.36667	0.30000	0.40000	0.05774	0.03333	11.00	0.0082
ELAPSED	3	3.00000	3.00000	3.00000	0.00000	0.00000	.	.
PCPU	3	98.33333	98.00000	99.00000	0.57735	0.33333	295.00	0.0001
MEMPROG	3	27.00000	27.00000	27.00000	0.00000	0.00000	.	.
MEMDATA	3	63.00000	63.00000	63.00000	0.00000	0.00000	.	.
NREAD	3	0.66667	0.00000	1.00000	0.57735	0.33333	2.00	0.1835
NWRITE	3	27.00000	24.00000	29.00000	2.64575	1.52753	17.68	0.0032
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	21504.00000	21504.00000	21504.00000	0.00000	0.00000	.	.
EXECTIME	3	3.56667	3.50000	3.60000	0.05774	0.03333	107.00	0.0001
EXECMEM	3	92160.00000	92160.00000	92160.00000	0.00000	0.00000	.	.
IDNO=F								
USERTIME	3	0.30000	0.30000	0.30000	0.00000	0.00000	.	.
SYSTIME	3	0.13333	0.10000	0.20000	0.05774	0.03333	4.00	0.0572
ELAPSED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
PCPU	3	110.66667	110.00000	111.00000	0.57735	0.33333	332.00	0.0001
MEMPROG	3	25.33333	25.00000	26.00000	0.57735	0.33333	76.00	0.0002
MEMDATA	3	54.33333	53.00000	56.00000	1.52753	0.88192	61.61	0.0003
NREAD	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NWRITE	3	4.66667	3.00000	8.00000	2.88675	1.66667	2.80	0.1074
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	21504.00000	21504.00000	21504.00000	0.00000	0.00000	.	.
EXECTIME	3	0.43333	0.40000	0.50000	0.05774	0.03333	13.00	0.0059
EXECMEM	3	81578.66667	79872.00000	83968.00000	2131.62598	1230.69484	66.29	0.0002

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 4323

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=G								
USERTIME	3	2.50000	2.30000	2.70000	0.20000	0.11547	21.65	0.0021
SYSTIME	3	0.23333	0.20000	0.30000	0.05774	0.03333	7.00	0.0198
ELAPSED	3	2.33333	2.00000	3.00000	0.57735	0.33333	7.00	0.0198
PCPU	3	99.66667	97.00000	101.00000	2.30940	1.33333	74.75	0.0002
MEMPROG	3	31.00000	31.00000	31.00000	0.00000	0.00000	.	.
MEMDATA	3	107.00000	107.00000	107.00000	0.00000	0.00000	.	.
NREAD	3	0.33333	0.00000	1.00000	0.57735	0.33333	1.00	0.4226
NWRITE	3	7.66667	4.00000	10.00000	3.21455	1.85592	4.13	0.0539
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	24576.00000	24576.00000	24576.00000	0.00000	0.00000	.	.
EXECTIME	3	2.73333	2.50000	3.00000	0.25166	0.14530	18.81	0.0028
EXECMEM	3	141312.00000	141312.00000	141312.00000	0.00000	0.00000	.	.

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 5323 A

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=A								
USERTIME	3	26.83333	26.30000	27.40000	0.55076	0.31798	84.39	0.0001
SYSTIME	3	3.10000	3.00000	3.30000	0.17321	0.10000	31.00	0.0010
ELAPSED	3	43.66667	43.00000	45.00000	1.15470	0.66667	65.50	0.0002
PCPU	3	67.33333	67.00000	68.00000	0.57735	0.33333	202.00	0.0001
MEMPROG	3	66.00000	66.00000	66.00000	0.00000	0.00000	.	.
MEMDATA	3	134.00000	134.00000	134.00000	0.00000	0.00000	.	.
NREAD	3	6.00000	5.00000	8.00000	1.73205	1.00000	6.00	0.0267
NWRITE	3	83.00000	80.00000	86.00000	3.00000	1.73205	47.92	0.0004
NPGFLT	3	2.00000	2.00000	2.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	52224.00000	52224.00000	52224.00000	0.00000	0.00000	.	.
EXECTIME	3	29.93333	29.60000	30.40000	0.41633	0.24037	124.53	0.0001
EXECMEM	3	204800.00000	204800.00000	204800.00000	0.00000	0.00000	.	.
IDNO=B								
USERTIME	3	28.93333	28.20000	30.00000	0.94516	0.54569	53.02	0.0004
SYSTIME	3	2.03333	2.00000	2.10000	0.05774	0.03333	61.00	0.0003
ELAPSED	3	43.00000	41.00000	45.00000	2.00000	1.15470	37.24	0.0007
PCPU	3	70.66667	69.00000	72.00000	1.52753	0.88192	80.13	0.0002
MEMPROG	3	65.00000	65.00000	65.00000	0.00000	0.00000	.	.
MEMDATA	3	123.00000	123.00000	123.00000	0.00000	0.00000	.	.
NREAD	3	5.00000	4.00000	7.00000	1.73205	1.00000	5.00	0.0377
NWRITE	3	36.66667	35.00000	39.00000	2.08167	1.20185	30.51	0.0011
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	51200.00000	51200.00000	51200.00000	0.00000	0.00000	.	.
EXECTIME	3	30.96667	30.30000	32.00000	0.90738	0.52387	59.11	0.0003
EXECMEM	3	192512.00000	192512.00000	192512.00000	0.00000	0.00000	.	.

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 5323 A

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=C								
USERTIME	3	65.60000	65.20000	65.90000	0.36056	0.20817	315.13	0.0001
SYSTIME	3	151.93333	150.30000	155.00000	2.65769	1.53442	99.02	0.0001
ELAPSED	3	226.00000	220.00000	230.00000	5.29150	3.05505	73.98	0.0002
PCPU	3	95.33333	94.00000	97.00000	1.52753	0.88192	108.10	0.0001
MEMPROG	3	73.00000	73.00000	73.00000	0.00000	0.00000	.	.
MEMDATA	3	86.00000	86.00000	86.00000	0.00000	0.00000	.	.
NREAD	3	4.00000	1.00000	6.00000	2.64575	1.52753	2.62	0.1201
NWRITE	3	93.33333	89.00000	99.00000	5.13160	2.96273	31.50	0.0010
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	56320.00000	56320.00000	56320.00000	0.00000	0.00000	.	.
EXECTIME	3	217.53333	215.70000	220.70000	2.75379	1.58990	136.82	0.0001
EXECMEM	3	162816.00000	162816.00000	162816.00000	0.00000	0.00000	.	.
IDNO=D								
USERTIME	3	52.60000	51.90000	53.00000	0.60828	0.35119	149.78	0.0001
SYSTIME	3	34.13333	32.90000	35.10000	1.12398	0.64893	52.60	0.0004
ELAPSED	3	98.33333	96.00000	100.00000	2.08167	1.20185	81.82	0.0001
PCPU	3	87.33333	86.00000	89.00000	1.52753	0.88192	99.03	0.0001
MEMPROG	3	63.00000	63.00000	63.00000	0.00000	0.00000	.	.
MEMDATA	3	188.00000	188.00000	188.00000	0.00000	0.00000	.	.
NREAD	3	4.00000	2.00000	8.00000	3.46410	2.00000	2.00	0.1835
NWRITE	3	120.00000	112.00000	126.00000	7.21110	4.16333	28.82	0.0012
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	49152.00000	49152.00000	49152.00000	0.00000	0.00000	.	.
EXECTIME	3	86.73333	85.80000	88.10000	1.20968	0.69841	124.19	0.0001
EXECMEM	3	257024.00000	257024.00000	257024.00000	0.00000	0.00000	.	.

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 5323 A

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=E								
USERTIME	3	29.13333	28.50000	29.90000	0.70946	0.40961	71.13	0.0002
SYSTIME	3	1.23333	1.20000	1.30000	0.05774	0.03333	37.00	0.0007
ELAPSED	3	33.33333	33.00000	34.00000	0.57735	0.33333	100.00	0.0001
PCPU	3	90.00000	90.00000	90.00000	0.00000	0.00000	.	.
MEMPROG	3	51.00000	51.00000	51.00000	0.00000	0.00000	.	.
MEMDATA	3	170.66667	170.00000	171.00000	0.57735	0.33333	512.00	0.0001
NREAD	3	1.66667	1.00000	2.00000	0.57735	0.33333	5.00	0.0377
NWRITE	3	53.00000	52.00000	54.00000	1.00000	0.57735	91.80	0.0001
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	37888.00000	37888.00000	37888.00000	0.00000	0.00000	.	.
EXECTIME	3	30.36667	29.70000	31.20000	0.76376	0.44096	68.87	0.0002
EXECMEM	3	226986.66667	226304.00000	227328.00000	591.20668	341.33333	665.00	0.0001
IDNO=F								
USERTIME	3	107.83333	107.20000	109.10000	1.09697	0.63333	170.26	0.0001
SYSTIME	3	1.26667	1.20000	1.30000	0.05774	0.03333	38.00	0.0007
ELAPSED	3	109.00000	108.00000	110.00000	1.00000	0.57735	188.79	0.0001
PCPU	3	99.00000	99.00000	99.00000	0.00000	0.00000	.	.
MEMPROG	3	65.00000	65.00000	65.00000	0.00000	0.00000	.	.
MEMDATA	3	207.00000	207.00000	207.00000	0.00000	0.00000	.	.
NREAD	3	0.33333	0.00000	1.00000	0.57735	0.33333	1.00	0.4226
NWRITE	3	75.66667	74.00000	78.00000	2.08167	1.20185	62.96	0.0003
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	57344.00000	57344.00000	57344.00000	0.00000	0.00000	.	.
EXECTIME	3	109.10000	108.50000	110.30000	1.03923	0.60000	181.83	0.0001
EXECMEM	3	278528.00000	278528.00000	278528.00000	0.00000	0.00000	.	.

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 5323 A

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
				IDNO=G				
USERTIME	3	33.86667	33.60000	34.10000	0.25166	0.14530	233.09	0.0001
SYSTIME	3	1.86667	1.80000	2.00000	0.11547	0.06667	28.00	0.0013
ELAPSED	3	46.33333	43.00000	48.00000	2.88675	1.66667	27.80	0.0013
PCPU	3	76.00000	73.00000	81.00000	4.35890	2.51661	30.20	0.0011
MEMPROG	3	55.00000	55.00000	55.00000	0.00000	0.00000	.	.
MEMDATA	3	119.00000	119.00000	119.00000	0.00000	0.00000	.	.
NREAD	3	2.66667	2.00000	4.00000	1.15470	0.66667	4.00	0.0572
NWRITE	3	87.00000	86.00000	89.00000	1.73205	1.00000	87.00	0.0001
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	36864.00000	36864.00000	36864.00000	0.00000	0.00000	.	.
EXECTIME	3	35.73333	35.60000	35.90000	0.15275	0.08819	405.18	0.0001
EXECMEM	3	178176.00000	178176.00000	178176.00000	0.00000	0.00000	.	.
				IDNO=H				
USERTIME	3	45.70000	44.20000	48.00000	2.02237	1.16762	39.14	0.0007
SYSTIME	3	2.63333	2.60000	2.70000	0.05774	0.03333	79.00	0.0002
ELAPSED	3	59.33333	57.00000	61.00000	2.08167	1.20185	49.37	0.0004
PCPU	3	80.66667	78.00000	83.00000	2.51661	1.45297	55.52	0.0003
MEMPROG	3	63.00000	63.00000	63.00000	0.00000	0.00000	.	.
MEMDATA	3	203.66667	203.00000	204.00000	0.57735	0.33333	611.00	0.0001
NREAD	3	1.66667	1.00000	2.00000	0.57735	0.33333	5.00	0.0377
NWRITE	3	92.00000	90.00000	96.00000	3.46410	2.00000	46.00	0.0005
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	47104.00000	47104.00000	47104.00000	0.00000	0.00000	.	.
EXECTIME	3	48.33333	46.80000	50.70000	2.07926	1.20046	40.26	0.0006
EXECMEM	3	273066.66667	272384.00000	273408.00000	591.20668	341.33333	800.00	0.0001

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 5323 A

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=I								
USERTIME	3	32.36667	29.40000	34.20000	2.59294	1.49703	21.62	0.0021
SYSTIME	3	1.36667	1.30000	1.40000	0.05774	0.03333	41.00	0.0006
ELAPSED	3	47.00000	40.00000	51.00000	6.08276	3.51188	13.38	0.0055
PCPU	3	71.00000	69.00000	75.00000	3.46410	2.00000	35.50	0.0008
MEMPROG	3	49.00000	49.00000	49.00000	0.00000	0.00000	.	.
MEMDATA	3	243.66667	243.00000	245.00000	1.15470	0.66667	365.50	0.0001
NREAD	3	3.66667	1.00000	6.00000	2.51661	1.45297	2.52	0.1276
NWRITE	3	82.33333	81.00000	85.00000	2.30940	1.33333	61.75	0.0003
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	35840.00000	35840.00000	35840.00000	0.00000	0.00000	.	.
EXECTIME	3	33.73333	30.70000	35.60000	2.65016	1.53007	22.05	0.0021
EXECMEM	3	299690.66667	299008.00000	301056.00000	1182.41335	682.66667	439.00	0.0001
IDNO=J								
USERTIME	3	88.36667	86.80000	89.20000	1.35769	0.78387	112.73	0.0001
SYSTIME	3	0.70000	0.60000	0.80000	0.10000	0.05774	12.12	0.0067
ELAPSED	3	92.66667	91.00000	94.00000	1.52753	0.88192	105.07	0.0001
PCPU	3	95.33333	95.00000	96.00000	0.57735	0.33333	286.00	0.0001
MEMPROG	3	47.00000	47.00000	47.00000	0.00000	0.00000	.	.
MEMDATA	3	134.00000	134.00000	134.00000	0.00000	0.00000	.	.
NREAD	3	3.00000	1.00000	6.00000	2.64575	1.52753	1.96	0.1885
NWRITE	3	46.00000	45.00000	47.00000	1.00000	0.57735	79.67	0.0002
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	35840.00000	35840.00000	35840.00000	0.00000	0.00000	.	.
EXECTIME	3	89.06667	87.50000	90.00000	1.36504	0.78811	113.01	0.0001
EXECMEM	3	185344.00000	185344.00000	185344.00000	0.00000	0.00000	.	.

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 5323 B

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=A								
USERTIME	3	15.13333	15.00000	15.30000	0.15275	0.08819	171.60	0.0001
SYSTIME	3	2.90000	2.80000	3.00000	0.10000	0.05774	50.23	0.0004
ELAPSED	3	32.33333	32.00000	33.00000	0.57735	0.33333	97.00	0.0001
PCPU	3	54.66667	54.00000	55.00000	0.57735	0.33333	164.00	0.0001
MEMPROG	3	66.00000	66.00000	66.00000	0.00000	0.00000	.	.
MEMDATA	3	128.33333	128.00000	129.00000	0.57735	0.33333	385.00	0.0001
NREAD	3	6.00000	6.00000	6.00000	0.00000	0.00000	.	.
NWRITE	3	71.66667	70.00000	74.00000	2.08167	1.20185	59.63	0.0003
NPGFLT	3	2.00000	2.00000	2.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	52224.00000	52224.00000	52224.00000	0.00000	0.00000	.	.
EXECTIME	3	18.03333	17.90000	18.20000	0.15275	0.08819	204.48	0.0001
EXECMEM	3	198997.33333	198656.00000	199680.00000	591.20668	341.33333	583.00	0.0001
IDNO=B								
USERTIME	3	13.00000	12.70000	13.30000	0.30000	0.17321	75.06	0.0002
SYSTIME	3	1.60000	1.50000	1.70000	0.10000	0.05774	27.71	0.0013
ELAPSED	3	26.33333	26.00000	27.00000	0.57735	0.33333	79.00	0.0002
PCPU	3	53.66667	53.00000	55.00000	1.15470	0.66667	80.50	0.0002
MEMPROG	3	64.33333	64.00000	65.00000	0.57735	0.33333	193.00	0.0001
MEMDATA	3	120.00000	120.00000	120.00000	0.00000	0.00000	.	.
NREAD	3	4.66667	1.00000	11.00000	5.50757	3.17980	1.47	0.2799
NWRITE	3	27.33333	25.00000	31.00000	3.21455	1.85592	14.73	0.0046
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	51200.00000	51200.00000	51200.00000	0.00000	0.00000	.	.
EXECTIME	3	14.60000	14.40000	14.90000	0.26458	0.15275	95.58	0.0001
EXECMEM	3	188757.33333	188416.00000	189440.00000	591.20668	341.33333	553.00	0.0001

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 5323 B

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=C								
USERTIME	3	36.83333	36.60000	37.10000	0.25166	0.14530	253.50	0.0001
SYSTIME	3	82.30000	81.40000	83.30000	0.95394	0.55076	149.43	0.0001
ELAPSED	3	125.66667	124.00000	128.00000	2.08167	1.20185	104.56	0.0001
PCPU	3	94.00000	92.00000	96.00000	2.00000	1.15470	81.41	0.0002
MEMPROG	3	73.00000	73.00000	73.00000	0.00000	0.00000	.	.
MEMDATA	3	82.00000	82.00000	82.00000	0.00000	0.00000	.	.
NREAD	3	5.00000	2.00000	9.00000	3.60555	2.08167	2.40	0.1383
NWRITE	3	67.33333	65.00000	69.00000	2.08167	1.20185	56.02	0.0003
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	56320.00000	56320.00000	56320.00000	0.00000	0.00000	.	.
EXECTIME	3	119.13333	118.50000	119.90000	0.70946	0.40961	290.85	0.0001
EXECMEM	3	158720.00000	158720.00000	158720.00000	0.00000	0.00000	.	.
IDNO=D								
USERTIME	3	31.60000	31.10000	32.00000	0.45826	0.26458	119.44	0.0001
SYSTIME	3	27.96667	27.50000	28.50000	0.50332	0.29059	96.24	0.0001
ELAPSED	3	70.00000	68.00000	72.00000	2.00000	1.15470	60.62	0.0003
PCPU	3	84.00000	82.00000	86.00000	2.00000	1.15470	72.75	0.0002
MEMPROG	3	63.00000	63.00000	63.00000	0.00000	0.00000	.	.
MEMDATA	3	189.33333	189.00000	190.00000	0.57735	0.33333	568.00	0.0001
NREAD	3	1.00000	0.00000	2.00000	1.00000	0.57735	1.73	0.2254
NWRITE	3	103.33333	101.00000	106.00000	2.51661	1.45297	71.12	0.0002
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	49152.00000	49152.00000	49152.00000	0.00000	0.00000	.	.
EXECTIME	3	59.56667	59.20000	59.90000	0.35119	0.20276	293.78	0.0001
EXECMEM	3	258389.33333	258048.00000	259072.00000	591.20668	341.33333	757.00	0.0001

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 5323 B

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=E								
USERTIME	3	17.36667	17.30000	17.40000	0.05774	0.03333	521.00	0.0001
SYSTIME	3	0.93333	0.90000	1.00000	0.05774	0.03333	28.00	0.0013
ELAPSED	3	22.33333	22.00000	23.00000	0.57735	0.33333	67.00	0.0002
PCPU	3	80.00000	78.00000	82.00000	2.00000	1.15470	69.28	0.0002
MEMPROG	3	51.00000	51.00000	51.00000	0.00000	0.00000	.	.
MEMDATA	3	159.66667	159.00000	160.00000	0.57735	0.33333	479.00	0.0001
NREAD	3	1.33333	0.00000	3.00000	1.52753	0.88192	1.51	0.2697
NWRITE	3	36.66667	35.00000	38.00000	1.52753	0.88192	41.58	0.0006
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	37888.00000	37888.00000	37888.00000	0.00000	0.00000	.	.
EXECTIME	3	18.30000	18.20000	18.40000	0.10000	0.05774	316.97	0.0001
EXECMEM	3	215722.66667	215040.00000	216064.00000	591.20668	341.33333	632.00	0.0001
IDNO=F								
USERTIME	3	16.13333	15.20000	16.70000	0.81445	0.47022	34.31	0.0008
SYSTIME	3	1.40000	1.30000	1.50000	0.10000	0.05774	24.25	0.0017
ELAPSED	3	17.33333	16.00000	18.00000	1.15470	0.66667	26.00	0.0015
PCPU	3	98.00000	98.00000	98.00000	0.00000	0.00000	.	.
MEMPROG	3	65.00000	65.00000	65.00000	0.00000	0.00000	.	.
MEMDATA	3	183.00000	182.00000	184.00000	1.00000	0.57735	316.97	0.0001
NREAD	3	0.66667	0.00000	1.00000	0.57735	0.33333	2.00	0.1835
NWRITE	3	72.00000	71.00000	73.00000	1.00000	0.57735	124.71	0.0001
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	57344.00000	57344.00000	57344.00000	0.00000	0.00000	.	.
EXECTIME	3	17.53333	16.60000	18.00000	0.80829	0.46667	37.57	0.0007
EXECMEM	3	253952.00000	252928.00000	254976.00000	1024.00000	591.20668	429.55	0.0001

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 5323 B

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=G								
USERTIME	3	16.63333	15.50000	17.70000	1.10151	0.63596	26.15	0.0015
SYSTIME	3	1.50000	1.40000	1.60000	0.10000	0.05774	25.98	0.0015
ELAPSED	3	26.66667	25.00000	28.00000	1.52753	0.88192	30.24	0.0011
PCPU	3	66.66667	65.00000	68.00000	1.52753	0.88192	75.59	0.0002
MEMPROG	3	55.00000	55.00000	55.00000	0.00000	0.00000	.	.
MEMDATA	3	104.33333	104.00000	105.00000	0.57735	0.33333	313.00	0.0001
NREAD	3	1.00000	0.00000	2.00000	1.00000	0.57735	1.73	0.2254
NWRITE	3	58.66667	58.00000	59.00000	0.57735	0.33333	176.00	0.0001
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	36864.00000	36864.00000	36864.00000	0.00000	0.00000	.	.
EXECTIME	3	18.13333	17.10000	19.20000	1.05040	0.60645	29.90	0.0011
EXECMEM	3	163157.33333	162816.00000	163840.00000	591.20668	341.33333	478.00	0.0001
IDNO=H								
USERTIME	3	26.46667	26.00000	26.90000	0.45092	0.26034	101.66	0.0001
SYSTIME	3	2.53333	2.50000	2.60000	0.05774	0.03333	76.00	0.0002
ELAPSED	3	41.00000	39.00000	43.00000	2.00000	1.15470	35.51	0.0008
PCPU	3	69.66667	68.00000	71.00000	1.52753	0.88192	78.99	0.0002
MEMPROG	3	62.00000	62.00000	62.00000	0.00000	0.00000	.	.
MEMDATA	3	205.33333	205.00000	206.00000	0.57735	0.33333	616.00	0.0001
NREAD	3	3.00000	2.00000	4.00000	1.00000	0.57735	5.20	0.0351
NWRITE	3	89.33333	88.00000	92.00000	2.30940	1.33333	67.00	0.0002
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	47104.00000	47104.00000	47104.00000	0.00000	0.00000	.	.
EXECTIME	3	29.00000	28.50000	29.40000	0.45826	0.26458	109.61	0.0001
EXECMEM	3	273749.33333	273408.00000	274432.00000	591.20668	341.33333	802.00	0.0001

STATISTICS OF THE DYNAMIC MEASUREMENTS BY IDNO

COMSC 5323 B

VARIABLE	N	MEAN	MINIMUM VALUE	MAXIMUM VALUE	STANDARD DEVIATION	STD ERROR OF MEAN	T	PR>|T|
IDNO=I								
USERTIME	3	17.76667	17.40000	18.20000	0.40415	0.23333	76.14	0.0002
SYSTIME	3	1.06667	1.00000	1.10000	0.05774	0.03333	32.00	0.0010
ELAPSED	3	29.00000	28.00000	30.00000	1.00000	0.57735	50.23	0.0004
PCPU	3	63.33333	59.00000	67.00000	4.04145	2.33333	27.14	0.0014
MEMPROG	3	49.00000	49.00000	49.00000	0.00000	0.00000	.	.
MEMDATA	3	190.33333	190.00000	191.00000	0.57735	0.33333	571.00	0.0001
NREAD	3	1.33333	1.00000	2.00000	0.57735	0.33333	4.00	0.0572
NWRITE	3	53.33333	53.00000	54.00000	0.57735	0.33333	160.00	0.0001
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	35840.00000	35840.00000	35840.00000	0.00000	0.00000	.	.
EXECTIME	3	18.83333	18.40000	19.30000	0.45092	0.26034	72.34	0.0002
EXECMEM	3	245077.33333	244736.00000	245760.00000	591.20668	341.33333	718.00	0.0001
IDNO=J								
USERTIME	3	3.63333	3.50000	3.70000	0.11547	0.06667	54.50	0.0003
SYSTIME	3	0.40000	0.40000	0.40000	0.00000	0.00000	.	.
ELAPSED	3	8.00000	7.00000	9.00000	1.00000	0.57735	13.86	0.0052
PCPU	3	49.00000	44.00000	53.00000	4.58258	2.64575	18.52	0.0029
MEMPROG	3	47.00000	47.00000	47.00000	0.00000	0.00000	.	.
MEMDATA	3	102.66667	102.00000	103.00000	0.57735	0.33333	308.00	0.0001
NREAD	3	1.00000	0.00000	3.00000	1.73205	1.00000	1.00	0.4226
NWRITE	3	19.66667	19.00000	21.00000	1.15470	0.66667	29.50	0.0011
NPGFLT	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
NSWAPPED	3	0.00000	0.00000	0.00000	0.00000	0.00000	.	.
OBJSIZE	3	35840.00000	35840.00000	35840.00000	0.00000	0.00000	.	.
EXECTIME	3	4.03333	3.90000	4.10000	0.11547	0.06667	60.50	0.0003
EXECMEM	3	153258.66667	152576.00000	153600.00000	591.20668	341.33333	449.00	0.0001

APPENDIX K

STATIC AND DYNAMIC MEASUREMENTS

STATIC AND DYNAMIC MEASUREMENTS

COMSC 2123

IDNO	# OF NBNC LINES OF CODE	MCCABE'S CYCLOMATIC COMPLEXITY	# OF UNIQUE OPERATORS	TOTAL # OF OPERATORS	# OF UNIQUE OPERANDS	TOTAL # OF OPERANDS	PROGRAM LENGTH	EST. PROGRAM LENGTH	PROGRAM VOLUME
A	145	30	42	448	62	283	731	595.64	4898.02
B	96	20	38	294	36	207	501	385.54	3110.94
C	117	19	42	347	42	212	559	452.95	3573.31
D	102	18	40	291	41	190	481	432.54	3049.47
E	92	18	41	292	36	189	481	405.78	3014.32
F	89	15	41	268	40	180	448	432.54	2840.25
G	117	20	40	371	34	223	594	385.85	3688.42
H	132	23	39	390	48	243	633	474.21	4078.38
I	85	13	39	228	35	151	379	385 66	2353.38
J	57	8	32	178	35	117	295	339.52	1789.50

IDNO	EST. POTENTIAL PROGRAM VOLUME	EST. PROGRAM LEVEL	EST. PROGRAM DIFFICULTY	# OF ELEM. MENTAL DISCRIM.	DEVELOPMENT TIME (HOURS)	LANGUAGE LEVEL	MEAN EXECUTION TIME (SECS)
A	51.10	0.0104	95.855	469499	7.24536	0.53	0.2549
B	28.48	0.0092	109.250	339870	5.24490	0.26	0.1541
C	33.71	0.0094	106.000	378770	5.84522	0.32	0.2164
D	32.90	0.0108	92.683	282634	4.36163	0.35	0.1843
E	28.01	0.0093	107.625	324417	5.00643	0.26	0.2015
F	30.79	0.0108	92.250	262013	4.04342	0.33	0.1381
G	28.12	0.0076	131.176	483833	7.46656	0.21	0.1353
H	41.31	0.0101	98.719	402613	6.21316	0.42	0.1226
I	27.97	0.0119	84.129	197987	3.05535	0.33	0.1424
J	33.46	0.0187	53.486	95712	1.47704	0.63	0.1871

IDNO	MEAN EXECUTION MEMORY (BYTES)	MEAN OBJECT CODE SIZE (BYTES)	MEAN TOTAL SYSTEM TIME (SECS)	MEAN TOTAL JOB TIME (SECS)	MEAN COMPILATION TIME (SECS)	MEAN # OF SOURCE STMTS. EXEC.	# OF COMPILATIONS
A	10176	10200	0.69	0.5892	0.3343	5885	13
B	10152	5963	0.40	0.3053	0.1512	2098	66
C	10176	6323	0.50	0.4010	0.1846	2687	59
D	10176	5614	0.45	0.3473	0.1629	2225	69
E	10152	4316	0.41	0.3082	0.1066	2452	283
F	10080	4177	0.35	0.2484	0.1102	2284	20
G	10080	6599	0.40	0.2995	0.1642	2651	1
H	10080	7535	0.41	0.3148	0.1921	2549	26
I	10080	4277	0.37	0.2700	0.1277	2081	32
J	10152	3045	0.36	0.2686	0.0815	2934	37

STATIC AND DYNAMIC MEASUREMENTS

COMSC 2123

IDNO	# OF NBNC LINES OF CODE	MCCABE'S CYCLOMATIC COMPLEXITY	# OF UNIQUE OPERATORS	TOTAL # OF OPERATORS	# OF UNIQUE OPERANDS	TOTAL # OF OPERANDS	PROGRAM LENGTH	EST. PROGRAM LENGTH	PROGRAM VOLUME
K	90	15	39	247	43	229	476	439.46	3026.19
L	96	17	42	289	41	186	475	446.14	3028.14

IDNO	EST. POTENTIAL PROGRAM VOLUME	EST. PROGRAM LEVEL	EST. PROGRAM DIFFICULTY	# OF ELEM. MENTAL DISCRIM.	DEVELOPMENT TIME (HOURS)	LANGUAGE LEVEL	MEAN EXECUTION TIME (SECS)
K	29.14	0.0096	103.849	314267	4.84980	0.28	0.1794
L	31.79	0.0105	95.268	288486	4.45194	0.33	0.1567

IDNO	MEAN EXECUTION MEMORY (BYTES)	MEAN OBJECT CODE SIZE (BYTES)	MEAN TOTAL SYSTEM TIME (SECS)	MEAN TOTAL JOB TIME (SECS)	MEAN COMPILATION TIME (SECS)	MEAN # OF SOURCE STMTS.	# OF COMPILATIONS EXEC.
K	10152	4782	0.41	0.3129	0.1335	2290	51
L	10152	5072	0.40	0.3038	0.1471	2462	50

STATIC AND DYNAMIC MEASUREMENTS

COMSC 2133

IDNO	# OF NBNC LINES OF CODE	MCCABE'S CYCLOMATIC COMPLEXITY	# OF UNIQUE OPERATORS	TOTAL # OF OPERATORS	# OF UNIQUE OPERANDS	TOTAL # OF OPERANDS	PROGRAM LENGTH	EST. PROGRAM LENGTH	PROGRAM VOLUME
A	309	52	54	859	76	547	1406	785.61	9873.4
B	344	65	64	1056	114	643	1699	1162.95	12701.3
C	232	37	51	664	67	392	1056	695.72	7268.1
D	412	52	57	1070	128	600	1670	1228.47	12577.4
E	351	49	52	890	118	546	1436	1108.57	10639.9
F	361	52	51	950	101	585	1535	961.77	11125.6
G	293	46	55	972	97	638	1610	958.17	11669.2
H	220	40	54	775	74	481	1256	770.26	8792.0
I	235	45	58	745	64	497	1242	723.76	8608.0
J	142	22	44	387	40	181	568	453.09	3630.8

IDNO	EST. POTENTIAL PROGRAM VOLUME	EST. PROGRAM LEVEL	EST. PROGRAM DIFFICULTY	# OF ELEM. MENTAL DISCRIM.	DEVELOPMENT TIME (HOURS)	LANGUAGE LEVEL	MEAN EXECUTION TIME (SECS)
A	50.81	0.0051	194.329	1918697	29.6095	0.26	0.8762
B	70.37	0.0055	180.491	2292468	35.3776	0.39	0.6742
C	48.72	0.0067	149.194	1084353	16.7338	0.33	0.3950
D	94.15	0.0075	133.594	1680263	25.9300	0.70	0.6121
E	88.44	0.0083	120.305	1280032	19.7536	0.74	0.5704
F	75.33	0.0068	147.698	1643224	25.3584	0.51	0.9000
G	64.51	0.0055	180.876	2110675	32.5721	0.36	0.5658
H	50.10	0.0057	175.500	1542996	23.8117	0.29	0.6308
I	38.22	0.0044	225.203	1938543	29.9158	0.17	0.4459
J	36.47	0.0100	99.550	361450	5.5779	0.37	0.2573

IDNO	MEAN EXECUTION MEMORY (BYTES)	MEAN OBJECT CODE SIZE (BYTES)	MEAN TOTAL SYSTEM TIME (SECS)	MEAN TOTAL JOB TIME (SECS)	MEAN COMPILATION TIME (SECS)	MEAN # OF SOURCE STMTS. EXEC.	# OF COMPILATIONS
A	10176	16963	1.42	1.3227	0.4465	14172	108
B	25176	21497	1.54	1.4403	0.7661	7379	1
C	10176	12736	0.89	0.7890	0.3939	5491	17
D	10176	22005	1.40	1.2973	0.6851	8384	10
E	10176	17074	1.19	1.0871	0.5166	10677	9
F	10176	17345	1.51	1.4055	0.5054	13549	31
G	10176	17638	1.24	1.1351	0.5693	4854	15
H	10176	14891	1.21	1.1168	0.4861	4819	12
I	10176	13881	0.99	0.8890	0.4431	4883	97
J	13416	6145	0.58	0.4772	0.2199	4390	264

STATIC AND DYNAMIC MEASUREMENTS

COMSC 2133

IDNO	# OF NBNC LINES OF CODE	MCCABE'S CYCLOMATIC COMPLEXITY	# OF UNIQUE OPERATORS	TOTAL # OF OPERATORS	# OF UNIQUE OPERANDS	TOTAL # OF OPERANDS	PROGRAM LENGTH	EST. PROGRAM LENGTH	PROGRAM VOLUME
K	291	41	56	879	100	550	1429	989.60	10410.8
L	263	48	57	781	80	455	1236	838.23	8773.2

IDNO	EST. POTENTIAL PROGRAM VOLUME	EST. PROGRAM LEVEL	EST. PROGRAM DIFFICULTY	# OF ELEM. MENTAL DISCRIM.	DEVELOPMENT TIME (HOURS)	LANGUAGE LEVEL	MEAN EXECUTION TIME (SECS)
K	67.60	0.0065	154.000	1603269	24.7418	0.44	0.5525
L	54.12	0.0062	162.094	1422076	21.9456	0.33	0.5009

IDNO	MEAN EXECUTION MEMORY (BYTES)	MEAN OBJECT CODE SIZE (BYTES)	MEAN TOTAL SYSTEM TIME (SECS)	MEAN TOTAL JOB TIME (SECS)	MEAN COMPILATION TIME (SECS)	MEAN # OF SOURCE STMTS. EXEC.	# OF COMPILATIONS
K	10176	16289	1.15	1.0484	0.4959	5656	68
L	10176	14298	1.01	0.9153	0.4144	4312	45

STATIC AND DYNAMIC MEASUREMENTS

COMSC 4323

IDNO	# OF NBNC LINES OF CODE	MCCABE'S CYCLOMATIC COMPLEXITY	# OF UNIQUE OPERATORS	TOTAL # OF OPERATORS	# OF UNIQUE OPERANDS	TOTAL # OF OPERANDS	PROGRAM LENGTH	EST. PROGRAM LENGTH	PROGRAM VOLUME
A	801	121	79	2222	156	1570	3792	1634.52	29867.8
B	694	103	72	2326	181	1665	3991	1801.71	31860.1
C	742	120	65	2570	159	1983	4553	1554.20	35546.9
D	817	125	64	2644	174	1970	4614	1679.07	36426.7
E	837	141	68	2582	189	1719	4301	1843.21	34432.2
F	1146	195	69	3201	177	2448	5649	1743.25	44867.3
G	833	121	63	3048	207	2264	5312	1969.12	42904.0

IDNO	EST. POTENTIAL PROGRAM VOLUME	EST. PROGRAM LEVEL	EST. PROGRAM DIFFICULTY	# OF ELEM. MENTAL DISCRIM.	DEVELOPMENT TIME (HOURS)	LANGUAGE LEVEL	MEAN EXECUTION TIME (SECS)
A	75.13	0.0025	397.532	11873389	183.231	0.19	2.6
B	96.21	0.0030	331.160	10550807	162.821	0.29	2.9
C	87.70	0.0025	405.330	14408226	222.349	0.22	2.1
D	100.54	0.0028	362.299	13197348	203.663	0.28	1.9
E	111.35	0.0032	309.238	10647745	164.317	0.36	3.6
F	94.03	0.0021	477.153	21408529	330.379	0.20	0.4
G	124.53	0.0029	344.522	14781376	228.108	0.36	2.7

IDNO	MEAN EXECUTION MEMORY (BYTES)	MEAN OBJECT CODE SIZE (BYTES)	MEAN USER TIME (SECS)	MEAN SYSTEM TIME (SECS)	MEAN ELAPSED TIME (H:MM:SS)	# OF COMPILATIONS
A	89088	21504	1.4	1.2	0:00:02	33
B	126293	22528	2.6	0.3	0:00:03	204
C	121856	23552	1.9	0.2	0:00:02	12
D	120491	23552	1.7	0.2	0:00:01	188
E	92160	21504	3.2	0.4	0:00:03	19
F	81579	21504	0.3	0.1	0:00:00	69
G	141312	24576	2.5	0.2	0:00:02	66

STATIC AND DYNAMIC MEASUREMENTS

COMSC 5323

IDNO	# OF NBNC LINES OF CODE	MCCABE'S CYCLOMATIC COMPLEXITY	# OF UNIQUE OPERATORS	TOTAL # OF OPERATORS	# OF UNIQUE OPERANDS	TOTAL # OF OPERANDS	PROGRAM LENGTH	EST. PROGRAM LENGTH	PROGRAM VOLUME
A	2571	547	148	8051	565	4629	12680	6232.29	120178
B	2576	587	141	8740	671	5114	13854	7307.48	133904
C	2553	426	131	8730	426	5416	14146	4642.36	129033
D	2210	470	146	7435	444	4689	12124	4954.44	111596
E	1381	369	95	5994	306	4099	10093	3150.90	87279
F	2559	349	158	7163	650	4143	11306	7227.79	109196
G	1511	328	79	6408	290	4971	11379	2870.17	97034
H	2455	486	115	7109	479	4590	11699	5052.19	107798
I	1582	302	71	5327	302	3913	9240	2924.63	78938
J	1878	397	119	6022	396	3981	10003	4237.71	90111

IDNO	EST. POTENTIAL PROGRAM VOLUME	EST. PROGRAM LEVEL	EST. PROGRAM DIFFICULTY	# OF ELEM. MENTAL DISCRIM.	DEVELOPMENT TIME (HOURS)	LANGUAGE LEVEL	MEAN EXECUTION TIME (SECS)
A	198.22	0.0016	606.276	72861035	1124.40	0.33	48.0
B	249.21	0.0019	537.313	71948121	1110.31	0.46	45.6
C	154.95	0.0012	832.742	107451348	1658.20	0.19	336.7
D	144.75	0.0013	770.939	86033900	1327.68	0.19	146.3
E	137.17	0.0016	636.283	55533987	857.01	0.22	48.7
F	216.86	0.0020	503.534	54983750	848.51	0.43	126.6
G	143.31	0.0015	677.084	65700325	1013.89	0.21	53.9
H	195.64	0.0018	550.992	59395974	916.60	0.36	77.3
I	171.61	0.0022	459.972	36309081	560.33	0.37	52.6
J	150.65	0.0017	598.155	53900559	831.80	0.25	93.1

IDNO	MEAN EXECUTION MEMORY (BYTES)	MEAN OBJECT CODE SIZE (BYTES)	MEAN USER TIME (SECS)	MEAN SYSTEM TIME (SECS)	MEAN ELAPSED TIME (H:MM:SS)	# OF COMPILATIONS
A	204800	52224	42.0	6.0	0:01:16	325
B	192512	51200	41.9	3.6	0:01:09	452
C	162816	56320	102.4	234.2	0:05:52	527
D	257024	49152	84.2	62.1	0:02:48	668
E	226987	37888	46.5	2.2	0:00:56	702
F	278528	57344	124.0	2.7	0:02:06	196
G	178176	36864	50.5	3.4	0:01:13	1181
H	273067	47104	72.2	5.2	0:01:40	1525
I	299691	35840	50.1	2.4	0:01:16	136
J	185344	35840	92.0	1.1	0:01:41	149

APPENDIX L

CORRELATIONS BETWEEN ALL MEASUREMENTS

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2123

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
LOC	12	101.5000000	23.3452351	1218.000000	57.00000000	145.0000000
VG	12	18.0000000	5.4104276	216.000000	8.00000000	30.0000000
NVAR	12	24.3333333	6.9848321	292.000000	11.00000000	35.0000000
NCONST	12	2.6666667	1.2309149	32.000000	1.000000000E+00	5.0000000
NTYPE	12	1.1666667	3.892494721E-01	14.000000	1.000000000E+00	2.0000000
NSUB	12	8.7500000	1.2154311	105.000000	6.00000000	10.0000000
ETA1	12	39.5833333	2.7455198	475.000000	32.00000000	42.0000000
N1	12	303.5833333	74.6524777	3643.000000	178.00000000	448.0000000
ETA2	12	41.0833333	7.7864490	493.000000	34.00000000	62.0000000
N2	12	200.8333333	43.0640819	2410.000000	117.00000000	283.0000000
PLEN	12	504.4166667	114.5849890	6053.000000	295.00000000	731.0000000
EPLEN	12	431.3191667	63.9069186	5175.830000	339.52000000	595.6400000
PVOL	12	3204.1933333	797.3271445	38450.320000	1789.50000000	4898.0200000
EPPVOL	12	33.0650000	6.8293132	396.780000	27.97000000	51.1000000
EPLEV	12	1.069166667E-02	2.741419119E-03	1.283000000E-01	7.600000000E-03	1.870000000E-02
EPDIF	12	97.5241667	18.2486850	1170.290000	53.48600000	131.1760000
EFFORT	12	320008.4166667	109071.0103227	3840101.000000	95712.00000000	483833.0000000
TIME	12	4.9384016	1.6831944	59.260819	1.47704444	7.4665639
LAMBDA	12	3.541666667E-01	1.199589576E-01	4.250000	2.100000000E-01	6.300000000E-01
MTSYSTIM	12	4.300000000E-01	9.087954065E-02	5.160000	3.500000000E-01	6.900000000E-01
MNSTMTS	12	2716.5000000	1029.1911652	32598.000000	2081.00000000	5885.0000000
MTJOBTIM	12	3.307625000E-01	9.033108829E-02	3.969150	2.484000000E-01	5.892000000E-01
MCOMPTIM	12	1.580083333E-01	6.431012162E-02	1.896100	8.150000000E-02	3.343500000E-01
MEXECTIM	12	1.727625000E-01	3.873296766E-02	2.073150	1.226500000E-01	2.549000000E-01
MOBJSIZE	12	5658.5833333	1892.9434251	67903.000000	3045.00000000	10200.0000000
MEXECMEM	12	10134.0000000	41.0941270	121608.000000	10080.00000000	10176.0000000
NCOMP	12	58.9166667	73.7150020	707.000000	1.000000000E+00	283.0000000

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2123

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	LOC	VG	NVAR	NCONST	NTYPE	NSUB	ETA1	N1	ETA2
LOC	1.00000	0.94646	0.56532	0.34800	0.10004	0.86986	0.63755	0.98085	0.73242
# OF NBNC LINES OF CODE	0.0000	0.0001	0.0554	0.2677	0.7571	0.0002	0.0257	0.0001	0.0068
VG	0.94646	1.00000	0.58215	0.38221	0.12950	0.77416	0.61200	0.95635	0.75743
MCCABE'S CYCLOMATIC COMPLEXITY	0.0001	0.0000	0.0470	0.2201	0.6883	0.0031	0.0344	0.0001	0.0043
NVAR	0.56532	0.58215	1.00000	0.02467	-0.15604	0.73887	0.88016	0.59376	0.49254
# OF VARIABLES	0.0554	0.0470	0.0000	0.9393	0.6282	0.0060	0.0002	0.0418	0.1038
NCONST	0.34800	0.38221	0.02467	1.00000	-0.44272	0.12153	0.08967	0.28624	0.53432
# OF CONSTANTS	0.2677	0.2201	0.9393	0.0000	0.1495	0.7067	0.7817	0.3671	0.0735
NTYPE	0.10004	0.12950	-0.15604	-0.44272	1.00000	0.09608	0.07089	0.10585	-0.12498
# OF TYPE DEFINITIONS	0.7571	0.6883	0.6282	0.1495	0.0000	0.7664	0.8267	0.7434	0.6988
NSUB	0.86986	0.77416	0.73887	0.12153	0.09608	1.00000	0.81047	0.87443	0.47309
# OF SUBROUTINES	0.0002	0.0031	0.0060	0.7067	0.7664	0.0000	0.0014	0.0002	0.1203
ETA1	0.63755	0.61200	0.88016	0.08967	0.07089	0.81047	1.00000	0.60718	0.39725
# OF UNIQUE OPERATORS	0.0257	0.0344	0.0002	0.7817	0.8267	0.0014	0.0000	0.0363	0.2010
N1	0.98085	0.95635	0.59376	0.28624	0.10585	0.87443	0.60718	1.00000	0.69509
TOTAL # OF OPERATORS	0.0001	0.0001	0.0418	0.3671	0.7434	0.0002	0.0363	0.0000	0.0121
ETA2	0.73242	0.75743	0.49254	0.53432	-0.12498	0.47309	0.39725	0.69509	1.00000
# OF UNIQUE OPERANDS	0.0068	0.0043	0.1038	0.0735	0.6988	0.1203	0.2010	0.0121	0.0000
N2	0.90616	0.91496	0.57051	0.26297	0.09400	0.75640	0.58603	0.88686	0.75483
TOTAL # OF OPERANDS	0.0001	0.0001	0.0527	0.4089	0.7714	0.0044	0.0452	0.0001	0.0045
PLEN	0.97959	0.96693	0.60125	0.28532	0.10429	0.85396	0.61582	0.98481	0.73653
PROGRAM LENGTH	0.0001	0.0001	0.0387	0.3687	0.7470	0.0004	0.0330	0.0001	0.0063
EPLEN	0.80614	0.82264	0.67236	0.47800	-0.08825	0.63300	0.62061	0.76773	0.96595
EST. PROGRAM LENGTH	0.0015	0.0010	0.0166	0.1160	0.7851	0.0271	0.0313	0.0036	0.0001
PVOL	0.97859	0.97046	0.61618	0.31890	0.08080	0.83885	0.61789	0.97966	0.78612
PROGRAM VOLUME	0.0001	0.0001	0.0329	0.3123	0.8029	0.0006	0.0323	0.0001	0.0024

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2123

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=O / N = 12

	LOC	VG	NVAR	NCONST	NTYPE	NSUB	ETA1	N1	ETA2
EPPVOL	0.67831	0.69045	0.29938	0.56613	-0.13474	0.37856	0.16613	0.67285	0.92751
EST. POTENTIAL PROGRAM VOLUME	0.0153	0.0129	0.3445	0.0550	0.6763	0.2250	0.6058	0.0165	0.0001
EPLEV	-0.62678	-0.61230	-0.59045	0.05837	-0.23712	-0.76462	-0.79043	-0.60636	-0.14093
EST. PROGRAM LEVEL	0.0292	0.0343	0.0433	0.8570	0.4581	0.0038	0.0022	0.0366	0.6622
EPDIF	0.56341	0.54475	0.49988	-0.24336	0.25854	0.71298	0.63263	0.58297	-0.00025
EST. PROGRAM DIFFICULTY	0.0564	0.0670	0.0979	0.4459	0.4171	0.0092	0.0273	0.0467	0.9994
EFFORT	0.90190	0.87946	0.60386	0.05001	0.16835	0.87286	0.63742	0.92435	0.49589
# OF ELEM. MENTAL DISCRIM.	0.0001	0.0002	0.0376	0.8773	0.6010	0.0002	0.0258	0.0001	0.1011
TIME	0.90190	0.87946	0.60386	0.05001	0.16835	0.87286	0.63742	0.92435	0.49589
DEVELOPMENT TIME (HOURS)	0.0001	0.0002	0.0376	0.8773	0.6010	0.0002	0.0258	0.0001	0.1011
LAMBDA	-0.07418	-0.05463	-0.27414	0.35504	-0.24985	-0.35384	-0.51594	-0.05562	0.43173
LANGUAGE LEVEL	0.8188	0.8661	0.3886	0.2574	0.4335	0.2592	0.0860	0.8637	0.1611
MTSYSTIM	0.73336	0.79040	0.54206	0.25599	0.10280	0.46912	0.42811	0.73230	0.85754
MEAN TOTAL SYSTEM TIME (SECS)	0.0066	0.0022	0.0687	0.4219	0.7506	0.1239	0.1650	0.0068	0.0004
MNSTMTS	0.56541	0.64685	0.45686	0.25934	-0.14705	0.32162	0.18655	0.61118	0.82015
MEAN # OF SOURCE STMTS. EXEC.	0.0554	0.0230	0.1354	0.4157	0.6484	0.3080	0.5616	0.0347	0.0011
MTJOBTIM	0.72733	0.78695	0.53304	0.24315	0.11577	0.45983	0.41819	0.72676	0.85665
MEAN TOTAL JOB TIME (SECS)	0.0073	0.0024	0.0743	0.4463	0.7201	0.1326	0.1761	0.0074	0.0004
MCOMPTIM	0.88359	0.90309	0.50426	0.34766	0.07185	0.61580	0.47947	0.86597	0.88162
MEAN COMPILATION TIME (SECS)	0.0001	0.0001	0.0946	0.2681	0.8244	0.0330	0.1147	0.0003	0.0001
MEXECTIM	0.22931	0.33592	0.40585	-0.01011	0.15059	0.04989	0.17919	0.25715	0.53439
MEAN EXECUTION TIME (SECS)	0.4734	0.2857	0.1905	0.9751	0.6404	0.8776	0.5774	0.4198	0.0735
MOBJSIZE	0.94968	0.95633	0.46092	0.34382	0.11953	0.70577	0.47397	0.94373	0.81092
MEAN OBJECT CODE SIZE (BYTES)	0.0001	0.0001	0.1315	0.2738	0.7114	0.0103	0.1196	0.0001	0.0014
MEXECMEM	0.01933	0.15701	0.16723	-0.12940	0.34100	-0.18565	0.06285	0.03360	0.29832
MEAN EXECUTION MEMORY (BYTES)	0.9525	0.6260	0.6034	0.6886	0.2781	0.5635	0.8461	0.9174	0.3463
NCOMP	-0.21207	-0.07157	0.20240	0.04575	0.02271	-0.05403	0.13681	-0.15557	-0.25799
# OF COMPILATIONS	0.5082	0.8251	0.5281	0.8877	0.9442	0.8675	0.6716	0.6292	0.4182

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2123

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=O / N = 12

	N2	PLEN	EPLEN	PVOL	EPPVOL	EPLEV	EPDIF	EFFORT	TIME
LOC	0.90616	0.97959	0.80614	0.97859	0.67831	-0.62678	0.56341	0.90190	0.90190
# OF NBNC LINES OF CODE	0.0001	0.0001	0.0015	0.0001	0.0153	0.0292	0.0564	0.0001	0.0001
VG	0.91496	0.96693	0.82264	0.97046	0.69045	-0.61230	0.54475	0.87946	0.87946
MCCABE'S CYCLOMATIC COMPLEXITY	0.0001	0.0001	0.0010	0.0001	0.0129	0.0343	0.0670	0.0002	0.0002
NVAR	0.57051	0.60125	0.67236	0.61618	0.29938	-0.59045	0.49988	0.60386	0.60386
# OF VARIABLES	0.0527	0.0387	0.0166	0.0329	0.3445	0.0433	0.0979	0.0376	0.0376
NCONST	0.26297	0.28532	0.47800	0.31890	0.56613	0.05837	-0.24336	0.05001	0.05001
# OF CONSTANTS	0.4089	0.3687	0.1160	0.3123	0.0550	0.8570	0.4459	0.8773	0.8773
NTYPE	0.09400	0.10429	-0.08825	0.08080	-0.13474	-0.23712	0.25854	0.16835	0.16835
# OF TYPE DEFINITIONS	0.7714	0.7470	0.7851	0.8029	0.6763	0.4581	0.4171	0.6010	0.6010
NSUB	0.75640	0.85396	0.63300	0.83885	0.37856	-0.76462	0.71298	0.87286	0.87286
# OF SUBROUTINES	0.0044	0.0004	0.0271	0.0006	0.2250	0.0038	0.0092	0.0002	0.0002
ETA1	0.58603	0.61582	0.62061	0.61789	0.16613	-0.79043	0.63263	0.63742	0.63742
# OF UNIQUE OPERATORS	0.0452	0.0330	0.0313	0.0323	0.6058	0.0022	0.0273	0.0258	0.0258
N1	0.88686	0.98481	0.76773	0.97966	0.67285	-0.60636	0.58297	0.92435	0.92435
TOTAL # OF OPERATORS	0.0001	0.0001	0.0036	0.0001	0.0165	0.0366	0.0467	0.0001	0.0001
ETA2	0.75483	0.73653	0.96595	0.78612	0.92751	-0.14093	-0.00025	0.49589	0.49589
# OF UNIQUE OPERANDS	0.0045	0.0063	0.0001	0.0024	0.0001	0.6622	0.9994	0.1011	0.1011
N2	1.00000	0.95362	0.80949	0.95425	0.59875	-0.68343	0.62930	0.90826	0.90826
TOTAL # OF OPERANDS	0.0000	0.0001	0.0014	0.0001	0.0397	0.0143	0.0283	0.0001	0.0001
PLEN	0.95362	1.00000	0.80440	0.99689	0.66339	-0.65190	0.61631	0.94356	0.94356
PROGRAM LENGTH	0.0001	0.0000	0.0016	0.0001	0.0187	0.0216	0.0328	0.0001	0.0001
EPLEN	0.80949	0.80440	1.00000	0.84734	0.84100	-0.34176	0.17893	0.60566	0.60566
EST. PROGRAM LENGTH	0.0014	0.0016	0.0000	0.0005	0.0006	0.2769	0.5779	0.0369	0.0369
PVOL	0.95425	0.99689	0.84734	1.00000	0.71021	-0.61418	0.56398	0.91817	0.91817
PROGRAM VOLUME	0.0001	0.0001	0.0005	0.0000	0.0097	0.0336	0.0561	0.0001	0.0001

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2123

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	N2	PLEN	EPLEN	PVOL	EPPVOL	EPLEV	EPDIF	EFFORT	TIME
EPPVOL	0.59875	0.66339	0.84100	0.71021	1.00000	0.09450	-0.17822	0.39366	0.39366
EST. POTENTIAL PROGRAM VOLUME	0.0397	0.0187	0.0006	0.0097	0.0000	0.7702	0.5795	0.2055	0.2055
EPLEV	-0.68343	-0.65190	-0.34176	-0.61418	0.09450	1.00000	-0.95072	-0.80640	-0.80640
EST. PROGRAM LEVEL	0.0143	0.0216	0.2769	0.0336	0.7702	0.0000	0.0001	0.0015	0.0015
EPDIF	0.62930	0.61631	0.17893	0.56398	-0.17822	-0.95072	1.00000	0.82995	0.82995
EST. PROGRAM DIFFICULTY	0.0283	0.0328	0.5773	0.0561	0.5795	0.0001	0.0000	0.0008	0.0008
EFFORT	0.90826	0.94356	0.60566	0.91817	0.39366	-0.80640	0.82995	1.00000	1.00000
# OF ELEM. MENTAL DISCRIM.	0.0001	0.0001	0.0369	0.0001	0.2055	0.0015	0.0008	0.0000	0.0001
TIME	0.90826	0.94356	0.60566	0.91817	0.39366	-0.80640	0.82995	1.00000	1.00000
DEVELOPMENT TIME (HOURS)	0.0001	0.0001	0.0369	0.0001	0.2055	0.0015	0.0008	0.0001	0.0000
LAMBDA	-0.15700	-0.09524	0.22572	-0.04100	0.65718	0.81091	-0.80987	-0.36382	-0.36382
LANGUAGE LEVEL	0.6260	0.7684	0.4806	0.8993	0.0202	0.0014	0.0014	0.2450	0.2450
MTSYSTIM	0.71243	0.74484	0.85884	0.78168	0.81631	-0.20580	0.13762	0.57776	0.57776
MEAN TOTAL SYSTEM TIME (SECS)	0.0093	0.0054	0.0003	0.0027	0.0012	0.5211	0.6697	0.0491	0.0491
MNSTMTS	0.55452	0.60659	0.76234	0.65062	0.86177	0.06748	-0.07936	0.41836	0.41836
MEAN # OF SOURCE STMTS. EXEC.	0.0613	0.0365	0.0039	0.0220	0.0003	0.8349	0.8063	0.1759	0.1759
MTJOBTIM	0.71115	0.74076	0.85528	0.77769	0.81547	-0.19971	0.13340	0.57369	0.57369
MEAN TOTAL JOB TIME (SECS)	0.0095	0.0059	0.0004	0.0029	0.0012	0.5337	0.6794	0.0511	0.0511
MCOMPTIM	0.82940	0.87589	0.89260	0.90184	0.84485	-0.34473	0.26755	0.71704	0.71704
MEAN COMPILATION TIME (SECS)	0.0008	0.0002	0.0001	0.0001	0.0005	0.2725	0.4005	0.0087	0.0087
MEXECTIM	0.28162	0.27337	0.51290	0.31646	0.49930	0.10666	-0.13325	0.14742	0.14742
MEAN EXECUTION TIME (SECS)	0.3752	0.3899	0.0881	0.3163	0.0984	0.7414	0.6797	0.6475	0.6475
MOBJSIZE	0.89397	0.95082	0.82965	0.96000	0.78844	-0.46392	0.41939	0.83353	0.83353
MEAN OBJECT CODE SIZE (BYTES)	0.0001	0.0001	0.0008	0.0001	0.0023	0.1287	0.1747	0.0008	0.0008
MEXECMEM	0.12514	0.06892	0.27268	0.09837	0.23887	0.10894	-0.13548	-0.01380	-0.01380
MEAN EXECUTION MEMORY (BYTES)	0.6984	0.8315	0.3912	0.7610	0.4547	0.7361	0.6746	0.9660	0.9660
NCOMP	-0.15070	-0.15799	-0.18062	-0.16749	-0.31069	-0.14009	0.12757	-0.08515	-0.08515
# OF COMPILATIONS	0.6401	0.6238	0.5743	0.6029	0.3257	0.6641	0.6928	0.7925	0.7925

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2123

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	LAMBDA	MTSYSTIM	MNSTMTS	MTJOBTIM	MCOMPTIM	MEXECTIM	MOBJSIZE	MEXECMEM	NCOMP
LOC	-0.07418	0.73336	0.56541	0.72733	0.88359	0.22931	0.94968	0.01933	-0.21207
# OF NBNC LINES OF CODE	0.8188	0.0066	0.0554	0.0073	0.0001	0.4734	0.0001	0.9525	0.5082
VG	-0.05463	0.79040	0.64685	0.78695	0.90309	0.33592	0.95633	0.15701	-0.07157
MCCABE'S CYCLOMATIC COMPLEXITY	0.8661	0.0022	0.0230	0.0024	0.0001	0.2857	0.0001	0.6260	0.8251
NVAR	-0.27414	0.54206	0.45686	0.53304	0.50426	0.40585	0.46092	0.16723	0.20240
# OF VARIABLES	0.3886	0.0687	0.1354	0.0743	0.0946	0.1905	0.1315	0.6034	0.5281
NCONST	0.35504	0.25599	0.25934	0.24315	0.34766	-0.01011	0.34382	-0.12940	0.04575
# OF CONSTANTS	0.2574	0.4219	0.4157	0.4463	0.2681	0.9751	0.2738	0.6886	0.8877
NTYPE	-0.24985	0.10280	-0.14705	0.11577	0.07185	0.15059	0.11953	0.34100	0.02271
# OF TYPE DEFINITIONS	0.4335	0.7506	0.6484	0.7201	0.8244	0.6404	0.7114	0.2781	0.9442
NSUB	-0.35384	0.46912	0.32162	0.45983	0.61580	0.04989	0.70577	-0.18565	-0.05403
# OF SUBROUTINES	0.2592	0.1239	0.3080	0.1326	0.0330	0.8776	0.0103	0.5635	0.8675
ETA1	-0.51594	0.42811	0.18655	0.41819	0.47947	0.17919	0.47397	0.06285	0.13681
# OF UNIQUE OPERATORS	0.0860	0.1650	0.5616	0.1761	0.1147	0.5774	0.1196	0.8461	0.6716
N1	-0.05562	0.73230	0.61118	0.72676	0.86597	0.25715	0.94373	0.03360	-0.15557
TOTAL # OF OPERATORS	0.8637	0.0068	0.0347	0.0074	0.0003	0.4198	0.0001	0.9174	0.6292
ETA2	0.43173	0.85754	0.82015	0.85665	0.88162	0.53439	0.81092	0.29832	-0.25799
# OF UNIQUE OPERANDS	0.1611	0.0004	0.0011	0.0004	0.0001	0.0735	0.0014	0.3463	0.4182
N2	-0.15700	0.71243	0.55452	0.71115	0.82940	0.28162	0.89397	0.12514	-0.15070
TOTAL # OF OPERANDS	0.6260	0.0093	0.0613	0.0095	0.0008	0.3752	0.0001	0.6984	0.6401
PLEN	-0.09524	0.74484	0.60659	0.74076	0.87589	0.27337	0.95082	0.06892	-0.15799
PROGRAM LENGTH	0.7684	0.0054	0.0365	0.0059	0.0002	0.3899	0.0001	0.8315	0.6238
EPLEN	0.22572	0.85884	0.76234	0.85528	0.89260	0.51290	0.82965	0.27268	-0.18062
EST. PROGRAM LENGTH	0.4806	0.0003	0.0039	0.0004	0.0001	0.0881	0.0008	0.3912	0.5743
PVOL	-0.04100	0.78168	0.65062	0.77769	0.90184	0.31646	0.96000	0.09837	-0.16749
PROGRAM VOLUME	0.8993	0.0027	0.0220	0.0029	0.0001	0.3163	0.0001	0.7610	0.6029

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2123

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=O / N = 12

	LAMBDA	MTSYSTIM	MNSTMTS	MTJOBTIM	MCOMPTIM	MEXECTIM	MOBJSIZE	MEXECMEM	NCOMP
EPPVOL	0.65718	0.81631	0.86177	0.81547	0.84485	0.49930	0.78844	0.23887	-0.31069
EST. POTENTIAL PROGRAM VOLUME	0.0202	0.0012	0.0003	0.0012	0.0005	0.0984	0.0023	0.4547	0.3257
EPLEV	0.81091	-0.20580	0.06748	-0.19971	-0.34473	0.10666	-0.46392	0.10894	-0.14009
EST. PROGRAM LEVEL	0.0014	0.5211	0.8349	0.5337	0.2725	0.7414	0.1287	0.7361	0.6641
EPDIF	-0.80987	0.13762	-0.07936	0.13340	0.26755	-0.13325	0.41939	-0.13548	0.12757
EST. PROGRAM DIFFICULTY	0.0014	0.6697	0.8063	0.6794	0.4005	0.6797	0.1747	0.6746	0.6928
EFFORT	-0.36382	0.57776	0.41836	0.57369	0.71704	0.14742	0.83353	-0.01380	-0.08515
# OF ELEM. MENTAL DISCRIM.	0.2450	0.0491	0.1759	0.0511	0.0087	0.6475	0.0008	0.9660	0.7925
TIME	-0.36382	0.57776	0.41836	0.57369	0.71704	0.14742	0.83353	-0.01380	-0.08515
DEVELOPMENT TIME (HOURS)	0.2450	0.0491	0.1759	0.0511	0.0087	0.6475	0.0008	0.9660	0.7925
LAMBDA	1.00000	0.32480	0.56355	0.32925	0.23308	0.38105	0.11578	0.22904	-0.28123
LANGUAGE LEVEL	0.0000	0.3030	0.0564	0.2960	0.4660	0.2217	0.7201	0.4740	0.3759
MTSYSTIM	0.32480	1.00000	0.88933	0.99962	0.92976	0.78780	0.84244	0.52871	-0.10334
MEAN TOTAL SYSTEM TIME (SECS)	0.3030	0.0000	0.0001	0.0001	0.0001	0.0023	0.0006	0.0772	0.7493
MNSTMTS	0.56355	0.88933	1.00000	0.89011	0.82921	0.69937	0.73057	0.33792	-0.20367
MEAN # OF SOURCE STMTS. EXEC.	0.0564	0.0001	0.0000	0.0001	0.0009	0.0114	0.0070	0.2827	0.5255
MTJOBTIM	0.32925	0.99962	0.89011	1.00000	0.92917	0.78966	0.84138	0.53837	-0.10942
MEAN TOTAL JOB TIME (SECS)	0.2960	0.0001	0.0001	0.0000	0.0001	0.0023	0.0006	0.0710	0.7350
MCOMPTIM	0.23308	0.92976	0.82921	0.92917	1.00000	0.50694	0.96752	0.26507	-0.31551
MEAN COMPILATION TIME (SECS)	0.4660	0.0001	0.0009	0.0001	0.0000	0.0925	0.0001	0.4050	0.3178
MEXECTIM	0.38105	0.78780	0.69937	0.78966	0.50694	1.00000	0.35602	0.81541	0.26819
MEAN EXECUTION TIME (SECS)	0.2217	0.0023	0.0114	0.0023	0.0925	0.0000	0.2560	0.0012	0.3993
MOBJSIZE	0.11578	0.84244	0.73057	0.84138	0.96752	0.35602	1.00000	0.14866	-0.29940
MEAN OBJECT CODE SIZE (BYTES)	0.7201	0.0006	0.0070	0.0006	0.0001	0.2560	0.0000	0.6447	0.3445
MEXECMEM	0.22904	0.52871	0.33792	0.53837	0.26507	0.81541	0.14866	1.00000	0.31277
MEAN EXECUTION MEMORY (BYTES)	0.4740	0.0772	0.2827	0.0710	0.4050	0.0012	0.6447	0.0000	0.3223
NCOMP	-0.28123	-0.10334	-0.20367	-0.10942	-0.31551	0.26819	-0.29940	0.31277	1.00000
# OF COMPILATIONS	0.3759	0.7493	0.5255	0.7350	0.3178	0.3993	0.3445	0.3223	0.0000

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2133

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
LOC	12	287.750000	74.3421146	3453.00000	142.0000000	412.000000
VG	12	45.750000	10.4457473	549.00000	22.0000000	65.000000
NVAR	12	45.333333	13.4592539	544.00000	22.0000000	70.000000
NCONST	12	3.416667	4.0330078	41.00000	0.0000000	14.000000
NTYPE	12	9.666667	2.1461735	116.00000	6.0000000	13.000000
NSUB	12	19.083333	3.3154825	229.00000	14.0000000	26.000000
ETA1	12	54.416667	4.8515852	653.00000	44.0000000	64.000000
N1	12	835.666667	187.3627174	10028.00000	387.0000000	1070.000000
ETA2	12	88.250000	25.7509929	1059.00000	40.0000000	128.000000
N2	12	509.583333	127.3009736	6115.00000	181.0000000	643.000000
PLEN	12	1345.250000	312.4658890	16143.00000	568.0000000	1699.000000
EPLEN	12	889.683333	221.7163223	10676.20000	453.0900000	1228.470000
PVOL	12	9672.470833	2526.7945985	116069.65000	3630.8400000	12701.270000
EPPVOL	12	61.570000	18.4352217	738.84000	36.4700000	94.150000
EPLEV	12	6.516666667E-03	1.529012357E-03	7.820000000E-02	4.400000000E-03	1.000000000E-02
EPDIF	12	160.236167	34.0599026	1922.83400	99.5500000	225.203000
EFFORT	12	1573170.500000	512866.0262854	18878046.00000	361450.0000000	2292468.000000
TIME	12	24.277323	7.9146004	291.32787	5.5779278	35.377592
LAMBDA	12	4.075000000E-01	1.693906190E-01	4.89000	1.700000000E-01	7.400000000E-01
MTSYSTIM	12	1.178750	2.796355176E-01	14.14500	5.800000000E-01	1.540000
MNSTMTS	12	7380.500000	3569.2331109	88566.00000	4312.0000000	14172.000000
MTJOBTIM	12	1.077000	2.794355065E-01	12.92400	4.772000000E-01	1.440350
MCOMPTIM	12	4.952166667E-01	1.392367022E-01	5.94260	2.199000000E-01	7.661000000E-01
MEXECTIM	12	5.817833333E-01	1.824611345E-01	6.98140	2.573500000E-01	9.000500000E-01
MOBJSIZE	12	15896.833333	4147.2588501	190762.00000	6145.0000000	22005.000000
MEXECMEM	12	11696.000000	4346.0828758	140352.00000	10176.0000000	25176.000000
NCOMP	12	56.416667	74.4830162	677.00000	1.000000000E+00	264.000000

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2133

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	LOC	VG	NVAR	NCONST	NTYPE	NSUB	ETA1	N1	ETA2
LOC	1.00000	0.81528	0.55813	0.34998	0.65183	0.30806	0.48904	0.91297	0.93516
# OF NBNC LINES OF CODE	0.0000	0.0012	0.0593	0.2648	0.0216	0.3300	0.1066	0.0001	0.0001
VG	0.81528	1.00000	0.68735	0.11923	0.41768	0.63589	0.79691	0.88362	0.74479
MCCABE'S CYCLOMATIC COMPLEXITY	0.0012	0.0000	0.0135	0.7121	0.1767	0.0262	0.0019	0.0001	0.0055
NVAR	0.55813	0.68735	1.00000	0.20991	0.62104	0.67976	0.68125	0.76059	0.60643
# OF VARIABLES	0.0593	0.0135	0.0000	0.5126	0.0311	0.0150	0.0147	0.0041	0.0366
NCONST	0.34998	0.11923	0.20991	1.00000	0.33260	-0.04363	-0.04220	0.27968	0.51011
# OF CONSTANTS	0.2648	0.7121	0.5126	0.0000	0.2908	0.8929	0.8964	0.3786	0.0902
NTYPE	0.65183	0.41768	0.62104	0.33260	1.00000	0.25978	0.31140	0.75435	0.75502
# OF TYPE DEFINITIONS	0.0216	0.1767	0.0311	0.2908	0.0000	0.4148	0.3245	0.0046	0.0045
NSUB	0.30806	0.63589	0.67976	-0.04363	0.25978	1.00000	0.94713	0.58748	0.42885
# OF SUBROUTINES	0.3300	0.0262	0.0150	0.8929	0.4148	0.0000	0.0001	0.0446	0.1642
ETA1	0.48904	0.79691	0.68125	-0.04220	0.31140	0.94713	1.00000	0.71533	0.53538
# OF UNIQUE OPERATORS	0.1066	0.0019	0.0147	0.8964	0.3245	0.0001	0.0000	0.0089	0.0728
N1	0.91297	0.88362	0.76059	0.27968	0.75435	0.58748	0.71533	1.00000	0.90987
TOTAL # OF OPERATORS	0.0001	0.0001	0.0041	0.3786	0.0046	0.0446	0.0089	0.0000	0.0001
ETA2	0.93516	0.74479	0.60643	0.51011	0.75502	0.42885	0.53538	0.90987	1.00000
# OF UNIQUE OPERANDS	0.0001	0.0055	0.0366	0.0902	0.0045	0.1642	0.0728	0.0001	0.0000
N2	0.83154	0.86692	0.83915	0.29820	0.73581	0.60556	0.72715	0.97111	0.82448
TOTAL # OF OPERANDS	0.0008	0.0003	0.0006	0.3465	0.0064	0.0369	0.0074	0.0001	0.0010
PLEN	0.88621	0.88303	0.79795	0.28919	0.75210	0.59897	0.72518	0.99526	0.88148
PROGRAM LENGTH	0.0001	0.0001	0.0019	0.3619	0.0048	0.0396	0.0076	0.0001	0.0002
EPLEN	0.92452	0.79265	0.64909	0.45390	0.72874	0.53135	0.63327	0.93063	0.99230
EST. PROGRAM LENGTH	0.0001	0.0021	0.0224	0.1383	0.0072	0.0754	0.0271	0.0001	0.0001
PVOL	0.90439	0.87932	0.78724	0.31061	0.76007	0.59816	0.71826	0.99818	0.91211
PROGRAM VOLUME	0.0001	0.0002	0.0024	0.3258	0.0041	0.0399	0.0085	0.0001	0.0001

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2133

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	LOC	VG	NVAR	NCONST	NTYPE	NSUB	ETA1	N1	ETA2
EPPVOL	0.90376	0.57899	0.42347	0.56450	0.72300	0.17335	0.28411	0.78183	0.95752
EST. POTENTIAL PROGRAM VOLUME	0.0001	0.0485	0.1701	0.0559	0.0079	0.5900	0.3708	0.0027	0.0001
EPLEV	-0.14400	-0.53305	-0.54806	0.22875	-0.15329	-0.61540	-0.70446	-0.43545	-0.06592
EST. PROGRAM LEVEL	0.6552	0.0743	0.0651	0.4745	0.6343	0.0332	0.0105	0.1571	0.8387
EPDIF	0.00542	0.42639	0.51053	-0.30589	-0.00599	0.57164	0.63470	0.28721	-0.08718
EST. PROGRAM DIFFICULTY	0.9867	0.1669	0.0899	0.3336	0.9853	0.0522	0.0266	0.3654	0.7876
EFFORT	0.57460	0.82754	0.84603	-0.01988	0.48461	0.75873	0.84325	0.81631	0.53309
# OF ELEM. MENTAL DISCRIM.	0.0507	0.0009	0.0005	0.9511	0.1103	0.0042	0.0006	0.0012	0.0743
TIME	0.57460	0.82754	0.84603	-0.01988	0.48461	0.75873	0.84325	0.81631	0.53309
DEVELOPMENT TIME (HOURS)	0.0507	0.0009	0.0005	0.9511	0.1103	0.0042	0.0006	0.0012	0.0743
LAMBDA	0.67002	0.22465	0.08812	0.61247	0.49513	-0.16147	-0.09154	0.42677	0.74356
LANGUAGE LEVEL	0.0171	0.4827	0.7854	0.0343	0.1017	0.6161	0.7772	0.1665	0.0056
MTSYSTIM	0.84472	0.88532	0.60458	0.17059	0.59228	0.42911	0.59445	0.91068	0.76094
MEAN TOTAL SYSTEM TIME (SECS)	0.0005	0.0001	0.0373	0.5961	0.0424	0.1639	0.0415	0.0001	0.0040
MNSTMTS	0.62794	0.49442	0.09969	0.23299	0.11451	-0.27716	-0.06143	0.41791	0.39349
MEAN # OF SOURCE STMTS. EXEC.	0.0288	0.1023	0.7579	0.4662	0.7231	0.3831	0.8496	0.1764	0.2057
MTJOBTIM	0.84304	0.88630	0.60304	0.16944	0.59006	0.43240	0.59781	0.91025	0.75992
MEAN TOTAL JOB TIME (SECS)	0.0006	0.0001	0.0379	0.5986	0.0434	0.1604	0.0401	0.0001	0.0041
MCOMPTIM	0.80305	0.84890	0.71052	0.14687	0.68070	0.71197	0.79226	0.92739	0.86311
MEAN COMPILATION TIME (SECS)	0.0017	0.0005	0.0096	0.6488	0.0148	0.0094	0.0021	0.0001	0.0003
MEXECTIM	0.67814	0.70947	0.38130	0.14744	0.38411	0.11889	0.31091	0.68623	0.50503
MEAN EXECUTION TIME (SECS)	0.0154	0.0098	0.2213	0.6475	0.2177	0.7129	0.3253	0.0137	0.0940
MOBJSIZE	0.91381	0.89617	0.68624	0.22858	0.68357	0.60413	0.74231	0.98546	0.90348
MEAN OBJECT CODE SIZE (BYTES)	0.0001	0.0001	0.0137	0.4749	0.0142	0.0375	0.0057	0.0001	0.0001
MEXECMEM	0.10453	0.42413	0.24773	-0.24543	-0.06706	0.55065	0.47426	0.20668	0.18676
MEAN EXECUTION MEMORY (BYTES)	0.7465	0.1694	0.4376	0.4420	0.8360	0.0635	0.1193	0.5192	0.5611
NCOMP	-0.65237	-0.67756	-0.50462	-0.32385	-0.71675	-0.44559	-0.59625	-0.77677	-0.72519
# OF COMPILATIONS	0.0215	0.0155	0.0943	0.3045	0.0087	0.1466	0.0407	0.0030	0.0076

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2133

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	N2	PLEN	EPLEN	PVOL	EPPVOL	EPLEV	EPDIF	EFFORT	TIME
LOC	0.83154	0.88621	0.92452	0.90439	0.90376	-0.14400	0.00542	0.57460	0.57460
# OF NBNC LINES OF CODE	0.0008	0.0001	0.0001	0.0001	0.0001	0.6552	0.9867	0.0507	0.0507
VG	0.86692	0.88303	0.79265	0.87932	0.57899	-0.53305	0.42639	0.82754	0.82754
MCCABE'S CYCLOMATIC COMPLEXITY	0.0003	0.0001	0.0021	0.0002	0.0485	0.0743	0.1669	0.0009	0.0009
NVAR	0.83915	0.79795	0.64909	0.78724	0.42347	-0.54806	0.51053	0.84603	0.84603
# OF VARIABLES	0.0006	0.0019	0.0224	0.0024	0.1701	0.0651	0.0899	0.0005	0.0005
NCONST	0.29820	0.28919	0.45390	0.31061	0.56450	0.22875	-0.30589	-0.01988	-0.01988
# OF CONSTANTS	0.3465	0.3619	0.1383	0.3258	0.0559	0.4745	0.3336	0.9511	0.9511
NTYPE	0.73581	0.75210	0.72874	0.76007	0.72300	-0.15329	-0.00599	0.48461	0.48461
# OF TYPE DEFINITIONS	0.0064	0.0048	0.0072	0.0041	0.0079	0.6343	0.9853	0.1103	0.1103
NSUB	0.60556	0.59897	0.53135	0.59816	0.17335	-0.61540	0.57164	0.75873	0.75873
# OF SUBROUTINES	0.0369	0.0396	0.0754	0.0399	0.5900	0.0332	0.0522	0.0042	0.0042
ETA1	0.72715	0.72518	0.63327	0.71826	0.28411	-0.70446	0.63470	0.84325	0.84325
# OF UNIQUE OPERATORS	0.0074	0.0076	0.0271	0.0085	0.3708	0.0105	0.0266	0.0006	0.0006
N1	0.97111	0.99526	0.93063	0.99818	0.78183	-0.43545	0.28721	0.81631	0.81631
TOTAL # OF OPERATORS	0.0001	0.0001	0.0001	0.0001	0.0027	0.1571	0.3654	0.0012	0.0012
ETA2	0.82448	0.88148	0.99230	0.91211	0.95752	-0.06592	-0.08718	0.53309	0.53309
# OF UNIQUE OPERANDS	0.0010	0.0002	0.0001	0.0001	0.0001	0.8387	0.7876	0.0743	0.0743
N2	1.00000	0.98971	0.85035	0.97812	0.66401	-0.58079	0.44977	0.89612	0.89612
TOTAL # OF OPERANDS	0.0000	0.0001	0.0005	0.0001	0.0185	0.0477	0.1424	0.0001	0.0001
PLEN	0.98971	1.00000	0.90447	0.99703	0.73933	-0.49773	0.35546	0.85456	0.85456
PROGRAM LENGTH	0.0001	0.0000	0.0001	0.0001	0.0060	0.0996	0.2568	0.0004	0.0004
EPLEN	0.85035	0.90447	1.00000	0.93307	0.91896	-0.14725	0.00055	0.60135	0.60135
EST. PROGRAM LENGTH	0.0005	0.0001	0.0000	0.0001	0.0001	0.6479	0.9986	0.0386	0.0386
PVOL	0.97812	0.99703	0.93307	1.00000	0.77994	-0.43864	0.29523	0.82374	0.82374
PROGRAM VOLUME	0.0001	0.0001	0.0001	0.0000	0.0028	0.1537	0.3515	0.0010	0.0010

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2133

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	N2	PLEN	EPLEN	PVOL	EPPVOL	EPLEV	EPDIF	EFFORT	TIME
EPPVOL	0.66401	0.73933	0.91896	0.77994	1.00000	0.18843	-0.33586	0.29409	0.29409
EST. POTENTIAL PROGRAM VOLUME	0.0185	0.0060	0.0001	0.0028	0.0000	0.5576	0.2858	0.3535	0.3535
EPLEV	-0.58079	-0.49773	-0.14725	-0.43864	0.18843	1.00000	-0.96833	-0.83199	-0.83199
EST. PROGRAM LEVEL	0.0477	0.0996	0.6479	0.1537	0.5576	0.0000	0.0001	0.0008	0.0008
EPDIF	0.44977	0.35546	0.00055	0.29523	-0.33586	-0.96833	1.00000	0.76585	0.76585
EST. PROGRAM DIFFICULTY	0.1424	0.2568	0.9986	0.3515	0.2858	0.0001	0.0000	0.0037	0.0037
EFFORT	0.89612	0.85456	0.60135	0.82374	0.29409	-0.83199	0.76585	1.00000	1.00000
# OF ELEM. MENTAL DISCRIM.	0.0001	0.0004	0.0386	0.0010	0.3535	0.0008	0.0037	0.0000	0.0001
TIME	0.89612	0.85456	0.60135	0.82374	0.29409	-0.83199	0.76585	1.00000	1.00000
DEVELOPMENT TIME (HOURS)	0.0001	0.0004	0.0386	0.0010	0.3535	0.0008	0.0037	0.0001	0.0000
LAMBDA	0.27680	0.36867	0.67451	0.42600	0.89582	0.58354	-0.68003	-0.14155	-0.14155
LANGUAGE LEVEL	0.3838	0.2383	0.0161	0.1673	0.0001	0.0464	0.0150	0.6608	0.6608
MTSYSTIM	0.89505	0.91072	0.77499	0.90160	0.64464	-0.45995	0.32615	0.78725	0.78725
MEAN TOTAL SYSTEM TIME (SECS)	0.0001	0.0001	0.0031	0.0001	0.0236	0.1325	0.3009	0.0024	0.0024
MNSTMTS	0.40270	0.41465	0.34509	0.40717	0.44920	-0.01370	-0.03761	0.23265	0.23265
MEAN # OF SOURCE STMTS. EXEC.	0.1943	0.1802	0.2720	0.1889	0.1429	0.9663	0.9076	0.4668	0.4668
MTJOBTIM	0.89457	0.91026	0.77454	0.90109	0.64260	-0.46257	0.32858	0.78811	0.78811
MEAN TOTAL JOB TIME (SECS)	0.0001	0.0001	0.0031	0.0001	0.0242	0.1300	0.2970	0.0023	0.0023
MCOMPTIM	0.87338	0.91191	0.90624	0.92751	0.71430	-0.37930	0.25807	0.77559	0.77559
MEAN COMPILATION TIME (SECS)	0.0002	0.0001	0.0001	0.0001	0.0091	0.2240	0.4180	0.0030	0.0030
MEXECTIM	0.70343	0.69806	0.49451	0.67210	0.43892	-0.41897	0.30629	0.61506	0.61506
MEAN EXECUTION TIME (SECS)	0.0107	0.0116	0.1022	0.0167	0.1534	0.1752	0.3329	0.0333	0.0333
MOBJSIZE	0.93814	0.97311	0.93100	0.97956	0.77497	-0.42175	0.27854	0.79719	0.79719
MEAN OBJECT CODE SIZE (BYTES)	0.0001	0.0001	0.0001	0.0001	0.0031	0.1721	0.3807	0.0019	0.0019
MEXECMEM	0.15390	0.18664	0.25326	0.21405	0.05750	-0.05423	0.06584	0.27993	0.27993
MEAN EXECUTION MEMORY (BYTES)	0.6330	0.5614	0.4271	0.5041	0.8591	0.8671	0.8389	0.3782	0.3782
NCOMP	-0.77034	-0.77961	-0.73741	-0.77405	-0.61490	0.44512	-0.25324	-0.59775	-0.59775
# OF COMPILATIONS	0.0034	0.0028	0.0062	0.0031	0.0333	0.1471	0.4271	0.0401	0.0401

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2133

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=O / N = 12

	LAMBDA	MTSYSTIM	MNSTMTS	MTJOBTIM	MCOMPTIM	MEXECTIM	MOBJSIZE	MEXECMEM	NCOMP
LOC	0.67002	0.84472	0.62794	0.84304	0.80305	0.67814	0.91381	0.10453	-0.65237
# OF NBNC LINES OF CODE	0.0171	0.0005	0.0288	0.0006	0.0017	0.0154	0.0001	0.7465	0.0215
VG	0.22465	0.88532	0.49442	0.88630	0.84890	0.70947	0.89617	0.42413	-0.67756
MCCABE'S CYCLOMATIC COMPLEXITY	0.4827	0.0001	0.1023	0.0001	0.0005	0.0098	0.0001	0.1694	0.0155
NVAR	0.08812	0.60458	0.09969	0.60304	0.71052	0.38130	0.68624	0.24773	-0.50462
# OF VARIABLES	0.7854	0.0373	0.7579	0.0379	0.0096	0.2213	0.0137	0.4376	0.0943
NCONST	0.61247	0.17059	0.23299	0.16944	0.14687	0.14744	0.22858	-0.24543	-0.32385
# OF CONSTANTS	0.0343	0.5961	0.4662	0.5986	0.6488	0.6475	0.4749	0.4420	0.3045
NTYPE	0.49513	0.59228	0.11451	0.59006	0.68070	0.38411	0.68357	-0.06706	-0.71675
# OF TYPE DEFINITIONS	0.1017	0.0424	0.7231	0.0434	0.0148	0.2177	0.0142	0.8360	0.0087
NSUB	-0.16147	0.42911	-0.27716	0.43240	0.71197	0.11889	0.60413	0.55065	-0.44559
# OF SUBROUTINES	0.6161	0.1639	0.3831	0.1604	0.0094	0.7129	0.0375	0.0635	0.1466
ETA1	-0.09154	0.59445	-0.06143	0.59781	0.79226	0.31091	0.74231	0.47426	-0.59625
# OF UNIQUE OPERATORS	0.7772	0.0415	0.8496	0.0401	0.0021	0.3253	0.0057	0.1193	0.0407
N1	0.42677	0.91068	0.41791	0.91025	0.92739	0.68623	0.98546	0.20668	-0.77677
TOTAL # OF OPERATORS	0.1665	0.0001	0.1764	0.0001	0.0001	0.0137	0.0001	0.5192	0.0030
ETA2	0.74356	0.76094	0.39349	0.75992	0.86311	0.50503	0.90348	0.18676	-0.72519
# OF UNIQUE OPERANDS	0.0056	0.0040	0.2057	0.0041	0.0003	0.0940	0.0001	0.5611	0.0076
N2	0.27680	0.89505	0.40270	0.89457	0.87338	0.70343	0.93814	0.15390	-0.77034
TOTAL # OF OPERANDS	0.3838	0.0001	0.1943	0.0001	0.0002	0.0107	0.0001	0.6330	0.0034
PLEN	0.36867	0.91072	0.41465	0.91026	0.91191	0.69806	0.97311	0.18664	-0.77961
PROGRAM LENGTH	0.2383	0.0001	0.1802	0.0001	0.0001	0.0116	0.0001	0.5614	0.0028
EPLEN	0.67451	0.77499	0.34509	0.77454	0.90624	0.49451	0.93100	0.25326	-0.73741
EST. PROGRAM LENGTH	0.0161	0.0031	0.2720	0.0031	0.0001	0.1022	0.0001	0.4271	0.0062
PVOL	0.42600	0.90160	0.40717	0.90109	0.92751	0.67210	0.97956	0.21405	-0.77405
PROGRAM VOLUME	0.1673	0.0001	0.1889	0.0001	0.0001	0.0167	0.0001	0.5041	0.0031

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 2133

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 12

	LAMBDA	MTSYSTIM	MNSTMTS	MTJOBTIM	MCOMPTIM	MEXECTIM	MOBJSIZE	MEXECMEM	NCOMP
EPPVOL	0.89582	0.64464	0.44920	0.64260	0.71430	0.43892	0.77497	0.05750	-0.61490
EST. POTENTIAL PROGRAM VOLUME	0.0001	0.0236	0.1429	0.0242	0.0091	0.1534	0.0031	0.8591	0.0333
EPLEV	0.58354	-0.45995	-0.01370	-0.46257	-0.37930	-0.41897	-0.42175	-0.05423	0.44512
EST. PROGRAM LEVEL	0.0464	0.1325	0.9663	0.1300	0.2240	0.1752	0.1721	0.8671	0.1471
EPDIF	-0.68003	0.32615	-0.03761	0.32858	0.25807	0.30629	0.27854	0.06584	-0.25324
EST. PROGRAM DIFFICULTY	0.0150	0.3009	0.9076	0.2970	0.4180	0.3329	0.3807	0.8389	0.4271
EFFORT	-0.14155	0.78725	0.23265	0.78811	0.77559	0.61506	0.79719	0.27993	-0.59775
# OF ELEM. MENTAL DISCRIM.	0.6608	0.0024	0.4668	0.0023	0.0030	0.0333	0.0019	0.3782	0.0401
TIME	-0.14155	0.78725	0.23265	0.78811	0.77559	0.61506	0.79719	0.27993	-0.59775
DEVELOPMENT TIME (HOURS)	0.6608	0.0024	0.4668	0.0023	0.0030	0.0333	0.0019	0.3782	0.0401
LAMBDA	1.00000	0.29558	0.37210	0.29286	0.38755	0.15266	0.43128	-0.04742	-0.32070
LANGUAGE LEVEL	0.0000	0.3509	0.2336	0.3556	0.2132	0.6358	0.1616	0.8837	0.3095
MTSYSTIM	0.29558	1.00000	0.64544	0.99997	0.82390	0.90264	0.90796	0.26022	-0.65348
MEAN TOTAL SYSTEM TIME (SECS)	0.3509	0.0000	0.0234	0.0001	0.0010	0.0001	0.0001	0.4140	0.0212
MNSTMTS	0.37210	0.64544	1.00000	0.64287	0.22337	0.81403	0.42224	-0.05692	-0.14480
MEAN # OF SOURCE STMTS. EXEC.	0.2336	0.0234	0.0000	0.0241	0.4853	0.0013	0.1715	0.8605	0.6534
MTJOBTIM	0.29286	0.99997	0.64287	1.00000	0.82473	0.90205	0.90816	0.26251	-0.65501
MEAN TOTAL JOB TIME (SECS)	0.3556	0.0001	0.0241	0.0000	0.0010	0.0001	0.0001	0.4098	0.0208
MCOMPTIM	0.38755	0.82390	0.22337	0.82473	1.00000	0.49984	0.95135	0.47641	-0.74176
MEAN COMPILATION TIME (SECS)	0.2132	0.0010	0.4853	0.0010	0.0000	0.0980	0.0001	0.1174	0.0058
MEXECTIM	0.15266	0.90264	0.81403	0.90205	0.49984	1.00000	0.66472	0.03850	-0.43700
MEAN EXECUTION TIME (SECS)	0.6358	0.0001	0.0013	0.0001	0.0980	0.0000	0.0184	0.9054	0.1555
MOBJSIZE	0.43128	0.90796	0.42224	0.90816	0.95135	0.66472	1.00000	0.26432	-0.77038
MEAN OBJECT CODE SIZE (BYTES)	0.1616	0.0001	0.1715	0.0001	0.0001	0.0184	0.0000	0.4064	0.0034
MEXECMEM	-0.04742	0.26022	-0.05692	0.26251	0.47641	0.03850	0.26432	1.00000	-0.04456
MEAN EXECUTION MEMORY (BYTES)	0.8837	0.4140	0.8605	0.4098	0.1174	0.9054	0.4064	0.0000	0.8906
NCOMP	-0.32070	-0.65348	-0.14480	-0.65501	-0.74176	-0.43700	-0.77038	-0.04456	1.00000
# OF COMPILATIONS	0.3095	0.0212	0.6534	0.0208	0.0058	0.1555	0.0034	0.8906	0.0000

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 4323

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
LOC	7	838.57143	145.286908	5870.00000	694.00000	1146.00000
VG	7	132.28571	29.792137	926.00000	103.00000	195.00000
NVAR	7	90.71429	16.245146	635.00000	71.00000	122.00000
NCONST	7	11.00000	14.329457	77.00000	2.00000	42.00000
NTYPE	7	3.28571	2.429972	23.00000	0.00000	6.00000
NSUB	7	23.42857	6.528327	164.00000	16.00000	36.00000
ETA1	7	68.57143	5.563486	480.00000	63.00000	79.00000
N1	7	2656.14286	356.127519	18593.00000	2222.00000	3201.00000
ETA2	7	177.57143	17.472427	1243.00000	156.00000	207.00000
N2	7	1945.57143	323.248541	13619.00000	1570.00000	2448.00000
PLEN	7	4601.71429	673.656123	32212.00000	3792.00000	5649.00000
EPLEN	7	1746.44000	139.066402	12225.08000	1554.20000	1969.12000
PVOL	7	36557.85000	5503.091658	255904.95000	29867.75000	44867.26000
EPPVOL	7	98.49857	15.994097	689.49000	75.13000	124.53000
EPLEV	7	2.714285714E-03	3.716116765E-04	1.900000000E-02	2.100000000E-03	3.200000000E-03
EPDIF	7	375.31914	56.566970	2627.23400	309.23800	477.15300
EFFORT	7	13838202.85714	3736788.308404	96867420.00000	10550807.00000	21408529.00000
TIME	7	213.55251	57.666489	1494.86759	162.82109	330.37854
LAMBDA	7	2.714285714E-01	7.128079953E-02	1.90000	1.900000000E-01	3.600000000E-01
MUTIME	7	1.96190	9.458346444E-01	13.73333	3.000000000E-01	3.20000
MSYSTIME	7	3.714285714E-01	3.587161115E-01	2.60000	1.333333333E-01	1.16667
MELAPSED	7	1.90476	9.946949228E-01	13.33333	0.00000	3.00000
MEXECTIM	7	2.33333	9.934973765E-01	16.33333	4.333333333E-01	3.56667
MOBJSIZE	7	22674.28571	1244.145452	158720.00000	21504.00000	24576.00000
MEXECMEM	7	110396.95238	22578.675088	772778.66667	81578.66667	141312.00000
NCOMP	7	84.42857	79.344937	591.00000	12.00000	204.00000

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 4323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 7

	LOC	VG	NVAR	NCONST	NTYPE	NSUB	ETA1	N1	ETA2
LOC	1.00000	0.97537	0.69882	-0.12809	0.38421	0.17524	-0.05779	0.76684	0.12000
# OF NBNC LINES OF CODE	0.0000	0.0002	0.0806	0.7843	0.3948	0.7070	0.9021	0.0443	0.7977
VG	0.97537	1.00000	0.62419	-0.14679	0.32560	0.17579	-0.05645	0.70401	0.04126
MCCABE'S CYCLOMATIC COMPLEXITY	0.0002	0.0000	0.1341	0.7535	0.4761	0.7062	0.9043	0.0775	0.9300
NVAR	0.69882	0.62419	1.00000	-0.43030	0.72438	-0.05366	-0.07719	0.64131	0.11341
# OF VARIABLES	0.0806	0.1341	0.0000	0.3352	0.0656	0.9090	0.8694	0.1206	0.8087
NCONST	-0.12809	-0.14679	-0.43030	1.00000	-0.78020	0.86587	0.84460	-0.57060	-0.42071
# OF CONSTANTS	0.7843	0.7535	0.3352	0.0000	0.0385	0.0118	0.0168	0.1810	0.3473
NTYPE	0.38421	0.32560	0.72438	-0.78020	1.00000	-0.62887	-0.63050	0.72390	0.18394
# OF TYPE DEFINITIONS	0.3948	0.4761	0.0656	0.0385	0.0000	0.1303	0.1290	0.0659	0.6930
NSUB	0.17524	0.17579	-0.05366	0.86587	-0.62887	1.00000	0.92825	-0.45675	-0.55043
# OF SUBROUTINES	0.7070	0.7062	0.9090	0.0118	0.1303	0.0000	0.0025	0.3029	0.2004
ETA1	-0.05779	-0.05645	-0.07719	0.84460	-0.63050	0.92825	1.00000	-0.60596	-0.51657
# OF UNIQUE OPERATORS	0.9021	0.9043	0.8694	0.0168	0.1290	0.0025	0.0000	0.1492	0.2352
N1	0.76684	0.70401	0.64131	-0.57060	0.72390	-0.45675	-0.60596	1.00000	0.54238
TOTAL # OF OPERATORS	0.0443	0.0775	0.1206	0.1810	0.0659	0.3029	0.1492	0.0000	0.2085
ETA2	0.12000	0.04126	0.11341	-0.42071	0.18394	-0.55043	-0.51657	0.54238	1.00000
# OF UNIQUE OPERANDS	0.7977	0.9300	0.8087	0.3473	0.6930	0.2004	0.2352	0.2085	0.0000
N2	0.72245	0.65348	0.71568	-0.61504	0.85931	-0.47914	-0.60733	0.96639	0.37925
TOTAL # OF OPERANDS	0.0667	0.1114	0.0705	0.1416	0.0132	0.2766	0.1481	0.0004	0.4015
PLEN	0.75205	0.68574	0.68244	-0.59677	0.79503	-0.47137	-0.61176	0.99236	0.46871
PROGRAM LENGTH	0.0512	0.0890	0.0912	0.1572	0.0326	0.2856	0.1443	0.0001	0.2887
EPLEN	0.11432	0.02523	0.10007	-0.21146	0.01450	-0.33606	-0.27583	0.42425	0.96546
EST. PROGRAM LENGTH	0.8072	0.9572	0.8310	0.6490	0.9754	0.4612	0.5494	0.3428	0.0004
PVOL	0.74075	0.66833	0.67129	-0.59049	0.77168	-0.47911	-0.61103	0.99635	0.52897
PROGRAM VOLUME	0.0568	0.1008	0.0987	0.1628	0.0421	0.2767	0.1449	0.0001	0.2222

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 4323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=O / N = 7

	LOC	VG	NVAR	NCONST	NTYPE	NSUB	ETA1	N1	ETA2
EPPVOL	0.04796	-0.00472	-0.02685	-0.52248	0.21722	-0.69699	-0.70539	0.54481	0.95722
EST. POTENTIAL PROGRAM VOLUME	0.9187	0.9920	0.9544	0.2289	0.6399	0.0818	0.0766	0.2060	0.0007
EPLEV	-0.63301	-0.60711	-0.63144	-0.06260	-0.48515	-0.31209	-0.19808	-0.35491	0.52988
EST. PROGRAM LEVEL	0.1270	0.1483	0.1283	0.8939	0.2698	0.4956	0.6703	0.4347	0.2212
EPDIF	0.69994	0.68505	0.67312	-0.00121	0.51230	0.27971	0.15124	0.42049	-0.48247
EST. PROGRAM DIFFICULTY	0.0800	0.0894	0.0975	0.9979	0.2398	0.5435	0.7462	0.3476	0.2728
EFFORT	0.87030	0.82705	0.81653	-0.35694	0.74451	-0.09696	-0.24896	0.83008	0.02497
# OF ELEM. MENTAL DISCRIM.	0.0108	0.0217	0.0250	0.4319	0.0549	0.8362	0.5903	0.0208	0.9576
TIME	0.87030	0.82705	0.81653	-0.35694	0.74451	-0.09696	-0.24896	0.83008	0.02497
DEVELOPMENT TIME (HOURS)	0.0108	0.0217	0.0250	0.4319	0.0549	0.8362	0.5903	0.0208	0.9576
LAMBDA	-0.28060	-0.29532	-0.35798	-0.31982	-0.14708	-0.56026	-0.51093	0.14949	0.85703
LANGUAGE LEVEL	0.5422	0.5202	0.4305	0.4844	0.7530	0.1908	0.2413	0.7490	0.0137
MUTIME	-0.70998	-0.65766	-0.70986	-0.05042	-0.45858	-0.37210	-0.20527	-0.38267	0.46377
MEAN USER TIME (SECS)	0.0739	0.1084	0.0739	0.9145	0.3007	0.4111	0.6588	0.3969	0.2945
MSYSTIME	-0.22935	-0.26423	-0.43064	0.98569	-0.74114	0.83879	0.84467	-0.63706	-0.46763
MEAN SYSTEM TIME (SECS)	0.6208	0.5669	0.3348	0.0001	0.0566	0.0183	0.0168	0.1239	0.2900
MELAPSED	-0.80762	-0.76006	-0.79616	0.21827	-0.60745	-0.14666	0.06167	-0.58588	0.25938
MEAN ELAPSED TIME (H:MM:SS)	0.0280	0.0474	0.0322	0.6382	0.1480	0.7537	0.8955	0.1669	0.5743
MEXECTIM	-0.75873	-0.72151	-0.83129	0.30790	-0.70418	-0.05139	0.10956	-0.59434	0.27268
MEAN EXECUTION TIME (SECS)	0.0480	0.0672	0.0205	0.5017	0.0774	0.9129	0.8151	0.1593	0.5541
MOBJSIZE	-0.37538	-0.47097	-0.05670	-0.54566	0.54839	-0.80748	-0.72913	0.24993	0.37236
MEAN OBJECT CODE SIZE (BYTES)	0.4067	0.2861	0.9039	0.2052	0.2025	0.0281	0.0630	0.5888	0.4108
MEXECMEM	-0.60080	-0.68249	-0.12255	-0.50446	0.38587	-0.76390	-0.56914	0.01364	0.41535
MEAN EXECUTION MEMORY (BYTES)	0.1537	0.0911	0.7935	0.2483	0.3926	0.0456	0.1824	0.9769	0.3541
NCOMP	-0.21877	-0.29019	0.33242	-0.37937	0.19548	-0.15968	-0.07842	-0.13533	0.14586
# OF COMPILATIONS	0.6374	0.5278	0.4663	0.4013	0.6744	0.7324	0.8673	0.7724	0.7550

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 4323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 7

	N2	PLEN	EPLEN	PVOL	EPPVOL	EPLEV	EPDIF	EFFORT	TIME
LOC	0.72245	0.75205	0.11432	0.74075	0.04796	-0.63301	0.69994	0.87030	0.87030
# OF NBNC LINES OF CODE	0.0667	0.0512	0.8072	0.0568	0.9187	0.1270	0.0800	0.0108	0.0108
VG	0.65348	0.68574	0.02523	0.66833	-0.00472	-0.60711	0.68505	0.82705	0.82705
MCCABE'S CYCLOMATIC COMPLEXITY	0.1114	0.0890	0.9572	0.1008	0.9920	0.1483	0.0894	0.0217	0.0217
NVAR	0.71568	0.68244	0.10007	0.67129	-0.02685	-0.63144	0.67312	0.81653	0.81653
# OF VARIABLES	0.0705	0.0912	0.8310	0.0987	0.9544	0.1283	0.0975	0.0250	0.0250
NCONST	-0.61504	-0.59677	-0.21146	-0.59049	-0.52248	-0.06260	-0.00121	-0.35694	-0.35694
# OF CONSTANTS	0.1416	0.1572	0.6490	0.1628	0.2289	0.8939	0.9979	0.4319	0.4319
NTYPE	0.85931	0.79503	0.01450	0.77168	0.21722	-0.48515	0.51230	0.74451	0.74451
# OF TYPE DEFINITIONS	0.0132	0.0326	0.9754	0.0421	0.6399	0.2698	0.2398	0.0549	0.0549
NSUB	-0.47914	-0.47137	-0.33606	-0.47911	-0.69699	-0.31209	0.27971	-0.09696	-0.09696
# OF SUBROUTINES	0.2766	0.2856	0.4612	0.2767	0.0818	0.4956	0.5435	0.8362	0.8362
ETA1	-0.60733	-0.61176	-0.27583	-0.61103	-0.70539	-0.19808	0.15124	-0.24896	-0.24896
# OF UNIQUE OPERATORS	0.1481	0.1443	0.5494	0.1449	0.0766	0.6703	0.7462	0.5903	0.5903
N1	0.96639	0.99236	0.42425	0.99635	0.54481	-0.35491	0.42049	0.83008	0.83008
TOTAL # OF OPERATORS	0.0004	0.0001	0.3428	0.0001	0.2060	0.4347	0.3476	0.0208	0.0208
ETA2	0.37925	0.46871	0.96546	0.52897	0.95722	0.52988	-0.48247	0.02497	0.02497
# OF UNIQUE OPERANDS	0.4015	0.2887	0.0004	0.2222	0.0007	0.2212	0.2728	0.9576	0.9576
N2	1.00000	0.99072	0.24123	0.98067	0.39318	-0.50664	0.55767	0.89586	0.89586
TOTAL # OF OPERANDS	0.0000	0.0001	0.6023	0.0001	0.3829	0.2459	0.1933	0.0064	0.0064
PLEN	0.99072	1.00000	0.34003	0.99729	0.47668	-0.43073	0.48988	0.86869	0.86869
PROGRAM LENGTH	0.0001	0.0000	0.4555	0.0001	0.2795	0.3347	0.2644	0.0112	0.0112
EPLEN	0.24123	0.34003	1.00000	0.40796	0.86036	0.53378	-0.49569	-0.04845	-0.04845
EST. PROGRAM LENGTH	0.6023	0.4555	0.0000	0.3636	0.0130	0.2172	0.2579	0.9178	0.9178
PVOL	0.98067	0.99729	0.40796	1.00000	0.52734	-0.37990	0.43978	0.84102	0.84102
PROGRAM VOLUME	0.0001	0.0001	0.3636	0.0000	0.2239	0.4006	0.3235	0.0177	0.0177

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 4323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 7

	N2	PLEN	EPLEN	PVOL	EPPVOL	EPLEV	EPDIF	EFFORT	TIME
EPPVOL	0.39318	0.47668	0.86036	0.52734	1.00000	0.57945	-0.52956	-0.01200	-0.01200
EST. POTENTIAL PROGRAM VOLUME	0.3829	0.2795	0.0130	0.2239	0.0000	0.1728	0.2216	0.9796	0.9796
EPLEV	-0.50664	-0.43073	0.53378	-0.37990	0.57945	1.00000	-0.99341	-0.80656	-0.80656
EST. PROGRAM LEVEL	0.2459	0.3347	0.2172	0.4006	0.1728	0.0000	0.0001	0.0284	0.0284
EPDIF	0.55767	0.48988	-0.49569	0.43978	-0.52956	-0.99341	1.00000	0.85129	0.85129
EST. PROGRAM DIFFICULTY	0.1933	0.2644	0.2579	0.3235	0.2216	0.0001	0.0000	0.0151	0.0151
EFFORT	0.89586	0.86869	-0.04845	0.84102	-0.01200	-0.80656	0.85129	1.00000	1.00000
# OF ELEM. MENTAL DISCRIM.	0.0064	0.0112	0.9178	0.0177	0.9796	0.0284	0.0151	0.0000	0.0001
TIME	0.89586	0.86869	-0.04845	0.84102	-0.01200	-0.80656	0.85129	1.00000	1.00000
DEVELOPMENT TIME (HOURS)	0.0064	0.0112	0.9178	0.0177	0.9796	0.0284	0.0151	0.0001	0.0000
LAMBDA	-0.02963	0.06481	0.80655	0.12265	0.90584	0.86739	-0.83241	-0.42277	-0.42277
LANGUAGE LEVEL	0.9497	0.8902	0.0284	0.7933	0.0050	0.0114	0.0201	0.3447	0.3447
MUTIME	-0.51279	-0.44836	0.45881	-0.40299	0.52885	0.95491	-0.95601	-0.80458	-0.80458
MEAN USER TIME (SECS)	0.2392	0.3130	0.3004	0.3700	0.2223	0.0008	0.0008	0.0291	0.0291
MSYSTIME	-0.64932	-0.64836	-0.26360	-0.64449	-0.56678	-0.06311	-0.01456	-0.40199	-0.40199
MEAN SYSTEM TIME (SECS)	0.1145	0.1152	0.5679	0.1181	0.1846	0.8931	0.9753	0.3713	0.3713
MELAPSED	-0.68869	-0.64018	0.31222	-0.59901	0.28892	0.86098	-0.89091	-0.88697	-0.88697
MEAN ELAPSED TIME (H:MM:SS)	0.0871	0.1214	0.4954	0.1552	0.5297	0.0128	0.0071	0.0078	0.0078
MEXECTIM	-0.72264	-0.66095	0.34162	-0.61636	0.29884	0.88631	-0.91540	-0.91113	-0.91113
MEAN EXECUTION TIME (SECS)	0.0666	0.1060	0.4533	0.1405	0.5150	0.0079	0.0038	0.0043	0.0043
MOBJSIZE	0.33883	0.29471	0.20132	0.30054	0.49995	0.21621	-0.24873	-0.02201	-0.02201
MEAN OBJECT CODE SIZE (BYTES)	0.4572	0.5211	0.6651	0.5125	0.2532	0.6415	0.5907	0.9626	0.9626
MEXECMEM	0.08485	0.04792	0.29682	0.06770	0.48811	0.43906	-0.47558	-0.27882	-0.27882
MEAN EXECUTION MEMORY (BYTES)	0.8565	0.9187	0.5180	0.8853	0.2664	0.3243	0.2807	0.5448	0.5448
NCOMP	-0.08584	-0.11273	0.13480	-0.10080	0.08800	0.24790	-0.24600	-0.18276	-0.18276
# OF COMPILATIONS	0.8548	0.8098	0.7732	0.8297	0.8512	0.5920	0.5949	0.6949	0.6949

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 4323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=O / N = 7

	LAMBDA	MUTIME	MSYSTIME	MELAPSED	MEXECTIM	MOBJSIZE	MEXECMEM	NCOMP
LOC	-0.28060	-0.70998	-0.22935	-0.80762	-0.75873	-0.37538	-0.60080	-0.21877
# OF NBNC LINES OF CODE	0.5422	0.0739	0.6208	0.0280	0.0480	0.4067	0.1537	0.6374
VG	-0.29532	-0.65766	-0.26423	-0.76006	-0.72151	-0.47097	-0.68249	-0.29019
MCCABE'S CYCLOMATIC COMPLEXITY	0.5202	0.1084	0.5669	0.0474	0.0672	0.2861	0.0911	0.5278
NVAR	-0.35798	-0.70986	-0.43064	-0.79616	-0.83129	-0.05670	-0.12255	0.33242
# OF VARIABLES	0.4305	0.0739	0.3348	0.0322	0.0205	0.9039	0.7935	0.4663
NCONST	-0.31982	-0.05042	0.98569	0.21827	0.30790	-0.54566	-0.50446	-0.37937
# OF CONSTANTS	0.4844	0.9145	0.0001	0.6382	0.5017	0.2052	0.2483	0.4013
NTYPE	-0.14708	-0.45858	-0.74114	-0.60745	-0.70418	0.54839	0.38587	0.19548
# OF TYPE DEFINITIONS	0.7530	0.3007	0.0566	0.1480	0.0774	0.2025	0.3926	0.6744
NSUB	-0.56026	-0.37210	0.83879	-0.14666	-0.05139	-0.80748	-0.76390	-0.15968
# OF SUBROUTINES	0.1908	0.4111	0.0183	0.7537	0.9129	0.0281	0.0456	0.7324
ETA1	-0.51093	-0.20527	0.84467	0.06167	0.10956	-0.72913	-0.56914	-0.07842
# OF UNIQUE OPERATORS	0.2413	0.6588	0.0168	0.8955	0.8151	0.0630	0.1824	0.8673
N1	0.14949	-0.38267	-0.63706	-0.58588	-0.59434	0.24993	0.01364	-0.13533
TOTAL # OF OPERATORS	0.7490	0.3969	0.1239	0.1669	0.1593	0.5888	0.9769	0.7724
ETA2	0.85703	0.46377	-0.46763	0.25938	0.27268	0.37236	0.41535	0.14586
# OF UNIQUE OPERANDS	0.0137	0.2945	0.2900	0.5743	0.5541	0.4108	0.3541	0.7550
N2	-0.02963	-0.51279	-0.64932	-0.68869	-0.72264	0.33883	0.08485	-0.08584
TOTAL # OF OPERANDS	0.9497	0.2392	0.1145	0.0871	0.0666	0.4572	0.8565	0.8548
PLEN	0.06481	-0.44836	-0.64836	-0.64018	-0.66095	0.29471	0.04792	-0.11273
PROGRAM LENGTH	0.8902	0.3130	0.1152	0.1214	0.1060	0.5211	0.9187	0.8098
EPLEN	0.80655	0.45881	-0.26360	0.31222	0.34162	0.20132	0.29682	0.13480
EST. PROGRAM LENGTH	0.0284	0.3004	0.5679	0.4954	0.4533	0.6651	0.5180	0.7732
PVOL	0.12265	-0.40299	-0.64449	-0.59901	-0.61636	0.30054	0.06770	-0.10080
PROGRAM VOLUME	0.7933	0.3700	0.1181	0.1552	0.1405	0.5125	0.8853	0.8297

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 4323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 7

	LAMBDA	MUTIME	MSYSTIME	MELAPSED	MEXECTIM	MOBJSIZE	MEXECMEM	NCOMP
EPPVOL	0.90584	0.52885	-0.56678	0.28892	0.29884	0.49995	0.48811	0.08800
EST. POTENTIAL PROGRAM VOLUME	0.0050	0.2223	0.1846	0.5297	0.5150	0.2532	0.2664	0.8512
EPLEV	0.86739	0.95491	-0.06311	0.86098	0.88631	0.21621	0.43906	0.24790
EST. PROGRAM LEVEL	0.0114	0.0008	0.8931	0.0128	0.0079	0.6415	0.3243	0.5920
EPDIF	-0.83241	-0.95601	-0.01456	-0.89091	-0.91540	-0.24873	-0.47558	-0.24600
EST. PROGRAM DIFFICULTY	0.0201	0.0008	0.9753	0.0071	0.0038	0.5907	0.2807	0.5949
EFFORT	-0.42277	-0.80458	-0.40199	-0.88697	-0.91113	-0.02201	-0.27882	-0.18276
# OF ELEM. MENTAL DISCRIM.	0.3447	0.0291	0.3713	0.0078	0.0043	0.9626	0.5448	0.6949
TIME	-0.42277	-0.80458	-0.40199	-0.88697	-0.91113	-0.02201	-0.27882	-0.18276
DEVELOPMENT TIME (HOURS)	0.3447	0.0291	0.3713	0.0078	0.0043	0.9626	0.5448	0.6949
LAMBDA	1.00000	0.81837	-0.35446	0.62908	0.65113	0.38214	0.48764	0.13838
LANGUAGE LEVEL	0.0000	0.0244	0.4353	0.1301	0.1132	0.3976	0.2670	0.7673
MUTIME	0.81837	1.00000	-0.05341	0.94621	0.93274	0.25691	0.47803	0.04571
MEAN USER TIME (SECS)	0.0244	0.0000	0.9095	0.0013	0.0022	0.5781	0.2779	0.9225
MSYSTIME	-0.35446	-0.05341	1.00000	0.22984	0.31022	-0.44796	-0.39297	-0.28252
MEAN SYSTEM TIME (SECS)	0.4353	0.9095	0.0000	0.6200	0.4983	0.3135	0.3832	0.5393
MELAPSED	0.62908	0.94621	0.22984	1.00000	0.98380	0.15104	0.39756	-0.06345
MEAN ELAPSED TIME (H:MM:SS)	0.1301	0.0013	0.6200	0.0000	0.0001	0.7465	0.3771	0.8925
MEXECTIM	0.65113	0.93274	0.31022	0.98380	1.00000	0.08284	0.31321	-0.05850
MEAN EXECUTION TIME (SECS)	0.1132	0.0022	0.4983	0.0001	0.0000	0.8598	0.4940	0.9009
MOBJSIZE	0.38214	0.25691	-0.44796	0.15104	0.08284	1.00000	0.92193	0.21882
MEAN OBJECT CODE SIZE (BYTES)	0.3976	0.5781	0.3135	0.7465	0.8598	0.0000	0.0031	0.6373
MEXECMEM	0.48764	0.47803	-0.39297	0.39756	0.31321	0.92193	1.00000	0.39820
MEAN EXECUTION MEMORY (BYTES)	0.2670	0.2779	0.3832	0.3771	0.4940	0.0031	0.0000	0.3763
NCOMP	0.13838	0.04571	-0.28252	-0.06345	-0.05850	0.21882	0.39820	1.00000
# OF COMPILATIONS	0.7673	0.9225	0.5393	0.8925	0.9009	0.6373	0.3763	0.0000

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 5323

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
LOC	10	2127.60000	491.74343	21276.0000	1381.00000	2576.0000
VG	10	426.10000	94.92506	4261.0000	302.00000	587.0000
NVAR	10	161.10000	72.49437	1611.0000	46.00000	243.0000
NCONST	10	34.60000	44.11651	346.0000	0.00000	132.0000
NTYPE	10	5.30000	10.15491	53.0000	0.00000	32.0000
NSUB	10	68.30000	27.24396	683.0000	25.00000	97.0000
ETA1	10	120.30000	30.18112	1203.0000	71.00000	158.0000
N1	10	7097.90000	1172.59659	70979.0000	5327.00000	8740.0000
ETA2	10	452.90000	139.28025	4529.0000	290.00000	671.0000
N2	10	4554.50000	513.25026	45545.0000	3913.00000	5416.0000
PLEN	10	11652.40000	1616.31565	116524.0000	9240.00000	14146.0000
EPLEN	10	4859.99600	1652.82045	48599.9600	2870.17000	7307.4800
PVOL	10	106506.69000	18159.91414	1065066.9000	78937.61000	133903.5600
EPPVOL	10	176.23700	37.36621	1762.3700	137.17000	249.2100
EPLEV	10	1.680000000E-03	3.084008935E-04	1.680000000E-02	1.200000000E-03	2.200000000E-03
EPDIF	10	617.32900	116.87230	6173.2900	459.97200	832.7420
EFFORT	10	66411808.00000	19728363.07595	664118080.0000	36309081.00000	107451348.0000
TIME	10	1024.87358	304.45005	10248.7358	560.32532	1658.1998
LAMBDA	10	3.010000000E-01	1.017021796E-01	3.0100	1.900000000E-01	4.600000000E-01
MUTIME	10	70.58000	28.96757	705.8000	41.93333	123.9667
MSYSTIME	10	32.28667	73.33602	322.8667	1.10000	234.2333
MELAPSED	10	119.73333	87.93457	1197.3333	55.66667	351.6667
MEXECTIM	10	102.86667	89.45956	1028.6667	45.56667	336.6667
MOBJSIZE	10	45977.60000	8614.18762	459776.0000	35840.00000	57344.0000
MEXECMEM	10	225894.40000	48156.31507	2258944.0000	162816.00000	299690.6667
NCOMP	10	586.10000	458.38520	5861.0000	136.00000	1525.0000

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 5323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 10

	LOC	VG	NVAR	NCONST	NTYPE	NSUB	ETA1	N1	ETA2
LOC	1.00000	0.69927	0.85834	0.53722	-0.02269	0.86923	0.85601	0.84728	0.86882
# OF NBNC LINES OF CODE	0.0000	0.0244	0.0015	0.1093	0.9504	0.0011	0.0016	0.0020	0.0011
VG	0.69927	1.00000	0.76947	0.51341	0.14047	0.67611	0.62963	0.77014	0.66707
MCCABE'S CYCLOMATIC COMPLEXITY	0.0244	0.0000	0.0093	0.1291	0.6987	0.0318	0.0511	0.0092	0.0351
NVAR	0.85834	0.76947	1.00000	0.58889	0.21790	0.82765	0.84491	0.80386	0.84650
# OF VARIABLES	0.0015	0.0093	0.0000	0.0733	0.5453	0.0031	0.0021	0.0051	0.0020
NCONST	0.53722	0.51341	0.58889	1.00000	-0.29732	0.55719	0.56889	0.46665	0.86334
# OF CONSTANTS	0.1093	0.1291	0.0733	0.0000	0.4041	0.0943	0.0861	0.1740	0.0013
NTYPE	-0.02269	0.14047	0.21790	-0.29732	1.00000	0.21691	0.21647	-0.00020	-0.08835
# OF TYPE DEFINITIONS	0.9504	0.6987	0.5453	0.4041	0.0000	0.5472	0.5480	0.9996	0.8082
NSUB	0.86923	0.67611	0.82765	0.55719	0.21691	1.00000	0.99146	0.72392	0.85173
# OF SUBROUTINES	0.0011	0.0318	0.0031	0.0943	0.5472	0.0000	0.0001	0.0179	0.0018
ETA1	0.85601	0.62963	0.84491	0.56889	0.21647	0.99146	1.00000	0.73479	0.85517
# OF UNIQUE OPERATORS	0.0016	0.0511	0.0021	0.0861	0.5480	0.0001	0.0000	0.0155	0.0016
N1	0.84728	0.77014	0.80386	0.46665	-0.00020	0.72392	0.73479	1.00000	0.70359
TOTAL # OF OPERATORS	0.0020	0.0092	0.0051	0.1740	0.9996	0.0179	0.0155	0.0000	0.0232
ETA2	0.86882	0.66707	0.84650	0.86334	-0.08835	0.85173	0.85517	0.70359	1.00000
# OF UNIQUE OPERANDS	0.0011	0.0351	0.0020	0.0013	0.8082	0.0018	0.0016	0.0232	0.0000
N2	0.47193	0.50208	0.37104	0.08287	-0.01608	0.25496	0.26764	0.80925	0.23987
TOTAL # OF OPERANDS	0.1685	0.1392	0.2912	0.8200	0.9648	0.4771	0.4547	0.0046	0.5044
PLEN	0.76454	0.71815	0.70100	0.36486	-0.00525	0.60615	0.61806	0.98245	0.58661
PROGRAM LENGTH	0.0100	0.0193	0.0239	0.2999	0.9885	0.0632	0.0569	0.0001	0.0747
EPLEN	0.87762	0.66847	0.85899	0.84247	-0.05033	0.88258	0.88798	0.71755	0.99767
EST. PROGRAM LENGTH	0.0008	0.0346	0.0015	0.0022	0.8902	0.0007	0.0006	0.0195	0.0001
PVOL	0.86179	0.77849	0.80641	0.51478	-0.00748	0.73569	0.74407	0.99616	0.74157
PROGRAM VOLUME	0.0013	0.0080	0.0048	0.1279	0.9836	0.0153	0.0136	0.0001	0.0141

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 5323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=O / N = 10

	LOC	VG	NVAR	NCONST	NTYPE	NSUB	ETA1	N1	ETA2
EPPVOL	0.68378	0.55850	0.65809	0.89211	-0.33451	0.51223	0.50452	0.50680	0.87539
EST. POTENTIAL PROGRAM VOLUME	0.0292	0.0933	0.0386	0.0005	0.3448	0.1301	0.1370	0.1349	0.0009
EPLEV	-0.08278	-0.19463	-0.08439	0.45749	-0.39878	-0.19096	-0.20699	-0.40994	0.21465
EST. PROGRAM LEVEL	0.8202	0.5900	0.8167	0.1837	0.2537	0.5972	0.5661	0.2394	0.5515
EPDIF	0.06254	0.10469	0.04513	-0.49071	0.40074	0.14135	0.16339	0.39762	-0.25730
EST. PROGRAM DIFFICULTY	0.8637	0.7735	0.9015	0.1498	0.2511	0.6969	0.6520	0.2552	0.4730
EFFORT	0.51880	0.46156	0.45788	-0.08260	0.24022	0.47538	0.49412	0.80346	0.20964
# OF ELEM. MENTAL DISCRIM.	0.1244	0.1793	0.1833	0.8205	0.5038	0.1650	0.1466	0.0051	0.5610
TIME	0.51880	0.46156	0.45788	-0.08260	0.24022	0.47538	0.49412	0.80346	0.20964
DEVELOPMENT TIME (HOURS)	0.1244	0.1793	0.1833	0.8205	0.5038	0.1650	0.1466	0.0051	0.5610
LAMBDA	0.43196	0.27955	0.42150	0.79726	-0.38440	0.26615	0.25618	0.15288	0.69091
LANGUAGE LEVEL	0.2125	0.4341	0.2251	0.0057	0.2727	0.4573	0.4750	0.6733	0.0269
MUTIME	0.38112	-0.25198	0.12967	-0.03547	0.11939	0.45513	0.47382	0.12875	0.24250
MEAN USER TIME (SECS)	0.2772	0.4825	0.7211	0.9225	0.7425	0.1863	0.1665	0.7230	0.4996
MSYSTIME	0.32843	0.05313	0.18184	-0.32956	0.05215	0.18266	0.20590	0.52321	-0.06554
MEAN SYSTEM TIME (SECS)	0.3542	0.8841	0.6151	0.3524	0.8862	0.6135	0.5682	0.1207	0.8573
MELAPSED	0.41740	0.01687	0.21380	-0.29264	0.09371	0.29710	0.31649	0.50821	0.02664
MEAN ELAPSED TIME (H:MM:SS)	0.2301	0.9631	0.5531	0.4119	0.7968	0.4045	0.3730	0.1337	0.9418
MEXECTIM	0.39264	-0.03804	0.19105	-0.28165	0.08141	0.29711	0.32222	0.47060	0.02480
MEAN EXECUTION TIME (SECS)	0.2617	0.9169	0.5970	0.4305	0.8231	0.4045	0.3639	0.1698	0.9458
MOBJSIZE	0.91191	0.52097	0.87656	0.46663	0.06803	0.81172	0.85045	0.85040	0.79478
MEAN OBJECT CODE SIZE (BYTES)	0.0002	0.1226	0.0009	0.1740	0.8519	0.0044	0.0018	0.0018	0.0060
MEXECMEM	-0.05618	-0.27078	0.07907	-0.02890	0.34920	-0.08399	-0.08090	-0.43047	0.03585
MEAN EXECUTION MEMORY (BYTES)	0.8775	0.4492	0.8281	0.9368	0.3227	0.8176	0.8242	0.2143	0.9217
NCOMP	-0.07584	0.10310	-0.05807	-0.35522	0.27330	-0.25315	-0.26473	0.03641	-0.22830
# OF COMPILATIONS	0.8350	0.7769	0.8734	0.3138	0.4448	0.4804	0.4598	0.9205	0.5258

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 5323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 10

	N2	PLEN	EPLEN	PVOL	EPPVOL	EPLEV	EPDIF	EFFORT	TIME
LOC	0.47193	0.76454	0.87762	0.86179	0.68378	-0.08278	0.06254	0.51880	0.51880
# OF NBNC LINES OF CODE	0.1685	0.0100	0.0008	0.0013	0.0292	0.8202	0.8637	0.1244	0.1244
VG	0.50208	0.71815	0.66847	0.77849	0.55850	-0.19463	0.10469	0.46156	0.46156
MCCABE'S CYCLOMATIC COMPLEXITY	0.1392	0.0193	0.0346	0.0080	0.0933	0.5900	0.7735	0.1793	0.1793
NVAR	0.37104	0.70100	0.85899	0.80641	0.65809	-0.08439	0.04513	0.45788	0.45788
# OF VARIABLES	0.2912	0.0239	0.0015	0.0048	0.0386	0.8167	0.9015	0.1833	0.1833
NCONST	0.08287	0.36486	0.84247	0.51478	0.89211	0.45749	-0.49071	-0.08260	-0.08260
# OF CONSTANTS	0.8200	0.2999	0.0022	0.1279	0.0005	0.1837	0.1498	0.8205	0.8205
NTYPE	-0.01608	-0.00525	-0.05033	-0.00748	-0.33451	-0.39878	0.40074	0.24022	0.24022
# OF TYPE DEFINITIONS	0.9648	0.9885	0.8902	0.9836	0.3448	0.2537	0.2511	0.5038	0.5038
NSUB	0.25496	0.60615	0.88258	0.73569	0.51223	-0.19096	0.14135	0.47538	0.47538
# OF SUBROUTINES	0.4771	0.0632	0.0007	0.0153	0.1301	0.5972	0.6969	0.1650	0.1650
ETA1	0.26764	0.61806	0.88798	0.74407	0.50452	-0.20699	0.16339	0.49412	0.49412
# OF UNIQUE OPERATORS	0.4547	0.0569	0.0006	0.0136	0.1370	0.5661	0.6520	0.1466	0.1466
N1	0.80925	0.98245	0.71755	0.99616	0.50680	-0.40994	0.39762	0.80346	0.80346
TOTAL # OF OPERATORS	0.0046	0.0001	0.0195	0.0001	0.1349	0.2394	0.2552	0.0051	0.0051
ETA2	0.23987	0.58661	0.99767	0.74157	0.87539	0.21465	-0.25730	0.20964	0.20964
# OF UNIQUE OPERANDS	0.5044	0.0747	0.0001	0.0141	0.0009	0.5515	0.4730	0.5610	0.5610
N2	1.00000	0.90463	0.24591	0.80203	0.14825	-0.60263	0.61740	0.84682	0.84682
TOTAL # OF OPERANDS	0.0000	0.0003	0.4934	0.0053	0.6827	0.0652	0.0572	0.0020	0.0020
PLEN	0.90463	1.00000	0.59865	0.97737	0.41474	-0.48876	0.48452	0.85179	0.85179
PROGRAM LENGTH	0.0003	0.0000	0.0675	0.0001	0.2334	0.1517	0.1558	0.0018	0.0018
EPLEN	0.24591	0.59865	1.00000	0.75250	0.84109	0.16437	-0.20730	0.24864	0.24864
EST. PROGRAM LENGTH	0.4934	0.0675	0.0000	0.0120	0.0023	0.6500	0.5655	0.4885	0.4885
PVOL	0.80203	0.97737	0.75250	1.00000	0.55746	-0.36144	0.34664	0.76769	0.76769
PROGRAM VOLUME	0.0053	0.0001	0.0120	0.0000	0.0941	0.3048	0.3265	0.0095	0.0095

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 5323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=O / N = 10

	N2	PLEN	EPLEN	PVOL	EPPVOL	EPLEV	EPDIF	EFFORT	TIME
EPPVOL	0.14825	0.41474	0.84109	0.55746	1.00000	0.55866	-0.58273	-0.09620	-0.09620
EST. POTENTIAL PROGRAM VOLUME	0.6827	0.2334	0.0023	0.0941	0.0000	0.0932	0.0771	0.7915	0.7915
EPLEV	-0.60263	-0.48876	0.16437	-0.36144	0.55866	1.00000	-0.97828	-0.83746	-0.83746
EST. PROGRAM LEVEL	0.0652	0.1517	0.6500	0.3048	0.0932	0.0000	0.0001	0.0025	0.0025
EPDIF	0.61740	0.48452	-0.20730	0.34664	-0.58273	-0.97828	1.00000	0.85943	0.85943
EST. PROGRAM DIFFICULTY	0.0572	0.1558	0.5655	0.3265	0.0771	0.0001	0.0000	0.0014	0.0014
EFFORT	0.84682	0.85179	0.24864	0.76769	-0.09620	-0.83746	0.85943	1.00000	1.00000
# OF ELEM. MENTAL DISCRIM.	0.0020	0.0018	0.4885	0.0095	0.7915	0.0025	0.0014	0.0000	0.0001
TIME	0.84682	0.85179	0.24864	0.76769	-0.09620	-0.83746	0.85943	1.00000	1.00000
DEVELOPMENT TIME (HOURS)	0.0020	0.0018	0.4885	0.0095	0.7915	0.0025	0.0014	0.0001	0.0000
LAMBDA	-0.18401	0.05248	0.64587	0.20838	0.92479	0.82966	-0.83618	-0.44565	-0.44565
LANGUAGE LEVEL	0.6108	0.8855	0.0437	0.5635	0.0001	0.0030	0.0026	0.1968	0.1968
MUTIME	-0.05501	0.07593	0.27626	0.13142	-0.04911	-0.14760	0.21439	0.23990	0.23990
MEAN USER TIME (SECS)	0.8800	0.8348	0.4397	0.7174	0.8928	0.6841	0.5520	0.5044	0.5044
MSYSTIME	0.61972	0.57636	-0.03344	0.46965	-0.26643	-0.65698	0.76267	0.82254	0.82254
MEAN SYSTEM TIME (SECS)	0.0560	0.0812	0.9269	0.1708	0.4568	0.0390	0.0103	0.0035	0.0035
MELAPSED	0.54209	0.54083	0.06241	0.46471	-0.21923	-0.60305	0.71053	0.78503	0.78503
MEAN ELAPSED TIME (H:MM:SS)	0.1055	0.1065	0.8640	0.1760	0.5428	0.0650	0.0213	0.0071	0.0071
MEXECTIM	0.49021	0.49707	0.06205	0.42756	-0.23431	-0.58636	0.69463	0.75197	0.75197
MEAN EXECUTION TIME (SECS)	0.1503	0.1438	0.8648	0.2178	0.5147	0.0748	0.0258	0.0121	0.0121
MOBJSIZE	0.50999	0.77889	0.81490	0.85034	0.55380	-0.20643	0.21545	0.61760	0.61760
MEAN OBJECT CODE SIZE (BYTES)	0.1321	0.0079	0.0041	0.0018	0.0967	0.5672	0.5500	0.0571	0.0571
MEXECMEM	-0.60619	-0.50479	0.02123	-0.40678	0.16925	0.60870	-0.56632	-0.57579	-0.57579
MEAN EXECUTION MEMORY (BYTES)	0.0632	0.1367	0.9536	0.2434	0.6402	0.0618	0.0879	0.0815	0.0815
NCOMP	0.37493	0.14547	-0.24167	0.04962	-0.16429	-0.30078	0.23338	0.14695	0.14695
# OF COMPILATIONS	0.2857	0.6884	0.5012	0.8917	0.6502	0.3984	0.5164	0.6854	0.6854

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 5323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 10

	LAMBDA	MUTIME	MSYSTIME	MELAPSED	MEXECTIM	MOBJSIZE	MEXECMEM	NCOMP
LOC	0.43196	0.38112	0.32843	0.41740	0.39264	0.91191	-0.05618	-0.07584
# OF NBNC LINES OF CODE	0.2125	0.2772	0.3542	0.2301	0.2617	0.0002	0.8775	0.8350
VG	0.27955	-0.25198	0.05313	0.01687	-0.03804	0.52097	-0.27078	0.10310
MCCABE'S CYCLOMATIC COMPLEXITY	0.4341	0.4825	0.8841	0.9631	0.9169	0.1226	0.4492	0.7769
NVAR	0.42150	0.12967	0.18184	0.21380	0.19105	0.87656	0.07907	-0.05807
# OF VARIABLES	0.2251	0.7211	0.6151	0.5531	0.5970	0.0009	0.8281	0.8734
NCONST	0.79726	-0.03547	-0.32956	-0.29264	-0.28165	0.46663	-0.02890	-0.35522
# OF CONSTANTS	0.0057	0.9225	0.3524	0.4119	0.4305	0.1740	0.9368	0.3138
NTYPE	-0.38440	0.11939	0.05215	0.09371	0.08141	0.06803	0.34920	0.27330
# OF TYPE DEFINITIONS	0.2727	0.7425	0.8862	0.7968	0.8231	0.8519	0.3227	0.4448
NSUB	0.26615	0.45513	0.18266	0.29710	0.29711	0.81172	-0.08399	-0.25315
# OF SUBROUTINES	0.4573	0.1863	0.6135	0.4045	0.4045	0.0044	0.8176	0.4804
ETA1	0.25618	0.47382	0.20590	0.31649	0.32222	0.85045	-0.08090	-0.26473
# OF UNIQUE OPERATORS	0.4750	0.1665	0.5682	0.3730	0.3639	0.0018	0.8242	0.4598
N1	0.15288	0.12875	0.52321	0.50821	0.47060	0.85040	-0.43047	0.03641
TOTAL # OF OPERATORS	0.6733	0.7230	0.1207	0.1337	0.1698	0.0018	0.2143	0.9205
ETA2	0.69091	0.24250	-0.06554	0.02664	0.02480	0.79478	0.03585	-0.22830
# OF UNIQUE OPERANDS	0.0269	0.4996	0.8573	0.9418	0.9458	0.0060	0.9217	0.5258
N2	-0.18401	-0.05501	0.61972	0.54209	0.49021	0.50999	-0.60619	0.37493
TOTAL # OF OPERANDS	0.6108	0.8800	0.0560	0.1055	0.1503	0.1321	0.0632	0.2857
PLEN	0.05248	0.07593	0.57636	0.54083	0.49707	0.77889	-0.50479	0.14547
PROGRAM LENGTH	0.8855	0.8348	0.0812	0.1065	0.1438	0.0079	0.1367	0.6884
EPLEN	0.64587	0.27626	-0.03344	0.06241	0.06205	0.81490	0.02123	-0.24167
EST . PROGRAM LENGTH	0.0437	0.4397	0.9269	0.8640	0.8648	0.0041	0.9536	0.5012
PVOL	0.20838	0.13142	0.46965	0.46471	0.42756	0.85034	-0.40678	0.04962
PROGRAM VOLUME	0.5635	0.7174	0.1708	0.1760	0.2178	0.0018	0.2434	0.8917

CORRELATIONS BETWEEN ALL MEASUREMENTS

COMSC 5323

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 10

	LAMBDA	MUTIME	MSYSTIME	MELAPSED	MEXECTIM	MOBJSIZE	MEXECMEM	NCOMP
EPPVOL	0.92479	-0.04911	-0.26643	-0.21923	-0.23431	0.55380	0.16925	-0.16429
EST. POTENTIAL PROGRAM VOLUME	0.0001	0.8928	0.4568	0.5428	0.5147	0.0967	0.6402	0.6502
EPLEV	0.82966	-0.14760	-0.65698	-0.60305	-0.58636	-0.20643	0.60870	-0.30078
EST. PROGRAM LEVEL	0.0030	0.6841	0.0390	0.0650	0.0748	0.5672	0.0618	0.3984
EPDIF	-0.83618	0.21439	0.76267	0.71053	0.69463	0.21545	-0.56632	0.23338
EST. PROGRAM DIFFICULTY	0.0026	0.5520	0.0103	0.0213	0.0258	0.5500	0.0879	0.5164
EFFORT	-0.44565	0.23990	0.82254	0.78503	0.75197	0.61760	-0.57579	0.14695
# OF ELEM. MENTAL DISCRIM.	0.1968	0.5044	0.0035	0.0071	0.0121	0.0571	0.0815	0.6854
TIME	-0.44565	0.23990	0.82254	0.78503	0.75197	0.61760	-0.57579	0.14695
DEVELOPMENT TIME (HOURS)	0.1968	0.5044	0.0035	0.0071	0.0121	0.0571	0.0815	0.6854
LAMBDA	1.00000	-0.08211	-0.47467	-0.41394	-0.41571	0.29753	0.42228	-0.23510
LANGUAGE LEVEL	0.0000	0.8216	0.1657	0.2344	0.2322	0.4038	0.2241	0.5132
MUTIME	-0.08211	1.00000	0.42030	0.61214	0.66835	0.45457	0.11261	-0.19821
MEAN USER TIME (SECS)	0.8216	0.0000	0.2265	0.0600	0.0346	0.1869	0.7568	0.5831
MSYSTIME	-0.47467	0.42030	1.00000	0.97164	0.95586	0.46226	-0.40048	-0.02090
MEAN SYSTEM TIME (SECS)	0.1657	0.2265	0.0000	0.0001	0.0001	0.1786	0.2514	0.9543
MELAPSED	-0.41394	0.61214	0.97164	1.00000	0.99473	0.53549	-0.29844	-0.05388
MEAN ELAPSED TIME (H:MM:SS)	0.2344	0.0600	0.0001	0.0000	0.0001	0.1107	0.4023	0.8825
MEXECTIM	-0.41571	0.66835	0.95586	0.99473	1.00000	0.52614	-0.29184	-0.08131
MEAN EXECUTION TIME (SECS)	0.2322	0.0346	0.0001	0.0001	0.0000	0.1182	0.4133	0.8233
MOBJSIZE	0.29753	0.45457	0.46226	0.53549	0.52614	1.00000	-0.03194	-0.09635
MEAN OBJECT CODE SIZE (BYTES)	0.4038	0.1869	0.1786	0.1107	0.1182	0.0000	0.9302	0.7912
MEXECMEM	0.42228	0.11261	-0.40048	-0.29844	-0.29184	-0.03194	1.00000	-0.01879
MEAN EXECUTION MEMORY (BYTES)	0.2241	0.7568	0.2514	0.4023	0.4133	0.9302	0.0000	0.9589
NCOMP	-0.23510	-0.19821	-0.02090	-0.05388	-0.08131	-0.09635	-0.01879	1.00000
# OF COMPILATIONS	0.5132	0.5831	0.9543	0.8825	0.8233	0.7912	0.9589	0.0000

VITA

Keith E. Moll

Candidate for the Degree of

Master of Science

Thesis: AN EMPIRICAL STUDY OF THE RELATIONSHIP BETWEEN STATIC SOFTWARE COMPLEXITY METRICS AND DYNAMIC MEASUREMENTS OF PASCAL AND C PROGRAMS

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Olathe, Kansas, February 17, 1964, the son of Mr. & Mrs. Kenneth G. Moll, Jr.

Education: Graduated from Gardner-Edgerton High School, Gardner, Kansas, May 1982; received Associate's degree in Computer Science Technology from Kansas Technical Institute, Salina, Kansas, May 1984; received Bachelor of Science and Master of Science degrees in Computer Science from Oklahoma State University, Stillwater, Oklahoma, May 1987 and July 1989, respectively.

Professional Experience: Computer Programmer, Conoco Inc., Ponca City, Oklahoma, June 1984 to August 1985; Instructor, Indian Meridian Vocational Technical School, Stillwater, Oklahoma, September 1986 to December 1987; Teaching Assistant, Department of Computer Science, Oklahoma State University, January 1988 to May 1989.