EXPERT SYSTEM FOR COMPUTER ASSISTED FLORISTIC CLASSIFICATION

By

LINDA WOODRING BARNES

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1974

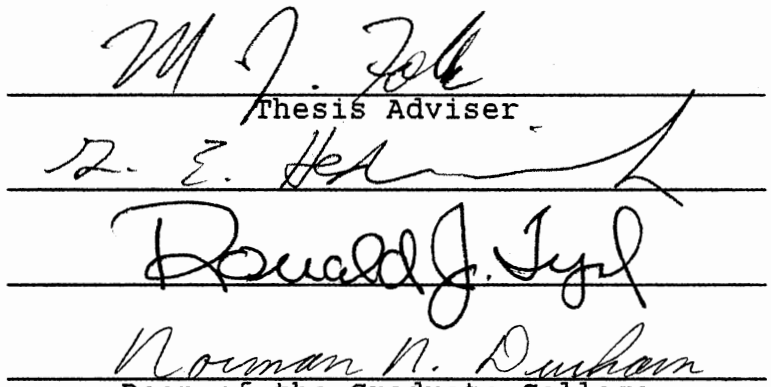
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 1988

EXPERT SYSTEM FOR COMPUTER

ASSISTED FLORISTIC

CLASSIFICATION

Thesis Approved:



Thesis Adviser

Dean of the Graduate College

ACKNOWLEDGMENTS

I wish to express my appreciation to IBM who allowed me time to attend graduate school on a full time basis to enhance my career opportunities within the company.

Many thanks to my advisors Dr. Mike Folk, Dr. G. E. Hedrick, and Dr. Ronald J. Tyrl for their suggestions and encouragement during my thesis.

A special thanks also goes to Dr. K. M. George and Dr. Donald Fisher for their guidance and support during my graduate studies.

I would also like to thank both by mother, Sherry Woodring and my mother-in-law Lois Barnes for their emotional support to myself and my family while working on my masters degree.

Thanks also goes to my two sons, Micah and Brandon and my daughter Andrea who joined our family during my studies.

Finally, a very special thanks to my husband, Frank. This endeavor would not have been possible without his loving support and belief in my abilities. Without his artistic ability, the illustrations in the students' guide would not have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Computers in Education

In the last few years, schools and universities in many parts of the world have acquired large numbers of computers (Bork, 1987). Students are now becoming computer literate in elementary school. Only a few years ago, computer literacy did not occur until college. Sitting before a microcomputer at home or at school is becoming as commonplace for students as turning on a television (Crovello, 1982).

Several state and national committees are investigating the crisis in science education. There has been a reexamination of the foundations of science education, and there has been a restructuring of the goals and methods of science teaching. Computer technology is receiving high priority among the new topics of study. As a result, science educators are now asking: "Why use computers in the classroom?" (Ellis, 1984)

Computers can permit one to teach subject matter that is already taught, but help teach it more efficiently by increasing the student's interest in the subject matter. For example, students required to take general biology

become excited about the subject of taxonomy when part of their assignment is to use the computer to identify what species an unknown maple specimen belongs (Crovello, 1974).

Most of the early applications of computers in education focused on the computer as a teacher. These programs presented material, asked questions, and branched appropriately. New educational programs differ from the traditional computer assisted instruction (CAI) approach that uses the computer for drill and practice. They instead use a new branch of computer science, artificial intelligence. Recent advances in artificial intelligence have opened up the possibility of using computers as "expert tutors". The key distinction between this form of software and early CAI is that in an expert tutor the student remains the primary agent in the student-computer dialogue. In an expert tutor, the student acts as the primary problem solver (Balkovich, Lerman, and Parmelee, 1985).

One area in science education that is traditionally frustrating for a beginning student is learning to classify unknown organisms. This study looks at how learning the taxonomy of one group of organisms, the flowering plants, can be made simpler and less confusing for the student by using the computer. This will be accomplished by the use of an expert system which takes the student step by step through the keying process.

Taxonomic Keys

Identification of unknown plants is typically accomplished by means of a dichotomous taxonomic key. Such a key presents the user with progressive choices between pairs of alternative characteristics. The user examines the unknown plant, then chooses one alternative or the other. Selection of the most applicable alternate leads to other pairs of alternatives and ultimately to a scientific name.

Keys provide a convenient shortcut method of identifying plants by outlining and grouping related types. There is somewhat of a "knack" in using keys which an individual develops mostly by constant practice and experience (Harrington and Durrell, 1957). Because keys frequently use scientific technical terminology and several characteristics at once to determine the next pair of alternatives, their use initially can be frustrating for the student. If the initial reaction to keying plants is negative, it is difficult to excite the student or even interest him further in identification and taxonomy. When the student has a good understanding of the descriptive terms commonly used and can relate them to the plant being identified, then use of the key is not difficult and becomes a learning experience.

Two types of taxonomic keys are commonly used (see Figure 1). In the bracket key the two choices of a pair always are positioned together and given the same number

or letter. Some authors do not indent alternate pairs of choices as is illustrated. The indented key is much like a outline and easier to use because each new pair of choices is indented to the right. Each pair of alternative choices is given a number or alphabetic character. This particularly is useful in a long key when the members of a pair may be separated by numerous other pairs. Groups and the characters that characterize them are more easily seen in an indented key. The majority of modern taxonomic manuals use the indented type of key (Harrington and Durrell, 1957).

BRACKET KEY

1. Flowers red..... 2
1. Flowers blue..... 5
   2. leaves simple..... 3
   2. leaves compound..... 4
3. Petals 4.....Species no. 1
3. Petals 5.....Species no. 2
   4. leaflets 5.....Species no. 3
   4. leaflets 9-11.....Species no. 4

INDENTED KEY

A. Flowers red
   B. Leaves simple
      C. Petals 4.....Species no. 1
      C. Petals 5.....Species no. 2
   B. Leaves compound
      D. Leaflets 5.....Species no. 3
      D. leaflets 9-11.....Species no. 4
A. Flowers blue
   E. Flowers sessile.....Species no. 5
   E. Flowers pedicIled
      F. Inflorescence a raceme.....Species no. 6
      F. Inflorescence a panicle.....Species no. 7

FIGURE 1. Types Of Taxonomic Keys

Keys may be difficult to use for several different reasons. The key may use characters not present on the specimen at hand. The meaning of some terms may vary from key to key. Several characteristics may be given in each alternative and the characters given in the one alternative of the pair may not be contrasted in the second.

The uniqueness of terms used in taxonomy poses a stumbling block for most beginning students. Many terms are encountered only in keys and the student must learn how to use the key plus a set of terms at the same time.

Intent of Study

The purpose of this study is to demonstrate how using a computer during a student's first introduction to keying unknown plants can make the process less frustrating and more enjoyable. This objective will be accomplished by writing a computer-assisted plant identification program utilizing an expert system written in Prolog. Instead of presenting the student with multiple pairs of choices simultaneously as occurs where a traditional taxonomic key is used, the program reduces the keying technique to one decision at-a-time regarding the plant's characteristics. The scientific terms used to describe the plants have been simplified and eliminated as much as possible.

Expert systems differ from traditional computer programs by usually using declarative languages or shells. This makes the program easier to modify and update. Once

the program has been written, a teacher with a limited computer programming background should be be able to make changes to the program to fit his or her own individual needs. A teachers' guide will be provided for this purpose (see Appendix A).

CHAPTER II

REVIEW OF LITERATURE

The Case For Computer Use In Education

Educational software has existed almost as long as computers have been available in academic settings. The amount, diversity, and quality of such software has undergone great changes, but never as rapidly as the present. Today 16 and even 32 bit microprocessor-based microcomputers are becoming available. Educators soon will have much of the capacity of a mainframe computer on their classroom tables. Crovello's article "Evolution of Educational Software" documents the changes in educational software and predicts future developments.

There have been many major hardware changes important in educational computing. These include increased abilities in storage, graphics, access ability, and decreased price. From 1980 to the present, microcomputers changed significantly the evolution of hardware and thus of software available to educators. New educational programs are taking advantage of these changes. To utilize the latest in educational packages, schools must budget or find outside endowment money to purchase the latest microcomputers available.

The abilities and attitudes of teachers also are changing. More educators have become comfortable with computers and are less hesitant to consider their use. They now are demanding quality software in their classrooms. The result is healthy competition among software suppliers which, in turn, are producing innovative, valuable programs.

A question frequently asked is: "Why isn't there more good instructional software for the microcomputer?" There are several reasons for the lack of high quality instructional software (Spain, 1985). One of the major problems in courseware development is that it simply takes a lot of time to develop a polished product. Between two and five years may elapse, after an idea is conceived and the time the program finally is published. Second, relatively few people have both the subject area knowledge and the skill to design instructional software and to program it as well. Third, the financial rewards are not very great for the author of an educational program. An author may make the equivalent of only $1.50 an hour for a program that is targeted for use by a very specific audience in the school.

Computing is now recognized as the fourth basic skill, along with reading, writing, and arithmetic (National Science Foundation, 1979). In the article "A Rationale for Using Computers in Science Education", Ellis relates the economic status of our country to our successful transfor-

mation to an information society and to the level of our nations' scientific and technological literacy. The rapid transformation of the nation into an information society compels educators to establish computer literacy as an important goal.

Computer literacy can best be developed in subject areas. Restricting computers to classes in computer literacy separates the skill from the application. That is similar to restricting the activity of reading only to a reading class. The skill obtains relevance in its use in a realistic problem situation (Ellis, 1984).

Botanical Classification

Programs were developed using the computer to create taxonomic keys during the late 60's and 70's. Programs that created keys embodied the use of data matrices. Information on the features of various taxa was presented in tabular form using the data matrix method (Morse, 1974). Taxa were positioned along one axis and various characters along the other. By providing matrices for taxa of different ranks, the data could be linked hierarchically using both forward and backward pointers. Hall (1970) also used a data matrix form and assigned a numeric property value for scaling characteristics observed. Advantages of computer efficient key can be found in "Botanical Keys Generated by Computer" by Pankhurst (1971). The major advantages are the ease of editing the key and the fact that through

computer networking one could get an immediate revision of a manuscript key for a taxonomist in his laboratory or herbarium.

Another extension of the key-editing system involves computer identification of individual specimens, using as input a list of observed characteristics (Morse, Beaman, and Shetler, 1968). Several programs of this nature have been developed. One such program is described by Goodall (1968). After the user has specified the value of an attribute displayed on the computer screen, he is told how many taxa are still consistent with the characters so far entered. He is given the options of specifying another attribute value, being given the names or full descriptions of the taxa, or being given a list of attributes which can distinguish among the remaining taxa. This program could only be done by using a large main frame computer due to the size of the required information. Morse recognized that routine application of computer-stored data matrices to specimen identification presents problems: (1) terminals must be located in herbaria; and (2) a network of accessible taxonomic data matrix files must be prepared and be available.

Due to the complexity of both types of programs, only those individuals competent in taxonomy could utilize them. There are advantages of such a national or international taxonomic information system: completeness, standardization, and revisability (Morse, 1971). Copies of

the entire data base printed periodically could be kept as historical records.

Identification aided by the computer is possible with present technology. Programs available are designed primarily for the experienced taxonomist and not easily used by novices. Today the use of artificial intelligence techniques can make identifying plants possible for the beginner using a microcomputer. An expert system can reduce the code so that by using a personal computer, a subset of a large plant identification program can be brought directly into the classroom.

Expert Systems

Artificial intelligence (AI) is simply the transfer of intelligence to machines (Levine, Drang, and Edelson, 1986). Expert Systems deal with a small area of expertise that can be converted from human to AI . They work with a knowledge base in a particular field, drawing inferences in one way or another (Simons, 1985). This single area of expertise is referred to as the domain of the expert system.

What is generally considered to be "intelligence" can be divided into a collection of observations or facts and a means of utilizing these facts to reach goals. For example, a goal might be to determine why a car will not start. The expert system prunes these facts to eliminate from consideration any facts and rules that won't lead the user to a specified goal. The portion of intelligence that

generates new facts from existing ones and to arrive at the goal is the "inference mechanism".

Expert systems can be applied to problems that are solved primarily using formal reasoning. The problem is solved through a dialog, or "consultation," with the expert system (Townsend, 1987). In a simple expert system, each question is answered with "yes" or "no". After each question, either the program may request an answer to another question or it makes an inference based on the facts it already has accumulated.

Knowledge engineers are used to develop expert systems. They are skilled at observing and analyzing the methods used by human experts to solve problems in a particular discipline. These methods, or heuristics, are stored as part of the data.

There are three basic components of an expert system. The first component, the rule-base, is a static database that contains all the knowledge about the domain. The second component, the working memory, houses the dynamic database to store the new facts obtained from the user or inferred from known facts. The inference engine is the third component. It contains the general problem-solving logic.

One of the most common types of expert systems is the ruled-based system. In a rule-based system, knowledge is represented as IF-THEN statements (rules). When the IF portion of a rule is true in the current situation, the

action specified by the THEN portion is executed or said to fire.

A typical rule for finding the disease of a plant might be (Latin, Miles, and Regginger, 1987):

IF the plant symptom is wilt,
   and the wilt is rapid,
   and no yellow tissue is associated with wilted
       leaves,
   and bacterial streaming can be demonstrated
       from freshly cut stems,

THEN the disease is bacterial wilt.

The working memory contains facts that describe what is known about a particular problem. When a program is started, the working memory is empty. As the consultation progresses and the system learns more about the problem, the new knowledge is put into working memory. The knowledge in working memory is used to fire additional rules. As each rule fires, the conclusion is added to working memory with the facts already known.

The inference engine has two tasks: one is inference, and the second is control. The inference component uses the facts in working memory to try to trigger new rules. After all conditions of a rule are triggered, the rule fires and the conclusion is added to working memory. The control component determines the order in which the rules are scanned.

Most expert systems use two types of search strategies, forward chaining and backward chaining, to control the order in which the system goes about using the rules

and finding the final goal. Like inductive logic, forward chaining reasons forward from existing facts and rules to derive additional facts that must hold, while following all possibilities suggested by the data. Like deductive logic, backward chaining reasons backwards from a given goal, searching the knowledge base for facts or rules supporting that goal and declaring them true (Williams, 1986).

Insight into the special nature of expert systems can be gained from a comparison of expert systems to conventional programs. Each require different developmental approaches. The most fundamental difference is that conventional programs deal with data, whereas expert systems deal with knowledge. Knowledge implies an awareness or understanding gained through experience or study. Conventional programs operate according to algorithms, formal procedures designed to produce correct or optimal solutions. Expert system rules embody judgmental knowledge, rules of thumb, or simplifications used by experts. Conventional programs use a top-down approach which make it difficult to change system design once coding has begun. Uncoupling knowledge from its application makes a data-driven system much easier to modify as the expert system evolves (Williams, 1986).

Programming Languages

Traditional programming languages have not proved to be well suited to computer applications in expert systems

(Simons, 1985). Expert systems have been written in Cobol, Pascal, Ada, Fortran, C, and Basic, but such languages are far from ideal in representing knowledge required by AI. The emergence of new programming tools has stimulated the development of AI-related systems. Perhaps more than anything else, the lack of an adequate language hindered the development of expert systems for productive applications on the personal computer (Townsend, 1987).

Special languages, notably Lisp and Prolog, have been developed to facilitate the programming of AI applications. Aware of the need for a language to process symbolic information, John McCarthy invented Lisp in the early 1960's at MIT. In 1972, Alain Colmerauer and P. Roussel at the University of Marseilles began the development of Prolog. Such languages are often called descriptive, declarative, relational, or logic programming languages. Traditional languages are referred to as procedural languages.

Four commands are central to Lisp's symbol-manipulation capability. These basic commands can sort out symbols, build up lists, determine the truth or falsity of a function, and can match the if-side of a production rule. In general, Lisp's goal is to evaluate something and return a value (Myers, 1986). Most of the larger expert systems have been written in LISP or in a LISP-based languages such as OPS5. Disadvantages to using LISP are that it is best suited to an expensive workstation or superminicomputer, and it is not the easiest tool to use

(Lisp experts are still in relatively short supply) (Simons, 1985).

Prolog usually is regarded as much easier for the novice to understand (Simons, 1985). It was selected in 1981 as the basis for the Japanese Fifth Generation Computer project. It is now gaining acceptance in the United State as well (Myers, 1986).

Prolog is short for Programming in Logic. It was created especially for answering questions about a knowledge base that consists of rules and facts (Levine, Drang, and Edelson, 1986). Writing expert systems using Prolog is particularly easy compared to other languages because it has backward chaining built in and also utilizes another technique known as "backtracking." Recall that backward chaining assumes a conclusion to be true, and then a knowledge base of rules and facts is examined to see if it supports the assumption. If the original assumption is not correct, backtracking replaces it with a new one.

Today, several powerful Prolog compilers are available for the personal computer. In 1985, the Arity Prolog compiler was marketed. Borland's Turbo Prolog was introduced in May 1986. A user can now compile a true expert system with hundreds of rules that will function on a personal computer.

Prolog does have a few disadvantages. First, the order of the rules and facts is important to their meaning. Second, all rules must reside in the computer's memory.

The number of rules that the expert system can use is limited by the memory size of the computer. With most versions of Prolog there are methods by which the disk can be used as an extension of memory, but this alternative virtually ensures a very slow program for an interactive session.

Programming in a AI language such as LISP or Prolog is completely different from using a procedural language. If a programmer has spent years learning procedural languages, he will have to go through an "unlearning" experience before he can begin to get proficient in these languages. There is an advantage: The computer can be used to solve new problems that are not adaptable to solution using traditional languages.

CHAPTER III

PROGRAM DESIGN CONSIDERATIONS

Program Objectives

The main objective of this project is to write an educational program that allows beginning botany students at the high school and college levels to identify flowering plants with the assistance of a computer. Various program designs can be employed to accomplish this goal. To ensure a quality program six goals are identified before the design process begins. These goals reflect upon both how the teacher and students will accept the program.

The first goal is that minimal typing is to be done by the student. Many students and teachers use the "hunt and peck" method when at a computer. Programs that require large amounts of keystrokes are frustrating for these individuals and add greatly to the time it takes to identify a plant. The user is assumed to have minimal typing skills. All questions are written to require only a one keystroke answer.

The second subgoal is to make the decision process as simple as possible. This can be accomplished by removing technical botanical terms whenever possible. Many terms used in plant taxonomy can be translated into

descriptions familiar to the user. The use of technical terminology in keys of professional taxonomists is one of the reasons that keying plants is so difficult for a beginner. To aid the student in understanding terms used in the program, a glossary of terminology used and illustrations is provided to the student with the program (see Appendix B).

The decision process can also be simplified by reducing the decision to one characteristic at a time. The characteristics that the student must observe and make a decision about can be minimized. By selecting the families to be keyed in advance, only those characteristics required to differentiate between the families need to be included in the program.

Making the program attractive and appealing to the eye and easy to read is the third goal. This is accomplished by a dual window screen. See figure 2 for an example of a running program. Questions to be answered always appear on the left hand side of the screen and facts gathered from the user always appear on the right hand side of the screen. For those students with color monitors, each window uses contrasting colors to further separate the two functions of the windows. When a class, family, subfamily, genus, or species is determined, the appropriate name of the taxon appears on a line by itself and is highlighted. The highlighting emphasizes which characteristics are used to determine the plant's family ,etc.

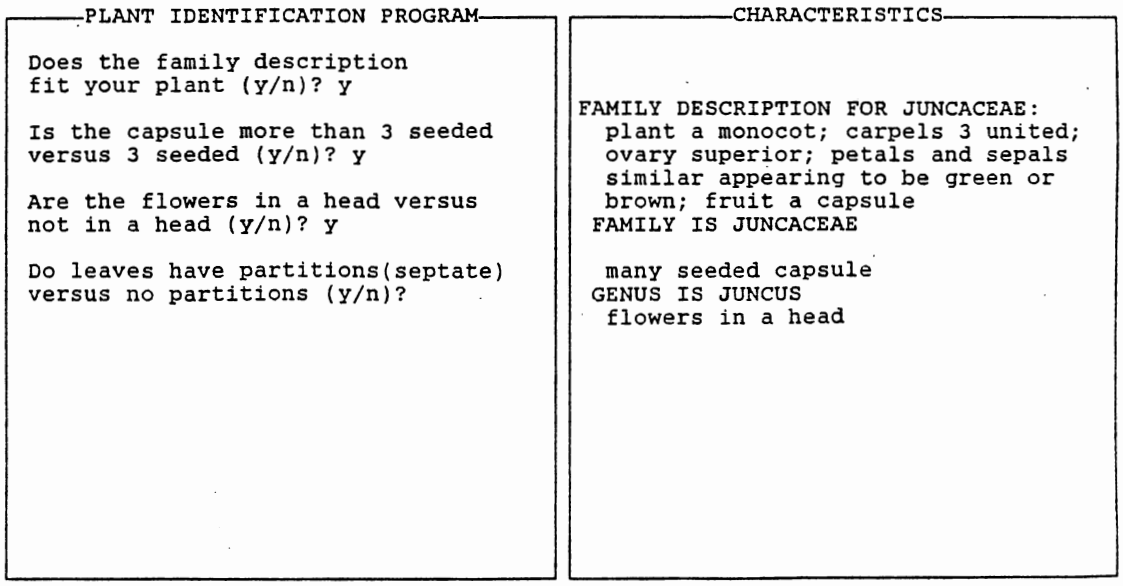


Figure 2. Example of a Running Program

The fourth requirement is to allow a teacher with a limited computer science background to modify the program. The teacher can add families of plants that are abundant or unique to his area. The design of the program should therefore be as simple as possible, but still incorporate the other goals. A maintenance manual written for the teacher accompanies the program (see Appendix A).

The program's fifth objective is to be accurate. In order for the teacher to have faith in the program, it is imperative the data inputted by the student leads to the correct identification. The program will be based on a dichotomous key authored by U. T. Waterfall (1972). This key is accepted as the standard reference for the vascular

plants of Oklahoma by the scientific community. The program will be compared with Waterfall two times for verification. As the teacher makes modifications to the program, he must ensure accuracy in order to maintain the integrity of the program.

Because most high schools do not have access to a mainframe computer, the sixth and last goal requires the program to run on a personal computer. To be less confusing to the students, the program is written to be a standalone program. The student, therefore needs only to insert the diskette and turn on the computer. This program was developed using an IBM Personal System/2 Model 30 and designed to run on any IBM or IBM compatible personal computer with 640K of memory.

Plant Families

Because the memory size in a personal computer is limited relative to a mainframe, a taxonomic key for all of the flowering plants of a one state could not fit in main memory. Oklahoma, for example, has approximately 152 families, 834 genera, and 2600 species. The selection of which families, genera, and species to incorporate into the program is a major design decision. With the assistance of Ronald J. Tyrl in the Department of Botany and Microbiology at Oklahoma State University, it was decided to include 10 families in the program. More families can be included but space constraints prevent the inclusion of

additional genera and species. It is important that the beginning student be introduced to the classification levels of genus and species in order to show how organisms are grouped in a hierarchy of categories and how organisms are classified from general characteristics shared to specific characteristics that are unique to that organism. The hierarchy of classification of organisms shows the student the diversity of the biological world.

The 10 families chosen include plants that are commonly found throughout the United States. Some families are similar except for one characteristic and others are quite different in their characteristics. Of the 10 families included, one family has a key to the subfamililes level, five families have keys to their genera, and three genera have keys to the species level. Table I lists the taxa represented in the program.

Table I

TAXA IN BOTANY PROGRAM

FAMILY	SUBFAMILY	GENUS	SPECIES
Asteraceae			
Iridaceae		Nemastylis	
		Tigridia	
		Sisyrinchium	
		Belamcanda	
		Iris	
Fabaceae	Mimosoideae		
	Caesalpinioideae		
	Papilionoideae		
Fagaceae		Fagus	
		Castanea	
		Quercus	
Juncaceae		Juncus	J. tenuis
			J. interior
			J. coriaceus
			J. torreyi
			J. accuminatus
			J. marginatus
		Luzula	L. echinata
			L. bulbosa
Lamiaceae			
Liliaceae			
Magnoliaceae		Magnolia	M. acuminata
			M. tripetala
Rosaceae			
Verbenaceae		Verbena	
		Phyla	
		Callicarpa	

Inference Engine Design

Most of the inference design and control of a program is built into Turbo Prolog by its pattern matching and backtracking techniques. Prolog systems are predominately backward-chaining systems. Through pattern matching, it starts with an hypothesis and tries to prove it working backwards. For example, in this program instead of gathering characteristics about the plant and then finding a family, the program finds the first family listed and questions the student about the characteristics that fit that family. If the hypothesis fails, Prolog goes forward until it can find the next family, then uses backward chaining again.

Prolog uses a depth-first search strategy. Details are pursued as deeply as possible until the goals fail. After an outcome is proven false, the system backs up and then pursues the next outcome. All characteristics relative to a specific family, genus, or species are considered together and either accepted or rejected.

Because the inference engine is internal to Prolog, the programmer does not have to spend time designing the inference engine. It does however, limit the programmer to the search strategies that Prolog supports: depth-first searching using backward chaining and limited forward chaining.

When classifying plants, one moves from the most general category to more specific ones. By listing the

most inclusive characteristics first as is done in a dichotomous key, the program can quickly reduce the search space in correctly identifying a family. Instead of repeating the general characteristics at the genus level, the first characteristic would be the family. At the species level, the first characteristic would be the genus. For example, if the student was identifying the species Luzula bulbosa, the following code is used.

```
family(juncaceae):-
   check(monocot),
   check(ovary_superior), not(check(sepals_petaloid)),
   !,desc(juncaceae).

desc(junaceae):-
   write("\nFAMILY DESCRIPTION FOR JUNCACAE:"),
   write("\n several write staments describing"),
   write("\n Juncacae"),
   ck_desc,    /* checks with student if the
                  description is correct */ write("\n
   FAMILY IS JUNCACEAE"),
   highlight,
   asserta(dbase(juncaceae,'y')),
   genus(_),
   species(_).

genus(luzula):-
   check(juncaceae),          /* family name */
   not(check(gt3_seed_capsule)),
   write("\n GENUS IS LUZULA"),
   highlight,
   asserta(dbase(luzula,'y')).

species(bulbosa):-
   check(luzula),             /* genus name */
   check(rectangular_head),
   write("\n SPECIES IS L. BULBOSA"),
   highlight,
   asserta(dbase(bulbosa,'y')).
```

CHAPTER IV

PROJECT ASSESSMENT

Program Design

The basic design of the program proved to be fairly simple because the inference engine is built into Turbo Prolog. Townsend's Mastering Expert Systems with Turbo Prolog (1987), presents a step by step procedure for designing and building an expert system specifically with Turbo Prolog. An expert system to diagnose failures for IBM PC compatible computers is included in the text as an appendix. Using portions of this code as templates was an invaluable time saving aid.

Prolog does not use calls to subroutines, gotos, if-then-elses, or other similar structures used in procedural languages. Instead, it basically employs one construct known as a rule for execution control. A rule takes the following form:

<conclusion> :- <requirements>.

An example of two rules might be:

```
parent(X,Y):- mother(X,Y).
parent(X,Y):- father(X,Y).
```

This would read X is Y's parent if X is Y's mother or X is Y's parent if X is Y's father. A goal somewhere else in

the program would call parent. If the first parent rule succeeds, then true would be returned to the calling goal. If the first parent rule failed, then the 2nd parent rule would be tried. If both rules fail, then a fail would be returned to the calling goal.

By asking the student questions with a yes/no answer, characteristics about the plant are easily retained in the database. For instance, if the student responded to the question "Is the ovary superior?" with a "y", then an entry of (ovary_superior,'y') is inserted into the database. If a "n" is entered an entry of (ovary_superior,'n') is inserted into the database.

The requirements for a conclusion in Prolog may be multiple, in which case, all must be true before the conclusion is proved to be true. To determine if a plant belongs to a certain family, the program checks multiple characteristic of each family one at a time until either all of the characteristics of one family match or defaults to the rule family(undet) (i.e. family undetermined). For example, the rule for Juncaceae was written as follows:

```
family(juncaceae):-
   check(monocot),
   check(ovary_superior),
   not(check(petaloid_perianth_seg)), desc(juncaceae).
```

When the goal family(_) is encountered, the program searches the database for the entry (monocot,'y') or (monocot,'n'). If neither entry is found it then asks

the user the appropriate question or questions to determine the status of monocot. If the answer is true the program then checks the data base for (ovary_superior,'y'). Again if the answer is true the program continues to the next subgoal. To succeed this time, however, the program looks for (petaloid_perianth_seg,'n'). The preceding "not" negates the fail returned from (petaloid_perianth_seg,'n'). A subgoal of desc is then triggered to describe Juncaceae and asks the student to verify if he is at the right family before proceeding to the genus level. If the student does not accept the family description, a fail is returned to the goal family(juncaceae) which then returns a fail to the run subgoal. The student then is asked if he wants to try again.

The main menu allows the student to enter the program at three places (see Figure 3). If the family is known, the student can go directly to the family's description. If the class is known (monocot/dicot), a jump to the class description is made. If neither family nor class is known, a "don't know" option is available which asks two questions to determine the class. This multiple entry approach allows the student to bypass several questions as he becomes more knowledgeable of taxa and more competent in identifying unknown plants.

At any point in the program where the student is asked a characteristic about the plant, the student may

return to the main menu by hitting the escape key. The student may realize that he entered the wrong characteristic after an entry has already been made. If he is working at the genus or species level, he can avoid repeating the first questions by selecting the appropriate family.

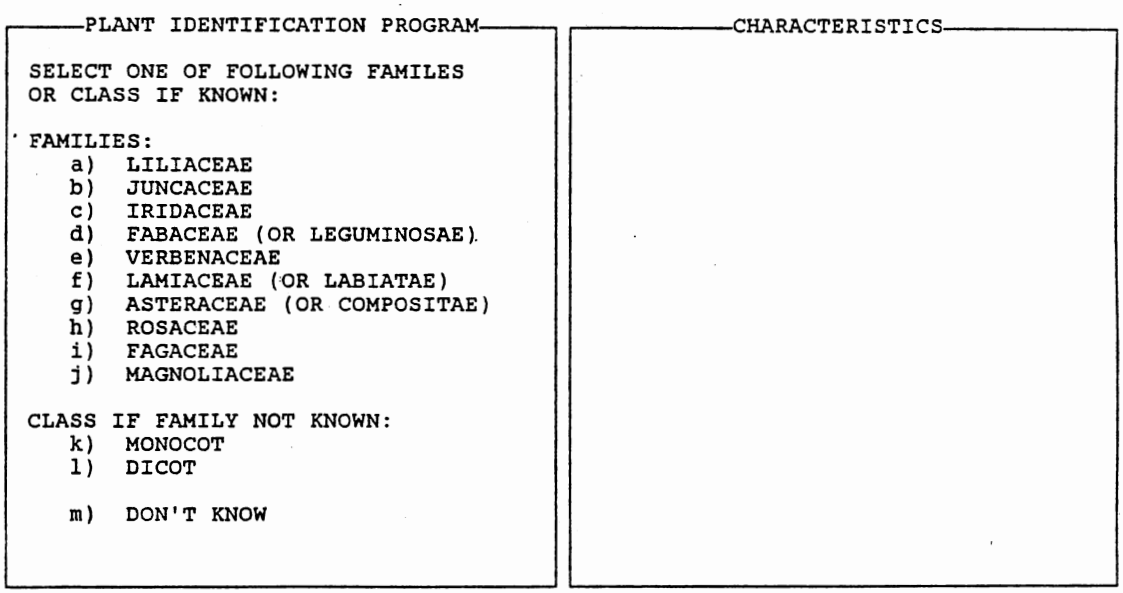


Figure 3. Main Menu

The database is cleared each time the main menu goal is called. An early version of the the program continued to question the student when the program should have ended, if the student used the escape key during the program session to return to the main menu. This occurred

because Prolog uses a backtracking method (tries to prove previous subgoals after a failure of a subsequent subgoal) and the database is cleared each time the main menu goal is called. To alleviate this problem a second database called "escape" was inserted. Before printing any questions, family descriptions, or class descriptions, the program first checks whether the escape key has been pressed. This allows the program to backtrack in the background without the user's knowledge and return the user to the main menu.

For the most part, this application proved to be easily done in Turbo Prolog. The database facility was used to hold the working memory of the expert system. The rule-base component was easily constructed by inserting the appropriate characteristics under each family, subfamily, genus, or species name. The built-in inference engine of Turbo Prolog was cumbersome only when the escape key was added. A trace feature of Turbo Prolog was invaluable in determining the inference engine's control pattern. Built-in features to create and manipulate windows made designing the program fun and easy.

Testing Results

The program was tested in two high schools. The first test class comprised 14 current or past botany students at McLoud High School, McLoud, Oklahoma. The majority of students had taken one semester of botany, while a few students had taken two semesters during the previous school

year. Taxonomy and identification of unknown plants was strongly emphasized by the instructor. The second test class, 38 students, came from two honors biology classes at Memorial High School, Tulsa, Oklahoma. The students received two days of instruction in plant identification immediately prior to the testing. The two groups were selected in order to test the suitability of the program for a broad spectrum of high school students - novices, beginners, experts.

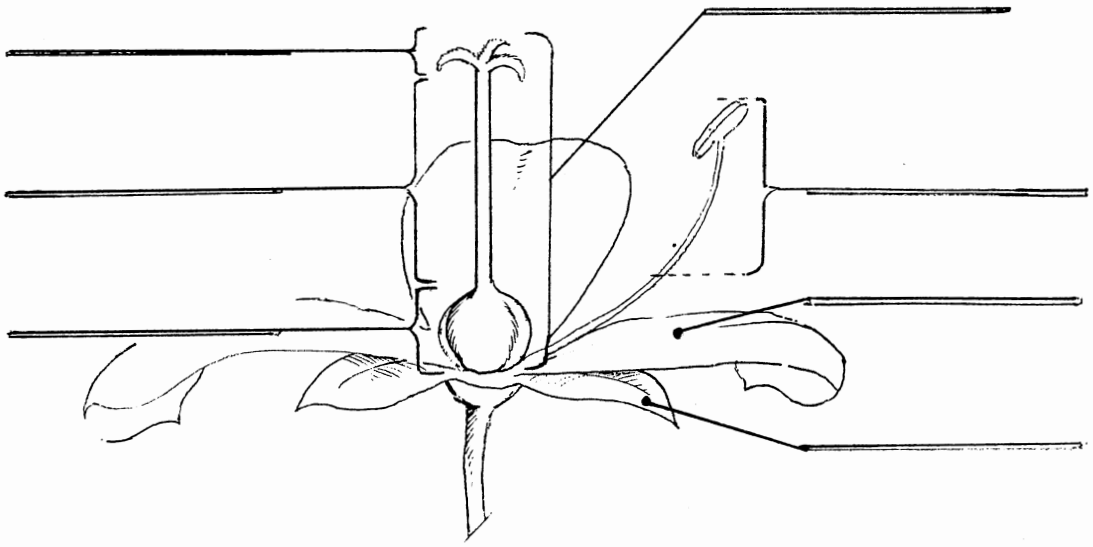
Before using the program, each student was given a short quiz covering basic plant structure to insure that each had a minimum knowledge of plant morphology and terms (see Figure 4). After each student or pair of students identified one to three unknown plants via the computer, an program evaluation questionnaire was taken anonymously (see Figure 5). The students working in pairs used an IBM Personal System/2 to test the program.

Both test classes passed the pretest. Individuals in the botany classes scored an average of 86 percent while those in the honors biology classes scored an average of 96 percent. The difference in average score is probably due to the fact the honors biology classes were drilled for the pretest the day before testing. In contrast, students in honors biology had never identified plants before using the program. The students in botany class had keyed an average of 24 plants prior to testing the program. Students in the botany class gave the program higher scores

in all aspects of the program. Table II compares the responses of three main questions on the student evaluation questionnaire. Their previous exposure to keying plants using a dichotomous key and using technical terms undoubtedly allowed them to make comparisons between the program and the traditional keys that they had used.

FLOWER PARTS PRE-TEST

Instructions: Name the parts of the flower indicated



Question: How many carpels are there in the above structure? _____

Figure 4. Pre-test

QUESTIONNAIRE TO BOTANY COMPUTER PROGRAM

1. List on the back of this paper any confusing terminology you could not understand **AFTER** using the user's guide.

2. Circle the grade you are currently in.

   Freshman Sophomore Junior Senior

3. Have you had before or are you currently enrolled in a botany class?

   Yes No

4. Is this the 1st time you have keyed a plant?

   Yes No

   If no, approximately how many plants have you keyed out.

5. On a **TIMED** classification test, which would you prefer to use?

   Computer Printed dichotomous key

6. Did you correctly identify your plant on the 1st try?

   Yes No

   If no, how many tries?

7. On a scale of 1 to 5 were the program's instructions (not questions) easy to understand?

   1 2 3 4 5
   Hard Easy

8. Overall on a scale of 1 to 5 how would you rank the program's ease of use?

   1 2 3 4 5
   Hard Easy

9. On a scale of 1 to 5 how would you rank the user's guide?

   1 2 3 4 5
   Poor Excellent

10. Are there any changes to the program and/or user's guide that would make them easier to use? If so, what are they?

Figure 5. Student Questionnaire

Table II

SUMMARY OF RESPONSES

Question no.	Min	Max	Ave	Class
7	2	5	4.3	Honors Biology
7	3	5	4.7	Botany
(Were the program's instructions easy to understand?)				
8	1	5	4.3	Honors Biology
8	4	5	4.9	Botany
(Rank the program's ease of use)				
9	1	5	2.9	Honors Biology
9	2	5	3.9	Botany
(Rank the Users' guide)				

Students in the the honors biology classes were given brief instructions on the use of a dichotomous key but none were successful in identifying their plants using a key. Both groups had a 71 percent success rate for correctly identifying the plant on the first try when using the computer. The honors biology classes, which had no previous botanical experience, were given easier plants to identify and given assistance regarding the intent of the questions when requested.

After using the program, many of the students ex-

pressed a desire to use the program again to identify plants. The botany students, after using a dichotomous key, were most vocal on enjoying the computer assisted method of keying plants. One hundred percent of the students in both test groups indicated on the questionnaire that they preferred to use the computer instead of a printed dichotomous key on a timed identification test.

The major problem that was revealed in the questionnaire responses concerned the users' guide. At test time, the glossary contained 13 definitions without illustrations and 11 definitions with illustrations. The students had problems locating terms in the guide. The guide was redesigned and expanded to contain 53 terms in alphabetical order. Every term used in the program is contained in the glossary. After each word is a brief description and refers to a figure number in the user's guide for a pictorial aide and further description where applicable (see Appendix B). This should facilitate quick access to key word information.

Another problem that surfaced in the honors biology class was one of technical terms. Even though many of the technical terms had been eliminated or simplified when writing the program, many students still had difficulty with those that remained. These terms were also replaced with less technical terminology. The reduction of technical terms, along with the expanded user's guide, should

alleviate the problems indicated by those students with minimal botanical experience.

CHAPTER V

SUMMARY AND FUTURE WORK

Summary

A challenge for botany teachers is finding an enjoyable and successful method of introducing beginning high school and college level students to identifying and classifying plants. The traditional method of using a dichotomous key is frustrating for most students. The key contains unique botanical technical terms and decisions which typically must be made about several characteristics at one time. This frustration can be reduced by replacing technical terms, by using the minimum number of characteristics the student needs to identify the plant, and by reducing the decision process to one characteristic at a time. To make the introduction to identifying plants more enjoyable, these modifications to the traditional key can be implemented with a educational computer package written for a personal computer and taken directly into the classroom.

By approaching the problem as an expert system, teachers have the capacity to modify the program by simply adding the characteristics of plants. Flowers that are commonly found in the teacher's locale can be added. Gen-

era keys can be added to families. A teachers' guide was developed with step by step instructions for program modifications (see Appendix A).

Honors biology classes were tested using the program. Thirty-eight students participated in the testing after two days of instruction on plant classification methods. With minimal assistance from the instructor, 71 percent of the class correctly identified a flowering plant on the first try and 14 percent on the second try. The main objective of the program was to reduce the frustration level beginning students have when keying their first plant. The students appeared to be having fun and were not bogged down with highly technical terms.

This program should be used quite easily, not only by students in botany classes but also by students in biology classes in school systems that do not offer botany. It is recommended that teachers spend more than two days of instruction on basic botany terminology because the test group consisted of honor students. Increasing preparation time also will give the student more confidence when keying for the first time and require less assistance from the teacher.

Results of the program testing indicated that this type of program can be used in the high school classroom to introduce plant identification. The success rate on the first attempt by novice students, which is not common when using printed dichotomous keys, suggests that frustration

was reduced substantially.

Future Work

Advanced botany students can also use this program. Technical terms could be used in abundance. As microcomputers with large memories become more common, the program can be extended to include more families or to include more genera and species of each family.

For college level classes that may have access to a mainframe database, the program can be expanded to include all the families, genera, and species known to a given area of the country. If this is done, a study to produce the most efficient search pattern should be considered. Families, genera, and species that are the most common should appear in the program at the top of their rule section because Prolog begins with the first rule and sequentially tries each rule in order until it succeeds.

A method of allowing the student to redo a characteristic when the escape key is engaged could be developed. Increasing the complexity of the program in this matter, however, would preclude an instructor from modifying the program unless he had a strong computer programming background.

The glossary and illustrations could be added online. According to a study by Houghton (1984), however, users without prior computer experience do poorly with online

aids. The memory requirements for illustrations and glossary however, would be better utilized for further expansion of genera and species levels. A very simplified version of this program could be written using illustrations for younger students limiting identification to the families.

BIBLIOGRAPHY

Balkovich, E., Lerman, S. and Parmelle, R. "Computing in Higher Education: The Athena Experience." Commun. ACM, Vol 28, Nov 1985, pp 1214-1224.

Bork, A. "The Potential for Interactive Technology". Byte, Feb 1987, pp 201-206.

Crovello, T. "Evolution of Educational Software". The American Biology Teacher, Vol 46, No 3, March 1984, pp 140-145.

Crovello, T. "Computers in Biological Teaching". Bioscience, Vol 24, Jan 1974, pp 20-23.

Ellis, J. "A Rationale for Using Computers in Science Education". The American Biology Teacher, Vol 46, April 1984, pp 200-206.

Goodall, D. "Identification by Computer". BioScience, Vol 18, June 1968, pp 485-488.

Hall, A. "A Computer-Based System for Forming Identification Keys". Taxon, Vol 19, Feb 1970, pp 12-18.

Harrington, H. and Durrell, L. How to Identify Plants. Chicago, IL: The Swallow Press Inc., 1957.

Latin, R., Miles, G. and Rettinger, J. "Expert Systems in Plant Pathology". Plant Disease, Vol 71, Oct 1987, pp 866-872.

Levine, R., Drang, E. and Edelson, B. A Comprehensive Guide To AI and Expert Systems. McGraw-Hill, Inc., 1986.

Morse, L. "Computer-Assisted storage and Retrieval of the Data of Taxonomy and Systematics". Taxon, Vol 23, Feb. 1974, pp 29-43.

Morse, L., Beaman, J., and Shetler, S. "A Computer System for Editing Diagnostic Keys for Flora North America". Taxon, Vol 17, Oct 1968, pp 479-483.

Morse, L. "Specimen Identification and Key Construction with Time-Sharing Computers". Taxon, Vol 20, May 1971, pp 269-282.

Myers, W. "Introduction to Expert Systems". IEEE Expert. Spring 1986. pp 100-109.

National Science Foundation. Technology in science education: the next ten years-perspectives and recommendations. Washington, D.C. 1979.

Pankhurst, R. "Botanical Keys Generated by Computer". Watsonia, 8, 1971, pp 357-368.

Simons, G. L. Experts Systems and Micros. NCC Publications, 1985.

Spain, J. "Why Isn't There More Good Instructional Software?". The American Biology Teacher, Vol 47, No 6, Sept 1985, pp 378-380.

Townsend, C. Mastering Expert Systems with Turbo Prolog. Indianapolis, IN: Howard W. Sams & Company, 1987.

Waterfall, U. T. Keys to the Flora of Oklahoma, 5th edit, 1972, Published by author, Oklahoma State University Bookstore, Stillwater, OK.

Williams, C. "Expert Systems, Knowledge Engineering, and AI Tools-An Overview". IEEE Expert, Winter 1986, pp 66-70.

APPENDICES

APPENDIX A

TEACHERS' GUIDE

NOTE TO THE TEACHER

This program was written as an introduction to keying plants for the high school student or college level student in a beginning botany class. It may also be used in a biology class with a botany unit. It is highly recommended before using this program your students are well versed in basic botany terminology.

Many of the difficult technical terms unique to botany have been removed and replaced with simpler terminology. The students' guide to the program contains a glossary with all the terms currently used in the program. The following is a list of terms that is considered a minimum knowledge level to successfully utilize the program by a student:

dicot
monocot
ovary
petals
pistil
sepals
stamens
stigma
style

The system requirements to run this program are a IBM or IBM compatible PC with 640K memory with two floppy disk drives or one floppy disk drive and a hard disk drive, and PC-DOS OR MS-DOS operating system, version 2.0 or later. If you want to modify the program to add your own families or take some of the existing families to a lower classification level you must purchase Turbo Prolog by Borland. This is a Prolog compiler which may be purchased directly from Borland, most major personal computer stores that sell

IBM or IBM compatible PCs, or mail order software businesses advertised in personal computer journals. Instructions to modify the program are presented in the next section of the teachers' guide.

Using this program should provide to your students a rewarding and fun first experience in keying plants. I hope as a teacher this tool will assist you to light the spark of interest in classifying plants in your students.

FAMILIES KEYED IN PLANT IDENTIFICATION PROGRAM

There are ten families (see Table 1) identified in this program. Several of the families key to the genus and species level. For those students more knowledgeable in classifying plants, the main menu provides them the opportunity to go directly to the family level.

Table I

TAXA IN BOTANY PROGRAM

FAMILY	SUBFAMILY	GENUS	SPECIES
Asteraceae			
Iridaceae		Nemastylis	
		Tigridia	
		Sisyrinchium	
		Belamcanda	
		Iris	
Fabaceae	Mimosoideae		
	Caesalpinioideae		
	Papilionoideae		
Fagaceae		Fagus	
		Castanea	
		Quercus	
Juncaceae		Juncus	J. Tenuis
			J. Interior
			J. Coriaceus
			J. Torreyi
			J. Accuminatus
			J. Marginatus
		Luzula	L. Echinata
			L. Bulbosa
Lamiaceae			
Liliaceae			
Magnoliaceae		Magnolia	M. Acuminata
			M. Tripetala
Rosaceae			
Verbenaceae		Verbena	
		Phyla	
		Callicarpa	

PROGRAM MODIFICATION

Files Required

You may modify this program to expand families to the genus or species level, add families, or substitute families already keyed. It is strongly recommended that you have some experience in programming, if you make modifications. You must have Turbo Prolog by Borland to make changes to the program.

Your program diskette contains four files:

autoexec.bat
command.com
botany.exe
botany.pro

To make a backup diskette, format a system diskette using the /S parameter when you format, to copy your command.com file. Copy the remaining three files to the backup diskette with the DOS copy command. Botany.pro contains the source code for the program.

The Turbo Prolog diskette used to modify your program must contain at least these seven files:

prolog.exe
prolog.ovl
prolog.sys
prolog.err
prolog.hlp
prolog.lib
init.obj

Make a backup diskette using the DOS copy command. The instructions will assume the Prolog diskette is in drive A and the botany diskette is in drive B. If your PC includes a hard drive, then load the seven Prolog files

onto it.

When you purchase Turbo Prolog, you will receive a detail manual from Borland describing how to use the Turbo Prolog system and features of the language. This guide will give you enough basics about Turbo Prolog to modify your program. Refer to the manual for any further assistance.

Loading Turbo Prolog

To use Turbo Prolog, you first load the program:

```
A> prolog
```

Press the enter key to display the copyright screen Now press the space bar and the Turbo Prolog main menu and four systems windows: editor, dialog, message, and trace will appear (see Figure 6). The editor window is used to enter or change programs.

The main menu shows the commands and pull-down menus available. Select an item on the menu by pressing the associated highlighted capital letter or by first moving the highlighted bar using the arrow keys and then pressing the enter key. You may return to the main menu anywhere in Turbo Prolog by continuing to hit the escape key until the main menu appears or by Ctrl Break.

Resizing Windows and Setup Option

Select the Setup option from the main menu. Now select the Windowsize option, then the Edit option to

enlarge the editor window. Use your right arrow key to expand the editor window over the dialog window to the right edge of the screen. If you want to use the whole screen for editing, you may also use the down arrow key to cover the message and trace windows with the editor window. Do not cover the bottom line of the screen. The function keys appear on the bottom line of the screen during an edit session. The message window is used for compile errors during compilation and the trace window traces the path of your program during a run if the trace command is in your program. You may exit from any portion of Turbo Prolog with the escape key and reformat your window sizes using the Setup option again.

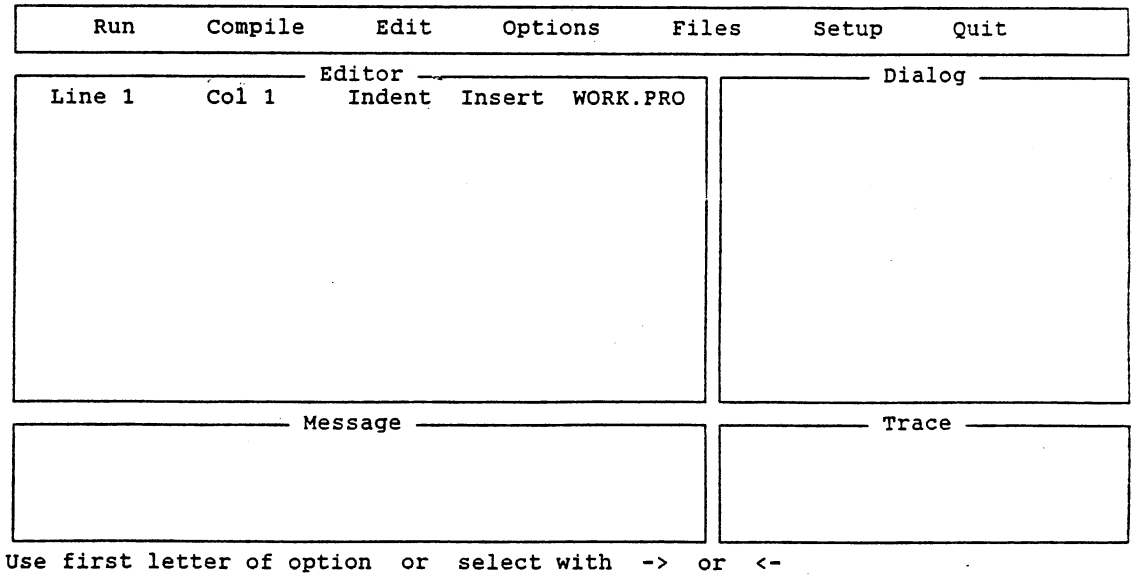


Figure 6. Turbo Prolog Main Menu

Hit the escape key twice to return to the Setup option's pull-down menu. Select the Directory option. Change the directory path of any files not correctly specified. For instance, with a two drive system, your botany.pro file, botany.obj, and botany.exe file are on drive B. You must one at a time move the menu bar to the file extension name, hit the enter key, enter b, and the enter key again. When all files are correct, hit escape to exit.

Loading the Botany Program

You are now ready to load the program into the editor. If you are not at the main menu (see Figure 7), hit escape until the main menu appears. Select the Files option, then the Load option. You can enter botany or use the enter key to display all the .pro files on the diskette in the drive specified with the Setup Directory option. Move the menu bar to the correct file to load then hit the enter key. The botany program is now loaded in the editor. If the system can not find your file, make sure the .pro file directory path in the Setup Directory option is correct. After the program has been loaded, Turbo automatically returns you to the main menu.

Saving Your Program

When you are through making changes to your program, you will need to save the edited program. Return to the

main menu. Select the Files option, then the Save option. The name of your program will appear on the screen. Hit enter if you want to save it under that name or enter the new name first if you want to change it. Hit the escape key to return to the main menu. You will probably want to save your program periodically during the edit session in case of a power outage. When you save a program, Turbo automatically creates a backup copy of your old version before edits were made using your file name and a .bak extension.

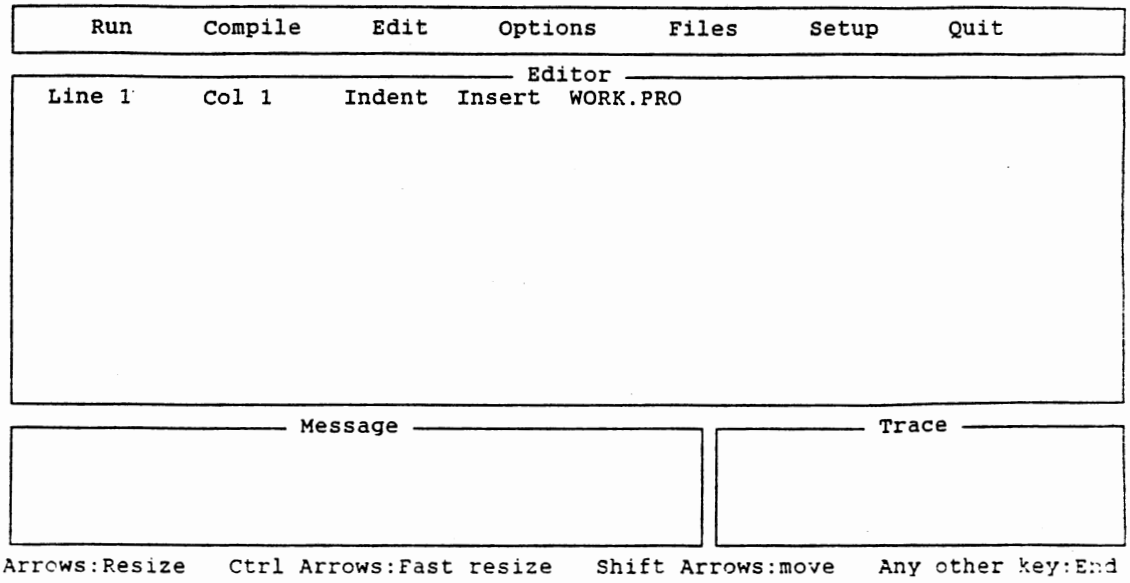


Figure 7. Turbo Prolog Main Menu after Resizing Windows

Editing

Select the Edit option and you are ready to edit the

Turbo Prolog program. The Turbo Prolog editor uses WordStar-like commands. A complete description of all the editor commands can be found in the Turbo Prolog manual. If a list of function keys is not at the bottom of your screen in edit mode, return to the Setup Window-size option and decrease the size of your editor window with the up arrow key. The insert key acts as a toggle switch to insert/overwrite when in edit mode. The cursor may be moved using the arrow keys, page up and page down keys, carriage return key, tab key, and backspace key. Only use the backspace key when you want to delete the character left of the cursor. Press the Help key, F1, to display a pop-up menu containing information about the function keys. Exit from the edit mode with the escape key or F10 key.

General Turbo Prolog Rules

Some general rules about Turbo Prolog follow:

1) All like clauses must be grouped together. (i.e. All Check(_) clauses appear together in one section of the code.)

2) To begin a new line use \n in a write statement between quotes or write nl.

3) When naming a symbol, you may use any character sequence of letters, numbers, and underscores, with the first character in lower-case. Symbols are objects used in predicates. For example, leaves_simple in check(leaves_simple) is a symbol for the predicate check.

4) To make comments in your program begin the comment section with /* and end the comment section with */.

Modifying the Program

Before you begin to modify your program, make sure you have a backup copy of the original program diskette. It will also be easier to the follow the instructions, if you obtain a printed copy of the program source code. You may get a copy after you have loaded the program into the editor. From the main menu, select the Files option, then the Print option. When the printout is completed, hit the escape key until the main menu appears again. When making changes to the program the following clauses will need modifying:

main_menu:-

1) To add a family

   a) Add a write statement using the next letter of the alphabet after the last family name.

   b) Push the letter of the alphabet down appropriately for the choices on the main menu for monocot, dicot, and don't know. For example, monocot is alphabet character 'k'. If you add one family, monocot would become a 'l', dicot a 'm', and don't know a 'n'.

   c) Change the statement Z > 96 to add one to the ASCII number for each family added. For example, if one family is added, then 96 would become 97.

   d) Go to ck_family instructions.

2) To substitute a family

   a) Change the write statement from the substituted family's name to the new family's name.

   b) Go to ck_family instructions.

3) To key a family to a lower classification

   a) Go to desc(family name) instructions.

ck_family('character'):-

1) To add a family

   a) Add a new ck_family('char') clause with the letter of the alphabet used in the main_menu clause for the new family.

   b) The symbol in the desc(symbol) statement should be the new family name.

   c) Change the characters in the ck_family clauses for monocot, dicot, and don't know to match the new alphabetic characters in the main_menu clause.

2) To substitute a family

   a) Find the ck_family clause that matches the substituted family name.

   b) Change the family name in the desc(symbol) statement to the new family name.

3) To key a family to a lower classification

   a) no changes needed.

desc(family name):-

1) To add a family

   a) Create a new desc(family name) clause using the new family name.

   b) Find a family to use for a template in the desc(family name) clauses. Use a family that keys to the same level of your new family. For example, the family Liliaceae keys only to the family level, the family Iridaceae keys to the genus level, and the family Juncaceae keys to the species level.

2) To substitute a family

   a) Follow the steps in adding a family.

b) Save the desc(family name) of the substituted family, in case you want to add it to the program at a later date, by commenting out that section of the code. Insert /* at the beginning of the code and */ at the end of the code.

3) To key a family to a lower classification

a) Follow step 1b in adding a family, changing your family desc clause to the correct format instead of adding a new family desc clause.

b) Go to the genus(genus name) instructions if adding genera.

c) Go to the species(species name) instructions if only adding species.

family(family name):-

1) To add a family

a) Use a dichotomous key to find the characteristics that distinguish the new family from the families in the program.

b) For those characteristics not in the check(characteristic) clause section, add the characteristics using one of the the check(characteristic) clauses as a template. Be sure to keep track which characteristic is assumed for a 'n' answer.

c) Using a family(family name) clause as a template, write a new clause for the added family. The characteristics should be given in the order you would find them in the dichotomous key. A not in front of the check(character-istic) clause means the student answered 'n' to the question regarding that characteristic.

d) Make sure your new family(family name) clause appears before the family(undet) clause in the code.

e) You must repeat steps 1a through 1d for each place in the dichotomous key your family can be identified to. For example: Asteraceae can be keyed to 4 different groups, so family(asteraceae) appears 4 times in my program code.

2) To substitute a family

a) Follow steps 1a through 1e in adding a family name.

b) Comment out the code for the substituted family in case you want to use add it in the program at a later date. Insert /* at the beginning of the code and */ at the end of the code.

3) To key a family to a lower level

a) No changes needed.

genus(genus name):-

1) To expand a family to genus level

a) Use a dichotomous key to find the characteristics that distinguish the genera of a family from each other.

b) For those characteristics not in the check(characteristic) clause section, add the characteristics using one of the the check(characteristic) clauses as a template. Be sure to keep track which characteristic is assumed for a 'n' answer.

c) Using a genus(genus name) clause as a template, write a new clause for each added genus. The characteristics should be given in the order you find them in the dichotomous key. A not in front of the check(characteristic) clause means the student answered 'n' to the question regarding that characteristic. Make sure that first characteristic is the family name.

d) Make sure to add your genus(genus name) clause in the code before genus(undet).

e) Go back to the desc(family name) clause section and modify the desc clause for your family name using as a template a family name that keys to the genus level if you have not done so.

2) To delete a genus

a) Comment out the code with a /* at the beginning of the code and a */ at the end of the code in case you want to add it back at a later date.

b) Go back to the desc(family name) clause section and modify the desc clause for your family name using as a template a family name that does not key to genus level.

species(species name):-

1) To expand a family to species level

a) Use a dichotomous key to the find the characteristics that distinguish the species of a genus from each other.

b) For those characteristics not in the check(characteristic) clause section, add the characteristics using one of the check(characteristic) clauses as a template. Be sure to keep track which characteristic is assumed for a 'n'.answer.

c) Using a species(species name) clause as a template, write a new clause for each added species. The characteristics should be given in the order you find them in the dichotomous key. A not in front of the check(characteristic) clause means the student answered 'n' to the question regarding that characteristic. Make sure that first characteristic is the genus name.

d) The write statement for the species name should include the first initial of the genus name immediately before the species name.

e) Make sure to add your species(species name) clause in the code before species(undet).

f) Go back to the desc(family name) clause section and modify the desc clause for your family name using as a template a family name that keys to the species level if you have not done so.

2) To delete a species

a) Comment out the code with a /* at the beginning of the code and a */ at the end of the code.

b) Go back to the desc(family name) clause section and modify the desc clause for your family name using as a template a family name that keys to the genus level or family name that keys to the family level if the genus is also removed.

Compiling and Running the Program

To compile your program, return to the main menu. Select the compile option. If you have an error during compilation, Turbo automatically puts you into the edit mode and positions the cursor under the error. An error message will appear at bottom left side of the screen. After the program successfully compiles, select the Run option from the main menu to run the program. When the program has completed its run, Turbo will instruct you to hit the space bar. You will then be returned to the main menu.

After you are satisfied with your program changes, you may want to have your program in an executable form. Once the program is in executable form, it is no longer necessary to have Turbo Prolog to run the program. To do this, select the Options option from the main menu. Move the selection bar to the Exe file (auto link)

selection. Hit the enter key. Turbo will then take you back to the main menu and convert your program to executable form the next time you compile the program.

To have the program compiled in memory again, return to the main menu and select the Options option. Move the selection bar to the Memory selection and hit the enter key. When you first load Turbo Prolog, the Memory option is automatically selected for you.

Programming Errors

There are several types of programming errors you might encounter. If you key to the wrong family, genus, or species check the following:

1) Make sure the spelling in all clauses are the same for characteristics and family, genus, or species names.

2) Make sure the characteristics are in the same order you find them in the dichotomous key.

3) Make sure you don't have the same characteristics used by another family, genus, or species. If you do, you need to find another characteristic to distinguish between them.

4) If you still can't find the error, retrace your steps in making the changes with the directions in this guide making sure to use correct templates.

5) You may also uncomment the trace feature in the program. Remove the /* and */ at the beginning of the program around the trace statement. Make sure your edit window does not cover the trace window and run the program after recompilation. The trace window displays each goal that is called and the cursor is under the current running portion of your program. Hit the F10 key to continue running the program. You may use the escape key any time you want to leave the running trace. After you have solved your problem, be sure to recomment out the trace statement.

If you have errors in formatting your output, remember these rules:

1) Both the Characteristics window and the Plant Identification Program window has a 38 character width length. If you use a write statement equal to or greater than the length of the window, the cursor will wrap to the next line automatically. This means if your write statement is exactly the length of your window, then the next write statement does not require a /n or nl to begin a new line.

2) If you want to highlight a line across the window, such as family is family name, etc., you must pad the right hand side of the write statement with spaces to the length of the window minus one (to allow for line wrapping) to highlight the entire line. If your line is not highlighted, you may have to many spaces padded.

APPENDIX B

USERS' GUIDE

GLOSSARY

ALTERNATE - one leaf arising at a node (see figure 1)

AROMATIC - Strong odor given off by flower or leaves

BRACTS - a modified leaf situated near a flower (see figure 2)

BULB - a underground stem with fleshy scale leaves and roots arising from base like an onion (see figure 3)

CAPSULE - a dry fruit splitting along several seams to release seeds

CARPEL - unit of a pistil consisting of highly modified leaf; pistil may have one carpel or more than one (see figure 4)

COMPOUND - structure consisting of more than one part; a compound leaf has blade completely divided into two or more leaflets (see figure 5); a compound pistil has two or more carpels.

CORM - thickened, vertical solid underground stem bearing aerial growth from single terminal bud (see figure 3)

DICOT - plant with flower parts usually in fours or fives, sometimes numerous; leaves net-veined; taproot or fibrous root system; woody or herbaceous

FIBROUS - root system composed of roots all same size and resembling fibers

FRUIT - ripened ovary and any other structure that encloses it at maturity

HEAD - dense cluster of sessile or nearly sessile flowers or fruits on a very short axis and partially surrounded by bracts (see figure)

HERB - plant whose stems and leaves are green and die back to the ground at the end of the growing season

IMPERFECT - flower with either stamens or pistils but not both

INFERIOR OVARY - ovary located below where the sepals are attached and appears to be sunken in the stem; flower parts appear to arise from top of the ovary (see figure 6)

INFLORESCENCE - the arrangement of flowers on a plant; may be solitary or only one per stem, or many in a head, or loosely clustered; inflorescence may be terminal (flowers located at the tip of the stems) or lateral (flowers found along the stem in axils of leaves)

IRREGULAR SYMMETRY - flower in which petals are not alike or different in size (see figure 7)

LEGUME - characteristic fruit of pea family; splits open along two seams (see figure 8)

LOMENT - legume fruit conspicuously constricted between seeds (see figure 8)

MONOCOT - petals, sepals, and stamens usually in threes; leaves parallel veined; fibrous root system only; herbaceous only

NET VEINED - leaves with one large vein in the center of the leaf with smaller veins radiating from it; the small veins connecting to each other and forming a net (see figure 9)

NUTLET - small, hard nut-like fruit characteristic of mint, vervain, and borage families; formed from four lobed ovary.

OPPOSITE - two leaves arising at a node and situated across the stem from each other (see figure 1)

OVARY - basal part the pistil that contains the seeds; develops into fruit (see figure 11)

PARALLEL VEINED - leaves with the major veins running the length of the leaf parallel to each other; most parallel veined leaves are long and narrow; (see figure 9)

PARTITIONS - structures that divide flower and vegetative parts

PERFECT - flower with both stamens and pistils

PETALOID - condition where the petals and sepals look alike and both appear to be colored and conspicuous

PETALS - parts of flower that are usually colored and conspicuous; found inside the green sepals (see figure 10)

PISTIL - female organ of the flower that produces the seeds; consists of the tip called the stigma, the middle portion called the style, and and enlarged base called the ovary which contains the seeds (see figure 10 & 11)

RECEPTACLE - the more or less expanded tip of the flower stalk from which the sepals, petals, stamens, and pistil arise (see figure 10)

REGULAR SYMMETRY - flower in which petals are all alike in size and form (see figure 7)

RHIZOME - A more or less horizontally elongated stem growing partly or completely beneath the surface of the ground (see figure 3)

SEPAL - outermost parts of flower that are are usually green and protect or enclose the petals in the bud (see figure 10)

SEPARATE - condition where flower parts are separate from each other and not fused together (see figure 12)

SEPTATE - divided by a partition

SHRUB - plant with several woody stems generally less than two meters in height

SIMPLE - structure consisting of only one part, not completely divided into separate segments; simple leaf has one blade (see figure 5)

SPHERICAL - round in outline or shape; like a globe

STAMENS - male organs of the flower that produce pollen; consists of anther and filament (see figure 10)

STANDARD - the upper, usually larger petal of flowers of pea family (see figure 13)

STIGMA - part of pistil that receives the pollen; at apex of style, usually hairy, bumpy, or sticky (see figure 11)

STIPULATE - pair of appendages of tissue(stipules) at the base of leaf petiole at either side of its attachment to the stem (see figure 14)

STYLE - the stalk-like part of the pistil connecting the ovary with the stigma (see figure 11)

SUBSPHERICAL - oval shaped; not quite round or spherical

SUPERIOR OVARY - ovary located above where sepals, petals, and stamens are attached (see figure 7)

TAPROOT - thick tapering root with much smaller lateral roots; like a beet or carrot (see figure 3)

TEPALS - petals and sepals that are alike in size, shape, and color; may be colored and showy or green and inconspicuous

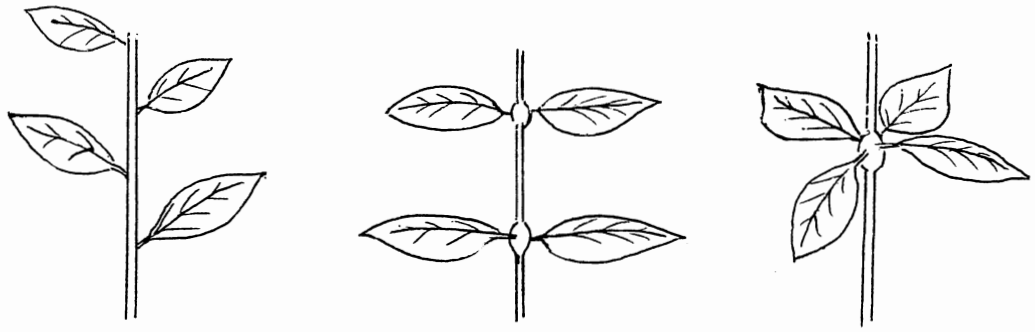
TERMINAL - located at tip of structure

TREE - plant with one large woody stem (trunk) and smaller branches; generally more than two meters in height

UNITED - condition where flower parts are fused together not separate; petals to petals or sepals to sepals (see figure 12)

WHORLED - three or more leaves arising at a node (see figure 1)

WOODY - plant of which some of its stems or trunk is not green, usually fibrous in nature



Alternate Opposite Whorled

Figure 1

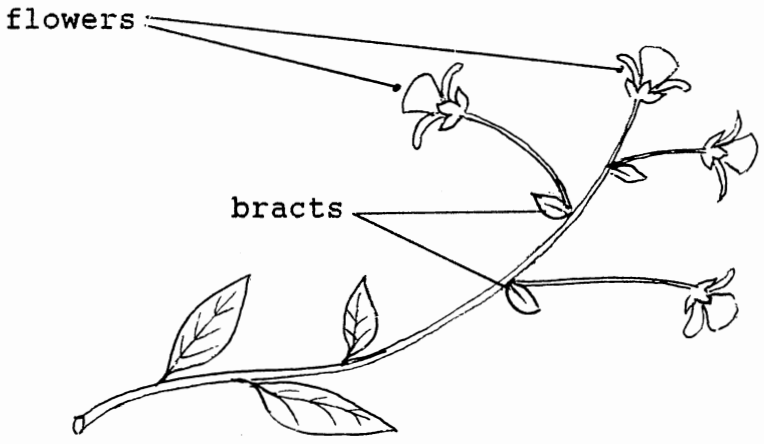
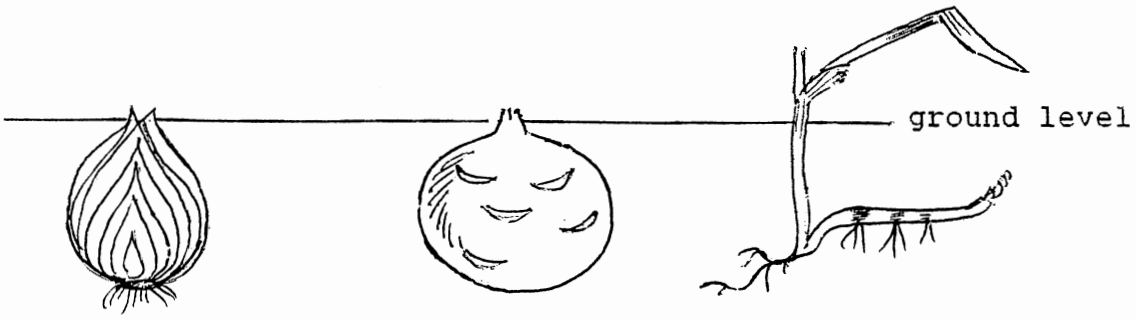
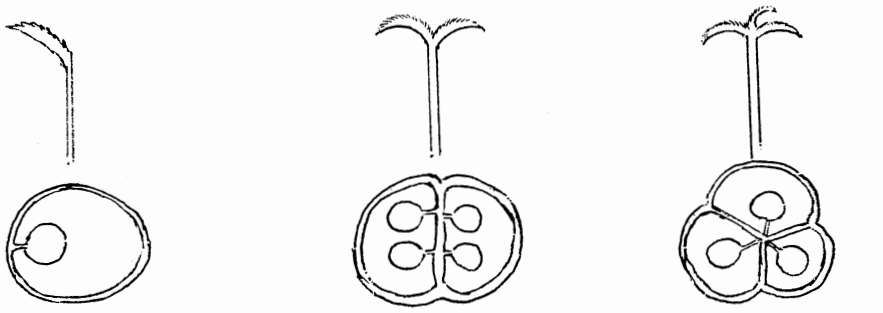


Figure 2



Bulb (l.s.) Corm Rhizome

Figure 3



(a) (b) (c)

A simple pistil (a) has one style, one undivided stigma and an unlobed ovary with seeds attached in one row inside. A compound pistil (b,c) has more than one style or more than one stigma and/or a lobed ovary and/or more than one row of seeds inside. The number of carpels is usually determined by counting the number of stigma lobes the pistil has.

Figure 4

Simple Compound

Figure 5

Inferior Superior

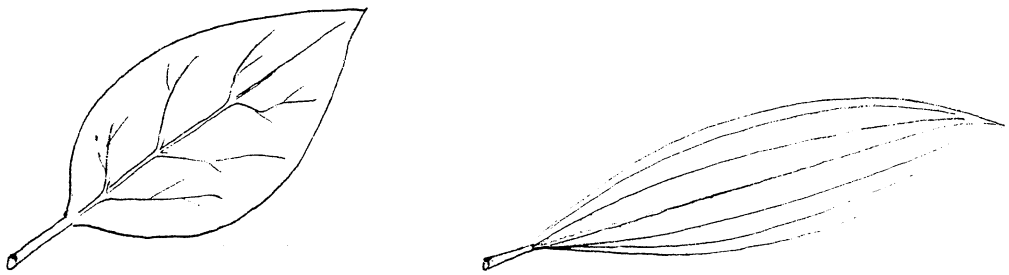
Figure 6

Irregular Regular

Figure 7

Legume Loment

Figure 8



Net-veined Paralled-veined

Figure 9

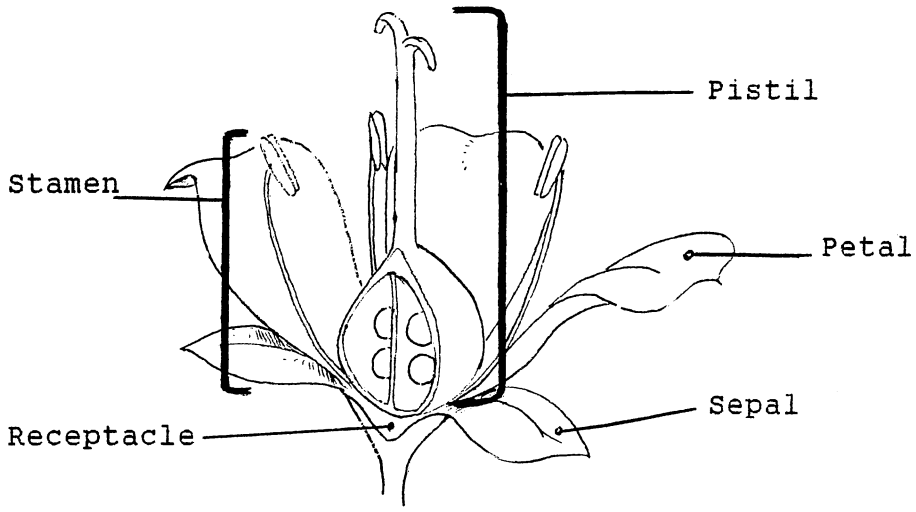


Figure 10

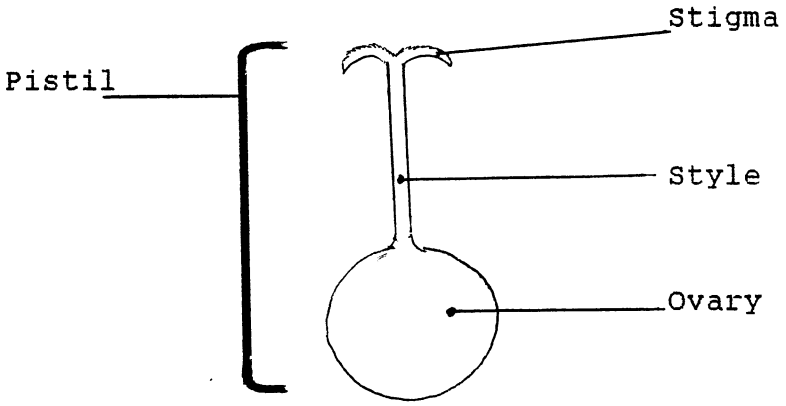
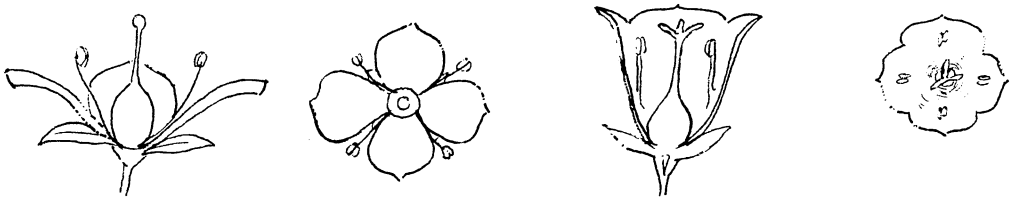


Figure 11



Separate United

Figure 12

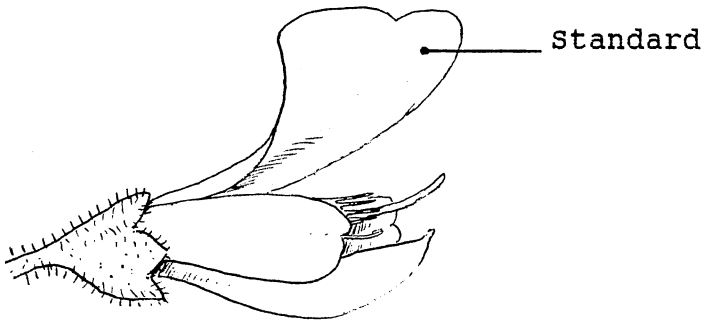


Figure 13

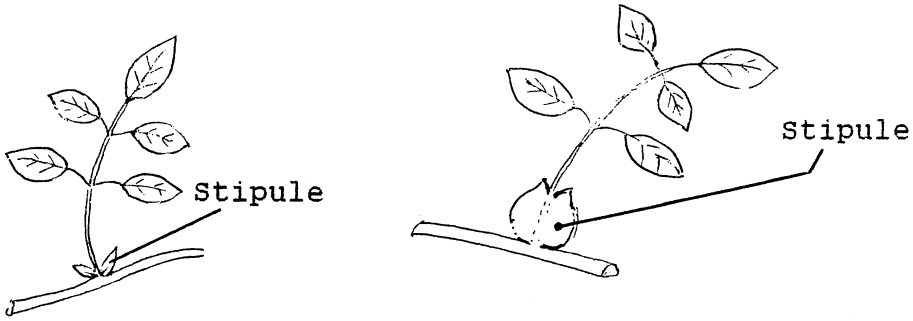


Figure 14

VITA

Linda W. Barnes

Candidate for the Degree of

Master of Science

Thesis: EXPERT SYSTEM FOR COMPUTER ASSISTED FLORISTIC CLASSIFICATION

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Bellville, Illinois, February 18, 1952, the daughter of Larry L. and Sherry L. Woodring. Married to Frank M. Barnes on August 19, 1972. Son Micah J. Barnes, born March 31, 1976. Son Brandon L. Barnes, born May 26, 1980. Daughter Andrea M. Barnes, born February 20, 1987.

Educational: Graduated from Sapulpa High School, Sapulpa, Oklahoma, in May, 1970; received Bachelor of Science Degree in Business Education from Oklahoma State Unversity in May, 1974; completed requirements for the Master of Science degree at Oklahoma State University in May, 1988.

Professional Experience: Dispatcher and Senior Inventory Control Specialist, International Business Machines, Tulsa, Oklahoma, June 1974 to June 1985.