IMPLEMENTATION OF AN EXPERT SYSTEM FOR PRODUCTION SCHEDULING USING OBJECT-ORIENTED TECHNIQUES

By

KRISHNASWAMI RAVI

Bachelor of Engineering

Madurai Kamaraj University

Madurai, India

1989

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, 1992

[illegible]

IMPLEMENTATION OF AN EXPERT SYSTEM FOR PRODUCTION SCHEDULING USING OBJECT-ORIENTED TECHNIQUES

Thesis Approved:

[illegible]

Thesis Adviser

Ramesh Sharda

[illegible]

Thomas C. Collins

Dean of the Graduate College

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to Prof. David Miller for his guidance, encouragement and serving as my major adviser. I am grateful to Dr. Ramesh Sharda for serving on my committee. Through his expertise in the area of Intelligent Scheduling Systems and generous contribution of his time, he has contributed sufficiently to the completion of my thesis. My special thanks goes to Dr. Mansur Samadzadeh for serving on my committee. His sincere guidance has helped me in all phases of my graduate program. I also thank Dr. John Chandler and Dr. George Hedrick for their guidance and support in completing my course of study at Oklahoma State University.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Definition of Scheduling

The scheduling problem can be generally defined as "assignment of execution times and resources to a set of orders in order to optimize certain criteria, given a set of constraints" [Arff90].

Introduction and Motivation

AI program can be subdivided into the following three basic categories [Rauc88]: expert systems, natural language systems, and perception systems. "Expert systems are currently the most emphasized area in artificial intelligence" [Mill88]. Feigenbaum defines an expert system as "an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution" [Mill88]. Expert systems are currently being applied to a wide variety of areas such as management, simulation, manufacturing, construction, communication, medicine, and engineering. Recently, there has been a focus of interest on the use of expert systems in solving scheduling problems.

Scheduling problems in general are complex and the end result heavily influences the factors such as resource utilization, inventory cost, tardiness, and service level to customers. Scheduling environments can be either static or dynamic. Conventional scheduling techniques are static, that is, schedules are made assuming the scheduling environments are static [Arff90]. In real life most scheduling environments will have unexpected problems such as production machinery failure, causing the static scheduling methods to become outdated [Arff90]. Arff and Hasle [Arff90] suggested two solutions:

[1] Provide functionality for taking care of dynamic changes in the scheduling environment;

[2] Regenerate schedules completely when changes in the scheduling environment occur.

Of the two solutions cited, solution [2] would be possible only if the schedule generation time is small.

Scheduling problems can be solved by using heuristic-based methods and by using mathematical optimization methods. Heuristic-based methods use scheduling heuristics to prune the search space. In the mathematical optimization method, a mathematical model of the scheduling problem is formulated and solved. Solving these kinds of scheduling problems involves enumeration and search [Arff90]. "Scheduling is a computationally complex problem and thus, search techniques which deterministically and exhaustively search the possibility space will probably fail due to time requirements" [Sysw90]. This kind of problem is referred to as an NP-complete problem [Sysw90]. Heuristic-based methods and mathematical optimization methods should produce reasonably good solutions to scheduling problems.

Currently many research groups have expressed interest towards object-oriented programming. The principles of object-oriented programming appears to be more promising than conventional procedural programming. "Object-Oriented techniques provide extensive support for knowledge encapsulation, reusability, and data abstraction" [Kowa90]. Kowa and Stipp cited the following advantages of an approach using both object-oriented techniques and expert system techniques [Kowa90]:

[1] Increased programmer productivity,

[2] Reduced maintenance costs.

This thesis describes the implementation of a scheduling system which uses object-oriented techniques with the expert system technology.

CHAPTER II

LITERATURE REVIEW

In this section literature on AI techniques in manufacturing and other closely related work is reviewed.

Baker discussed OR, AI, and Data Base technologies and outlined the problems involved in integrating these technologies for production planning and scheduling applications [Bake90]. He compared the characteristics of AI and OR techniques and pointed out the relative strengths and weaknesses of these techniques. He concluded that production planning and scheduling problems are very complex and an AI/OR/DB integrated approach could use the advantages of all three techniques and produce efficient solutions to such problems.

The advantages of knowledge-based systems in solving problems such as scheduling were discussed by Lee in his paper on UNIK-R project experience [Lee90]. He described the implementation details of the project UNIK-R (UNIfied Knowledge-Refinery). The main feature of the project was the formulation of an optimization model for a monthly crude oil purchase plan. Other main features of this project were knowledge-based delivery scheduling and knowledge-based unload and charge scheduling for plant operations.

Selen and Heuts described an innovative approach for solving production scheduling problems [Sele90]. They developed an expert system which produces an initial optimal solution to the production scheduling problem. This initial optimal solution is improved by applying the scheduling heuristics developed by Selen and Heuts [Sele90] to produce a final solution to the scheduling problem. They concluded that expert systems coupled with heuristic approaches could be used to solve complex production scheduling problems, yielding efficient solutions.

Pargas and Peck developed a simulation software tool for assisting management in solving scheduling problems [Parg90]. The objective of production scheduling is to optimize operating a plant. The authors observed that in general "one cannot predefine an objective function for optimization" [Parg90]. The objective might vary at different points of time. The objective might be to maximize output at one point of time and to minimize inventory cost at another time [Parg90]. According to the authors, developing an expert system which will assist planners in solving scheduling problems is going to be their future topic of research.

Arff and Hasle discussed a technique for solving production scheduling problems [Arff90]. They described a heuristic-based algorithm for schedule synthesis. They compared heuristic search and mathematical programming techniques for solving scheduling problems. They also described the architecture of a scheduling system using heuristic, constraint-guided search. The authors suggested that future research should focus on heuristic, constraint-guided search techniques.

Czigler and Whitaker described a scheduling system for solving complex scheduling problems [Czig90]. It was a PC-based system and it used an algorithmic method for arriving at optimal solutions to scheduling problems. This system was implemented in a manufacturing company, and, as reported, was expanded for more generic use. The system was written in Pascal and its objective was to develop optimum production schedules in a reasonable amount of time. Based on their results, they concluded that solutions produced by heuristic-based methods are not optimal and an approach using mathematical programming methods would minimize the total production costs.

Kerr and Ebsary [Kerr90] described the implementation of an expert system for production scheduling. They used a heuristic-based approach. They identified the facts and heuristics used by the human scheduler and based on that arrived at a set of rules. They developed a rule-based expert system based on this set of rules. According to the authors the major problem they had was, "the difficulty of capturing a changing knowledge base in a highly dynamic environment" [Kerr90]. They concluded that a fully automated scheduling system should use an advanced knowledge base that would enable the system to adjust to the changes in its environment intelligently using its own facts and heuristics.

Kowalski and Stipp discussed "object processing for knowledge-based systems" [Kowa90]. Their article highlighted the advantages of using object-oriented techniques for solving problems such as scheduling. They cited two applications: Extruder and Ace Shipping [Kowa90]. Both applications used object-oriented techniques for solving manufacturing problems such as scheduling. The authors discussed the implementation details of both systems. The authors concluded that object-oriented techniques would play an important role in solving a wide variety of problems in the future [Kowa90].

Most of the expert scheduling systems implemented so far were based on procedural programming approaches. Object-oriented programming techniques appear to be more advantageous than procedural approaches. The expert scheduling system to be implemented will use an object-oriented approach so that the program is easier to visualize and much easier to expand and modify at a later date.

CHAPTER III

EXPERT SYSTEM DEVELOPMENT TOOL

This chapter discusses the expert system development tool KAPPA, developed by IntelliCorp [KAPP90], which is used to implement the expert system for production scheduling.

Introduction to KAPPA

KAPPA is an object-oriented application development tool that could be used for developing knowledge-based applications [KAPP90]. KAPPA runs under the Microsoft Windows environment. The basic component of the KAPPA knowledge base is an object [KAPP90]. KAPPA provides extensive application development features that include a variety of tools for building knowledge-based applications, a set of object-oriented programming tools that could be used to manipulate the knowledge base, features for rule-based programming, and a variety of graphic images that could be used for building the user interface to the knowledge-based system [KAPP90].

Main Features of KAPPA

In this section the main features of KAPPA such as Objects, Inheritance, Methods, KAL, and KAPPA Application Interface are discussed [KAPP90].

Objects

Objects are the basic components of KAPPA knowledge bases [KAPP90]. Objects within a KAPPA knowledge base could be defined either as Classes or Instances [KAPP90]. A class represents a group of objects; an instance is a specific object within a class. Objects can have several slots to represent the pieces of information related to the object. The slots describe the characteristics of the object. One or more values can be assigned to slots depending on whether the slot is a single valued slot or a multiple valued slot.

Inheritance

Inheritance is one of the important features of KAPPA knowledge bases [KAPP90]. Figure 1 shows a typical example of inheritance [KAPP90]. In Figure 1, the class sedans contains the slots Color, NumberofDoors, and Owner. Maryscar is an instance of the class sedans. The instance Maryscar inherits the slots from its parent class sedans. If the value for a slot is the same for all objects in a hierarchy, it could be assigned to the highest class in the hierarchy [KAPP90]. The slot value is then inherited by all the objects below it in the hierarchy. Subclasses inherit properties from their parent classes. Classes can have subclasses and each subclass can have several instances. This entire hierarchy is referred to as an object hierarchy [KAPP90]. Figure 2 shows an example illustrating object hierarchy [KAPP90]. Objects lower in the hierarchy inherit features from objects higher in the hierarchy. New classes can be built upon existing classes which makes creating new classes easier. To create a new class which is similar in attributes to an older class, the newclass could be created as a subclass of the older class. The new class inherits all the attributes of its parent class. One advantage of inheritance is that it simplifies building a knowledge base [KAPP90]. Modifying a knowledge base is also made simpler with inheritance [KAPP90]. If the attributes of a parent class are modified, the modifications are inherited by all the objects below the parent class in the hierarchy.

Methods

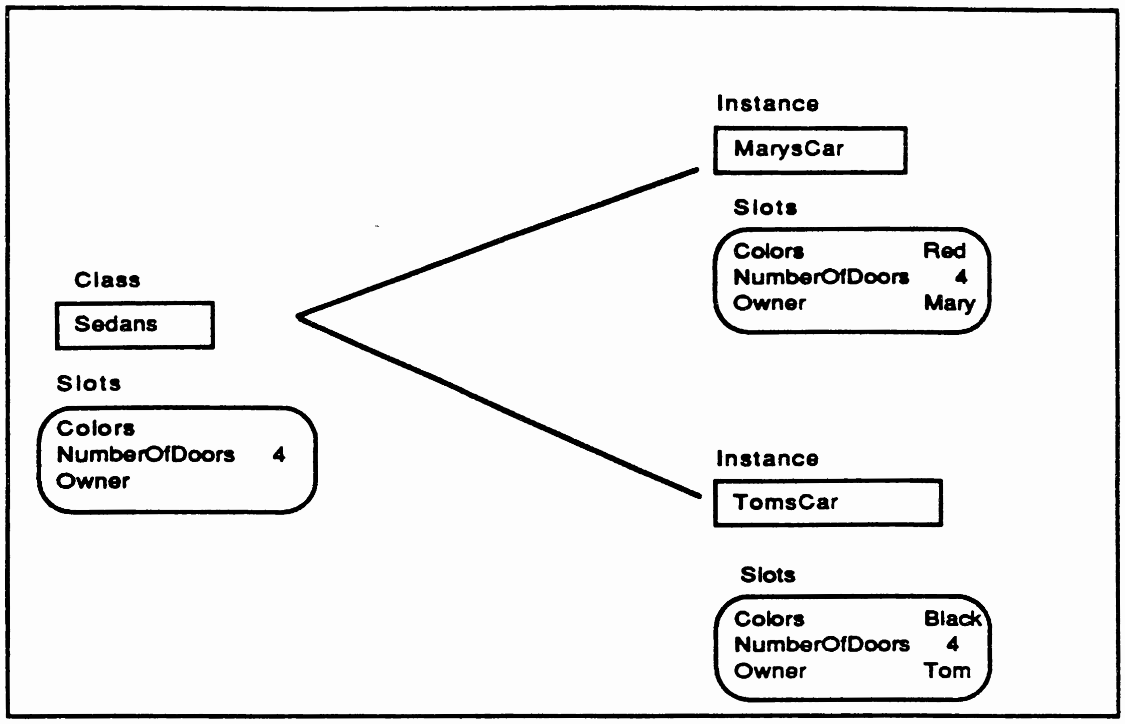
Methods can be generally defined as functions that work on the objects associated with it [KAPP90]. The objects in the knowledge base communicate by using methods. "Activating a method related to a particular object is referred to as sending a message" [KAPP90]. When activated, methods usually change the state of the objects associated with them. Methods could also perform functions such as activating other methods.

Methods associated with KAPPA objects could be classified into three types [KAPP90]:

[1] If Needed Method: This method is activated whenever a value for a slot associated with the object is required.

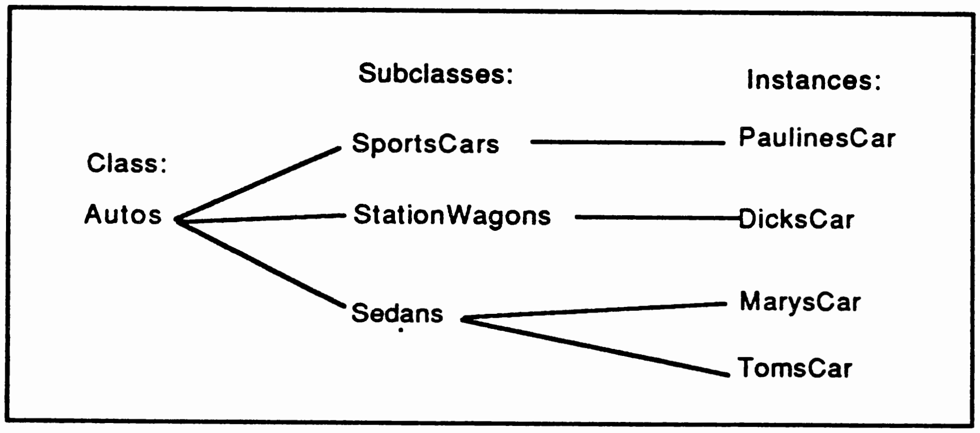
[2] Before Change Method: This method is activated just before a new value is assigned to the slot of the associated object.

[3] After Change Method: This method is activated just after a new value is assigned to the slot of the associated object.



Source: KAPPA - User's Guide, IntelliCorp, Inc.

Figure 1. Example Illustrating Inheritance



Source: KAPPA - User's Guide, Intellicorp, Inc.

Figure 2. Example Illustrating Object Hierarchy

The principle of inheritance also applies to methods [KAPP90]. Objects lower in the hierarchy inherit methods from objects higher in the hierarchy. Methods could also be made local to specific objects. KAPPA provides a method editor to create and edit methods.

KAL

KAPPA Language is KAPPA's application language which could be used for writing application programs. KAL provides functions for building and modifying the knowledge base more efficiently than using the KAPPA interface [KAPP90]. The KAPPA interface provides a KAL Interpreter which could be used for immediately evaluating KAL expressions.

KAPPA Language syntax comprise the following six types of divisions [KAPP90]:

[1] Atoms: Atoms are basic units of KAPPA Language. Atoms could be single words or words enclosed within quotation marks.

[2] Pairs: Pairs are also basic units of KAL and they refer to a slot value within an object. The format for a pair is Objectname:Slotname.

[3] Infix Expressions: Infix Expressions refer to KAL expressions separated by infix operators. The operators include value assignment operators, arithmetic operators, arithmetic test operators, text append operators, string test operators, and logic operators [KAPP90].

[4] Special Expressions: Special Expressions consists of KAL control functions. KAL control functions include: For, ForAll, While, Let, and If

[KAPP90]. The syntax of special expressions consists of a function name followed by parameters specific to the function.

[5] Block Expressions: Block Expressions refer to groups of KAL expressions separated by semicolons and enclosed within curly brackets.

[6] Function Calls: KAPPA provides a library of functions that could be used for performing a variety of tasks. KAL functions could be divided into nine categories [KAPP90]: Knowledge, Math, String, List, Logical, File, Control, Windows, and User functions. The syntax of a function call consists of the function name and a list of arguments separated by commas.

<u>KAPPA Application Interface</u>

KAPPA provides excellent features for building an interface between the application and the end-user. KAPPA provides several functions for performing specific tasks while building the user interface. There are functions for showing and hiding windows, changing position of windows, removing windows, and resetting windows [KAPP90]. KAPPA also provides facilities for enhancing the user interface by using pop-up menus and dialog boxes. Application graphics could be added to the user interface by using the graphical presentation tools in KAPPA [KAPP90]. The graphical presentation tools could be used for building custom graphic objects. Each graphic object is referred to as an image [KAPP90]. Graphical images could be used for the following purposes [KAPP90]: displaying static information, displaying information from single valued slots, displaying information from multiple valued slots, etc. The graphical presentation tools provide features for building the following ten different types of images [KAPP90]: Text images, Transcript images, Drawing images, Bitmap images, Button images, State Box images, Meter images, Slider images, Edit images, and Line Plot images. KAPPA has an image class and each of the above images is a subclass of the image class. KAPPA Language provides functions for creating and editing images. Each user-created image is saved as an instance of the corresponding image subclass. Functions could also be assigned to images so that specific tasks could be performed when the user clicks on that image. KAPPA provides facilities for accessing ASCII files. KAPPA Language provides functions for reading from ASCII files and writing to ASCII files. KAPPA provides a good interface to other software such as Lotus and dBase. These features could be used for developing a powerful integrated applications.

KAPPA Advanced Features

In addition to the features mentioned above, KAPPA also provides certain advanced features.

KAPPA has a powerful spreadsheet and database interface. KAPPA has features for directly reading a Lotus 1-2-3 spreadsheet or a dBase file. The user could also write to a Lotus 1-2-3 file or a dBase file. The KAPPA Language has the following functions for handling spreadsheet and database files [KAPP90]: DBOpenFile, DBSelectFile, DBCloseFile, DBReadCell, DBGetNumberOfRows, DBGetNumberOfFields, DBGetRowPosition, DBSetRowPosition, DBWriteCell, and DBFindRecord.

KAPPA also provides powerful features for controlling rules such as the setting break points in rules and selective evaluation of rules [KAPP90]. Another powerful feature of KAPPA is its source code interface. KAPPA provides facilities for the user to add their own functions, written in C, to the KAPPA source code to enhance its features.

CHAPTER IV

TECHNIQUES FOR SCHEDULING

Linear Programming-Based Approach

"Linear Programming is a mathematical procedure for determining optimal allocation of scarce resources" [Schr86]. The linear programming technique is a traditional OR technique for solving problems which could be stated using linear equations. The general form of a linear equation is as follows:

$$p_1x_1 + p_2x_2 + p_3x_3 + \ldots\ldots\ldots \qquad + p_nx_n = y$$

where the p_i's are known coefficients and the x_i's are unknown variables.
A linear programming problem consists of a linear function which is referred to as an objective function and a set of linear equations which represents the constraints of the problem. Solving a linear programming problem consists of finding a solution that optimizes the objective function. Any problem, if formulated as a linear programming model, could be solved to obtain an optimal solution if an optimal solution exists for that problem.

The scheduling problem in general could be stated as, "given a set of orders, each defined as a set of (resource, duration) pairs, and resources, allocate the orders to resources and time such that a set of requirements are met" [Arff90]. The objective in most real life scheduling problems involves maximizing or minimizing some linear combination of production variables subject to a set of production constraints. In the linear programming-based approach to be implemented, an LP model for the scheduling problem is formulated by specifying an objective function for optimization and posing a set of production constraints to the schedule. This linear programming model is solved to generate an initial optimal solution to the scheduling problem. An expert system operates on this initial optimal solution and generates a schedule which satisfies various production and packaging constraints. This recommended schedule is presented to the user. The user is given the option to either accept the schedule or modify the schedule. If the recommended schedule is not acceptable, the expert system guides the user in modifying the schedule.

Heuristic-Based Approach

"A heuristic is a technique that improves the efficiency of a search process, possibly by sacrificing claims of completeness" [Rich87]. Heuristic-based approaches could be used to solve complex problems such as scheduling. In production scheduling problems the goal is to find the best possible schedule from the set of all possible schedules. In real life, production scheduling problems are complex because of the extremely large number of possible schedules. In heuristic-based approaches for solving scheduling problems, scheduling heuristics are used to build the schedule. The scheduling heuristic that has been used in this heuristic-based approach consists of prioritizing the products to be scheduled based on their ABC code and the difference between the safety stock and inventory on-hand values for that product. The products are then scheduled one by one, starting from the highest priority product.

An initial optimal schedule is generated from this heuristic-based approach. An expert system module operates on this initial optimal schedule generated by the heuristic-based approach and produces a recommended schedule which satisfies various production and packaging constraints. The expert system presents the recommended schedule to the user. If the recommended schedule is not acceptable for some reason, the expert system guides the user in modifying the schedule.

Thomas E. Baker [Bake90] analyzed AI and OR techniques for solving scheduling problems. He compared the features of both techniques as illustrated in Table I [Bake90].

TABLE I

COMPARISON OF AI AND OR TECHNIQUES

Operations Research	Artificial Intelligence
Well Structured	Ill Structured
Exact	Inexact
Quantitative	Qualitative
Numerical	Inferential
Optimizing	Satisficing

CHAPTER V

EXPERT SYSTEM IMPLEMENTATION DETAILS

Background on the Scheduling Environment

Purolator Products, Inc. currently produces about 305 models of filters in 13 different generic product families. The product manufacturing line produces 13 generic filter families which are then packaged into one of 305 models of filters for shipment to customers. Currently, production scheduling in Purolator is being done manually. This process of manual scheduling is very complex and results in high inventory costs and inefficient schedules. Also, manual scheduling leads to back-order situations which results in poor customer service. The main objective of the proposed expert scheduling system is to reduce inventory costs and produce optimum production schedules. The expert system is designed to perform automatic scheduling of the product manufacturing line and the product packaging line. In the proposed scheduling system, both the product packaging line and the product manufacturing line are scheduled based on Just-In-Time technique in which the goal is to replenish the safety stock levels. The safety stock is used to meet customer order requirements. The rules and constraints used in the scheduling system are formulated based on interviews with the human schedulers at Purolator. The proposed scheduling system is designed for use on a production and packaging line at Purolator Products, Inc. in Fayetteville, North Carolina.

The Data Base Module

The data base module of the scheduling system is implemented using Paradox, a relational database management tool, a product of Borland. The data for the scheduling system is downloaded from Purolator's mainframe databases and fed to the database on a PC. The data include the following:

[A] Bill of Materials information for the products

[B] Inventory information

[C] Shipments information

[D] Backorder information

[E] Resources information

[F] Safety Stock, Forecasts, Orders etc.

Data such as Bill of Materials information which are static are downloaded and input only when there is a change in the Bill of Materials information. The other data are downloaded on a daily basis and input to the scheduling system. At the start of each scheduling session the user has provision to update Bill of Materials information if required. Otherwise, the system automatically transfers the data it needs on a daily basis. The system maintains the product inventory and updates it on a daily basis after each production. Before the scheduling system is run, the user is given the option to confirm the previous day's production schedule maintained by the database module. The database module manages the data and provides it to the optimization module which generates an initial optimal schedule. The system automatically updates the raw material usage for each of the raw materials after each run.

Linear Programming Approach Implementation Details

In this section the implementation details of the linear programming based version of the expert system is discussed. The major components of the system are shown in Figure 3. In this version of the expert system the optimization module uses a linear programming approach to arrive at an initial optimal solution to the scheduling problem. This process of generating an optimal solution to the scheduling problem using a linear programming approach consists of two steps. The first step is to formulate a linear programming model of the scheduling problem and the second step is to solve the linear programming model to obtain an optimal solution. The database module manages the data for the expert system and provides it to the optimization module. The optimization module is essentially comprised of two integrated modules, one module written in C which reads in all the scheduling data and generates a linear programming model for the scheduling problem and the other module which solves the linear programming problem using a linear programming application LINDO to arrive at an optimal solution. The linear programming problem generated by the first part of the optimization module is represented in Mathematical Programming System (MPS) format [Schr86]. This format is chosen to establish an interface between the C module and the linear programming application module.

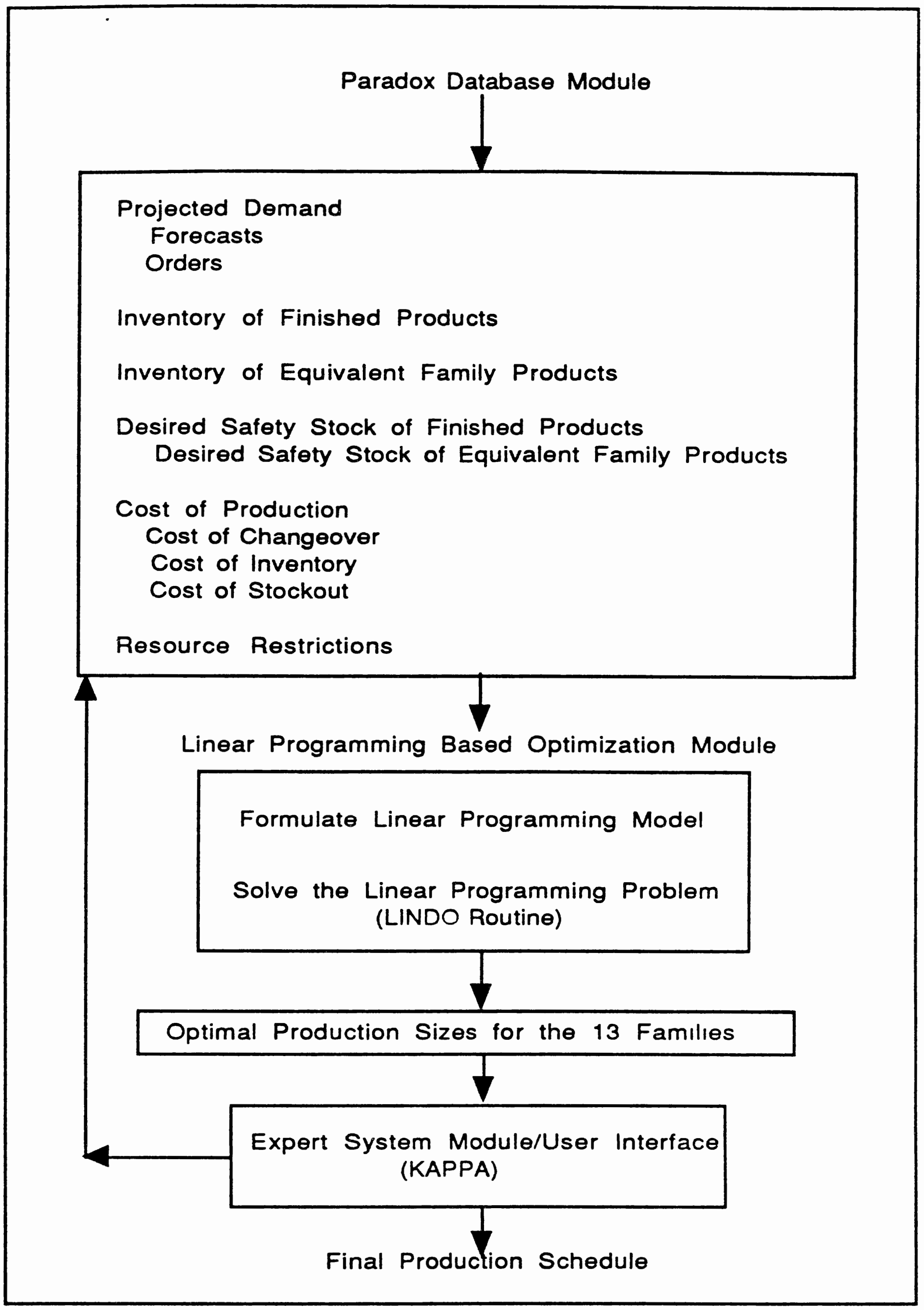

Paradox Database Module
Projected Demand
Forecasts
Orders
Inventory of Finished Products
Inventory of Equivalent Family Products
Desired Safety Stock of Finished Products
Desired Safety Stock of Equivalent Family Products
Cost of Production
Cost of Changeover
Cost of Inventory
Cost of Stockout
Resource Restrictions
Linear Programming Based Optimization Module
Formulate Linear Programming Model
Solve the Linear Programming Problem
(LINDO Routine)
Optimal Production Sizes for the 13 Families
Expert System Module/User Interface
(KAPPA)
Final Production Schedule

Figure 3. Block Diagram Showing Components of the Linear Programming Based System

Linear Programming Formulation of the Scheduling Problem

This section describes the linear programming formulation of the scheduling problem. One of the major objectives of the scheduling system is to maximize the total number of products scheduled. Priority is given to products based on their distance from safety stock levels. The objective function for the LP formulation of the scheduling problem is of the form:

$$\text{MAXIMIZE:}\quad D_1P_1 + D_2P_2 + D_3P_3 + \ldots\ldots\ldots \qquad + D_{305}P_{305}$$

where

$$D_i = ((\text{SafetyStock}_i - \text{Inv}_i)/(\text{SafetyStock}_i)) * 100 * \text{ABCcode}_i$$

for $i = 1,2,3, \ldots\ldots\ldots,305$.

Constraints to the scheduling problem consists of resource constraints for each resource and maximum production size constraints for each product.

The products use four types of resources: cartons, shipping containers, labels, and equivalent families. There are 182 types of cartons, 52 types of shipping containers, 20 types of labels, and 13 types of families. The LP problem has 182 constraints for cartons, 52 constraints for shipping containers, 20 constraints for labels, and 13 constraints for families giving a total of 267 resource constraints. The resource constraints are formulated in the following explanation. For example, if products P_1, P_3, P_5, and P_{300} use carton C_1, shipping container S_2, label L_3, and Family F_4, then the resource constraints would be of the form:

$$P_1 + P_3 + P_5 + P_{300} <= InvC_1$$

$$P_1 + P_3 + P_5 + P_{300} <= InvS_2$$

$$P_1 + P_3 + P_5 + P_{300} <= InvL_3$$

$$P_1 + P_3 + P_5 + P_{300} <= InvF_4$$

where $InvC_1$ is the inventory of carton type C_1, $InvS_2$ is the inventory of shipping container type S_2, $InvL_3$ is the inventory of label type L_3, and $InvF_4$ is the inventory of family type F_4. The constraints for each of the resource type is formulated in this fashion based on the products using that resource type.

In addition to the resource constraints, there are constraints on maximum production sizes for each product; i.e., there are 305 production constraints giving a total of 572 constraints for the linear programming problem. Since the scheduling is based on replenishing the safety stock levels, the maximum production size for each product is computed in the following manner:

If ($SafetyStock_i$ - $Invonhand_i$) > 0 then the maximum production size is the difference between the safety stock and inventory on hand;

If ($SafetyStock_i$ - $Invonhand_i$) < 0 then the maximum production size is 0. The complete linear programming formulation of the scheduling problem and its equivalent MPS representation is listed in the appendix.

The optimization module formulates a linear programming model of the scheduling problem and represents it in MPS format. Linear, INteractive, Discrete Optimizer (LINDO) program solves the linear programming problem formulated by the first part of the optimization module and generates an optimal solution to the scheduling problem.

Heuristic-Based Approach Implementation Details

In this section the implementation details of the heuristic-based version of the expert system is discussed. In this version of the expert system the optimization module uses a heuristic-based approach to arrive at an initial optimal solution to the scheduling problem. The major components of this version of the scheduling system are shown in Figure 4. The database module manages all the data for the expert system and provides it to the heuristic-based optimization module in an ASCII quote delimited format. The optimization module is written in C and utilizes the data provided by the Paradox database module.

The optimization module uses sequencing heuristics to reduce the complexity involved in solving the scheduling problem. The sequencing heuristics are very efficient because they impose an ordering to the set of products to be scheduled, which reduces the complexity involved in sequencing the products. The process of generating an initial optimal solution to the scheduling problem using this heuristic-based approach also consists of two steps. The first step is to apply sequencing heuristics to the set of products to be scheduled to impose an ordering among the products. The second step is to traverse the ordered set of products and generate a schedule satisfying the resource and other constraints.

The sequencing heuristic used in this approach is based on the "farthest away from the safety stock first heuristic". We define a metric called distance which refers to the percentage difference between the safety stock and the inventory onhand quantity. The initial set of products is traversed and the

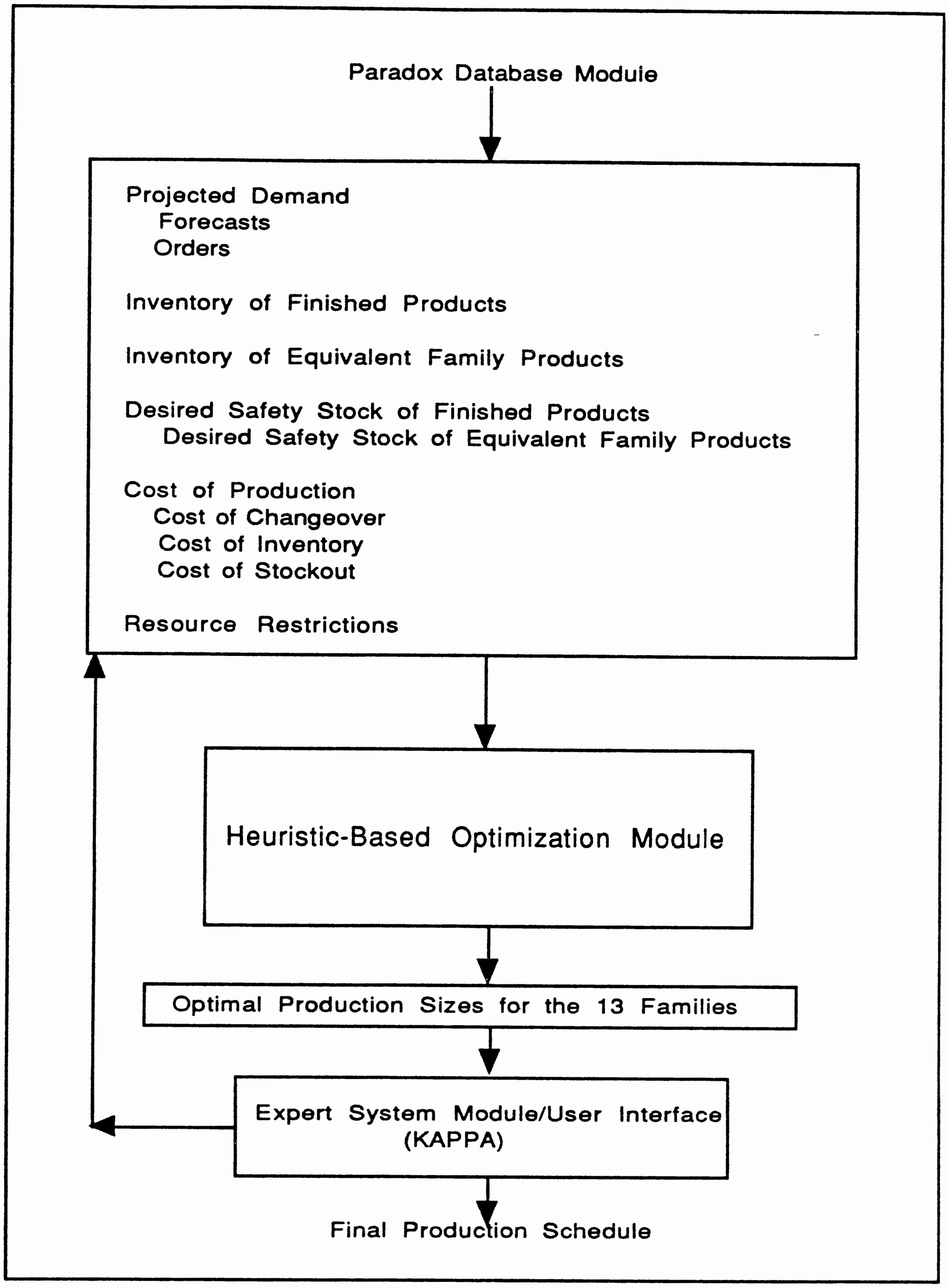

Paradox Database Module
Projected Demand
Forecasts
Orders
Inventory of Finished Products
Inventory of Equivalent Family Products
Desired Safety Stock of Finished Products
Desired Safety Stock of Equivalent Family Products
Cost of Production
Cost of Changeover
Cost of Inventory
Cost of Stockout
Resource Restrictions
Heuristic-Based Optimization Module
Optimal Production Sizes for the 13 Families
Expert System Module/User Interface (KAPPA)
Final Production Schedule

Figure 4. Block Diagram Showing Components of the Heuristic-Based System

distance measure is computed for each product. Each product has an associated ABC code which classifies them according to their profit potential. The distance measure computed for each product is enhanced by using the ABC code of that product. The distance measure D is computed for each product as follows:

$$D_1 = ((SafetyStock_1 - Inv_1)/SafetyStock_1) * 100$$

The enhanced distance ED for each product is computed as follows:

$$ED_1 = D_1 * ABCcode_1$$

The set of products is then ordered based on its enhanced distance measure.

After the set of products is ordered based on its enhanced distance measure, the next step is to traverse the ordered set of products to generate an initial schedule. The ordered set of products is traversed and the products are scheduled one by one, starting with the product with the highest enhanced distance measure and subject to the resource requirements. Thus an initial optimal schedule is generated using this heuristic-based approach. This initial optimal schedule is passed on to the expert system module which generates the final production schedule.

Expert System Module and User Interface Details

In this section the implementation details of the expert system module and the user interface part of the scheduling system are discussed. The expert system module is written using KAPPA, an object-oriented expert system shell.

Figure 5 shows the organization of the objects in the knowledge base. The knowledge base consists of two major classes: products and resources. There are

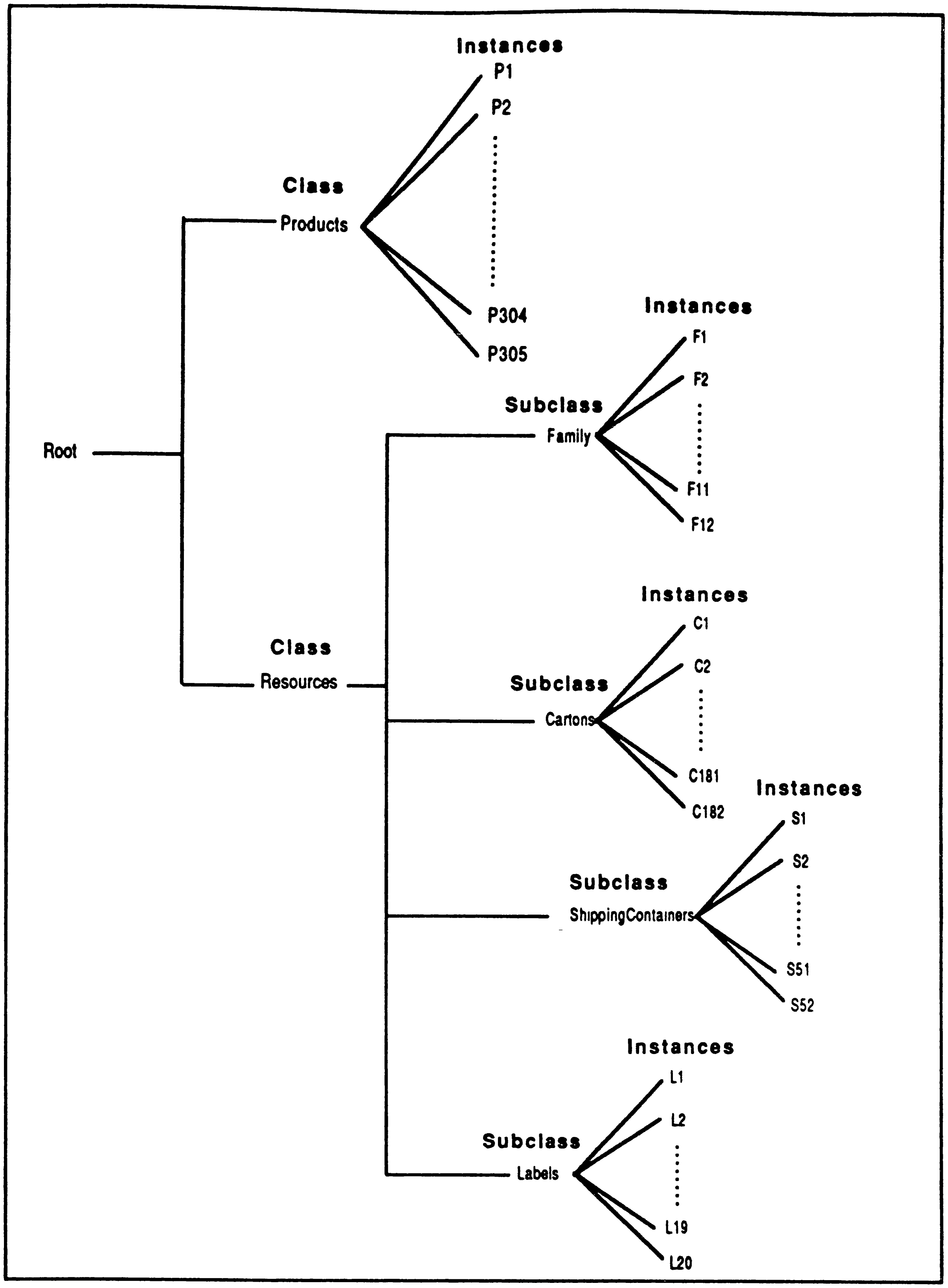


Figure 5. Organization of Objects in the Knowledge Base

four different types of resources that are used by the products to be scheduled: Carton, Shipping Container, Label, and Equivalent Family. Each of these resources is defined as a subclass of the class resources. There are 305 different products to be scheduled and each of them is defined as an instance of the class products. Similarly, there are 13 types of resource families, 182 types of resource cartons, 52 types of resource shipping containers, and 20 types of resource labels. Each of these is defined as an instance of their corresponding resource subclass.

The products class contains slots, for specifying the information related to each product, such as:

Abccode (text)

AverageMonthlyForecast (numeric)

BackOrder? (boolean)

BackOrderQuantity (numeric)

Cartonnumber (text)

ContainerUsage (numeric)

FamilyCode (text)

Labelnumber (text)

PackingType (text)

Partnumber (text)

Priority (numeric)

Productionsize (numeric)

SafetyStock (numeric)

Seqnumber (numeric)

ShippingContainerNumber (text)

The slots for the products class are defined once in the main class products and are inherited by all its instances. Similarly, the resource class contains slots, for specifying the information related to each resource such as the following:

Available (numeric)

PartnumbersUsingThisType (list)

Inventory (numeric)

PlannedReceipts (numeric)

PlannedUsage (numeric)

ProjectedInventory (numeric)

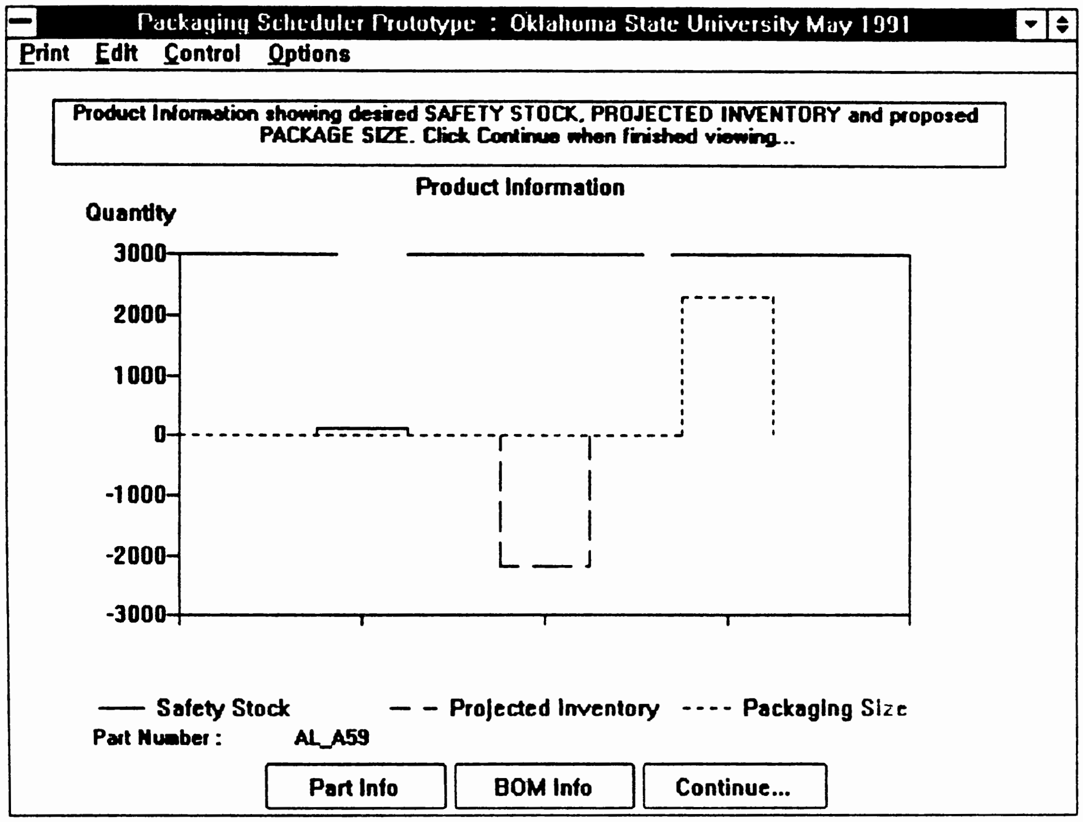
ProjectedUsage (numeric)

Type (text)

These slots are defined in the class resources and are inherited by all its subclasses and their instances.

The scheduling system keeps track of the recommended production sizes of products for each run and updates the raw materials usage on a daily basis. The expert system starts execution by first presenting the previous day's production sizes of each product to the user. The user should confirm the production size for all the products. The user also has the option to change the size, if what is actually produced is different from what the system recommends. The system uses these confirmed production sizes to project inventories of the products and resources for the current run. The system then calls the optimization module for generating an initial schedule. The implementation details of the linear programming based optimization module and heuristic-based optimization module were discussed in the previous sections. The initial schedule information and other relevant product and resource information are loaded into the instances of products and resource classes. After loading the instances, the system generates a recommended schedule by firing the sequencing rules and production rules built into the system. At each step of the schedule generation all resource and production constraints are taken into consideration. The system presents the generated schedule to the user with specific information such as the sequence number, equivalent family type, scheduled quantity, packaging type, and back order information. The user also has the option to view information on products that were not scheduled. The system has features for displaying product information graphically. Figure 6 shows a sample screen from the system displaying product information graphically.

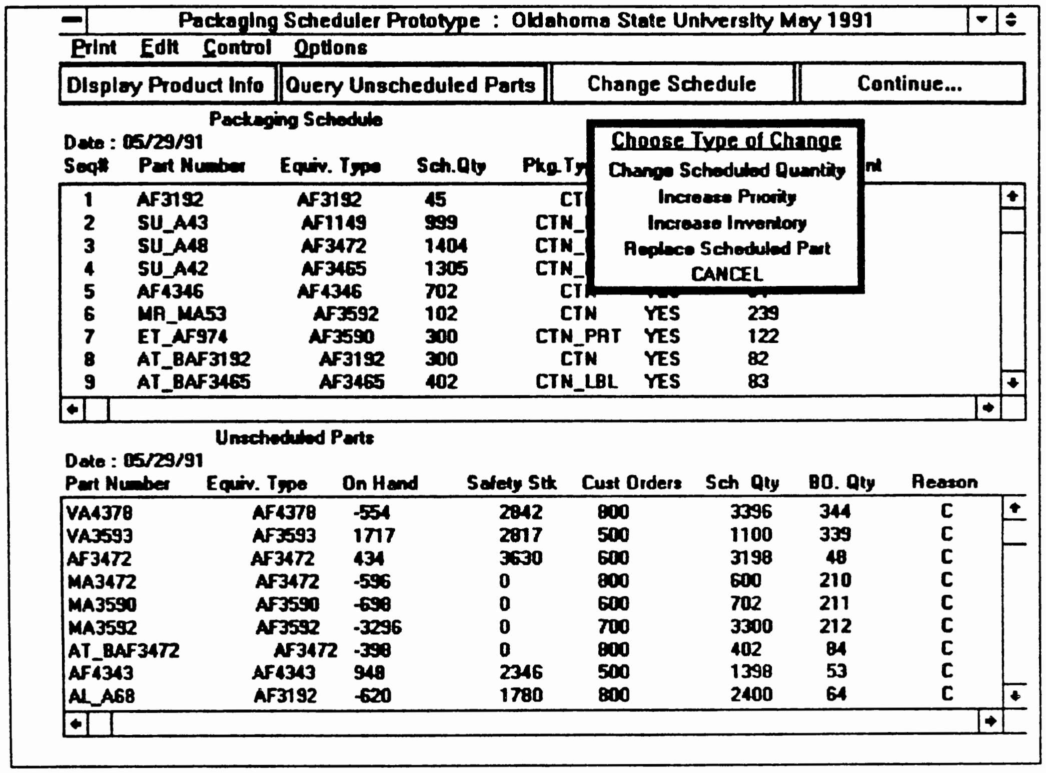
Another important feature of the system is that it provides the user with the capability to query about those products that were not scheduled due to resource shortages. If the schedule generated by the expert system is not appropriate for some reason, the user is given the option to change the schedule. If the user decides to change the recommended schedule, the expert system guides the user in changing the schedule. Figure 7 shows a sample screen from the system illustrating the ways the user could change the recommended schedule. The user has options to change quantity of a schedule product, increase priority of a product, increase inventory of a resource, and replace scheduled product for an unscheduled product. The expert system guides the user in changing the schedule



Product Information showing desired SAFETY STOCK, PROJECTED INVENTORY and proposed PACKAGE SIZE. Click Continue when finished viewing...

Part Number : AL_A59

Figure 6. Sample Screen From the System Displaying Product Information Graphically



Packaging Schedule

Date : 05/29/91

Seq#	Part Number	Equiv. Type	Sch.Qty	Pkg.Ty		nt
1	AF3192	AF3192	45	CT		
2	SU_A43	AF1149	999	CTN_		
3	SU_A48	AF3472	1404	CTN_		
4	SU_A42	AF3465	1305	CTN_		
5	AF4346	AF4346	702	CT		
6	MR_MA53	AF3592	102	CTN	YES	239
7	ET_AF974	AF3590	300	CTN_PRT	YES	122
8	AT_BAF3192	AF3192	300	CTN	YES	82
9	AT_BAF3465	AF3465	402	CTN_LBL	YES	83

Unscheduled Parts

Date : 05/29/91

Part Number	Equiv. Type	On Hand	Safety Stk	Cust Orders	Sch Qty	BO. Qty	Reason
VA4378	AF4378	-554	2842	800	3396	344	C
VA3593	AF3593	1717	2817	500	1100	339	C
AF3472	AF3472	434	3630	600	3198	48	C
MA3472	AF3472	-596	0	800	600	210	C
MA3590	AF3590	-698	0	600	702	211	C
MA3592	AF3592	-3296	0	700	3300	212	C
AT_BAF3472	AF3472	-398	0	800	402	84	C
AF4343	AF4343	948	2346	500	1398	53	C
AL_A68	AF3192	-620	1780	800	2400	64	C

Figure 7. Sample Screen From the System Illustrating the Ways User Could Change the Recommended Schedule

by relaxing resource and other constraints. Once the user is satisfied with the schedule, a final schedule report containing detailed information on scheduled and unscheduled products could be printed out for use by production personnel.

CHAPTER VI

SCHEDULING SYSTEM RESULTS AND CONCLUSIONS

Scheduling System Results

The performance of the linear programming-based system and heuristic-based system are compared by measuring a performance metric, Total Number of Products Scheduled, for a test run. This performance metric is chosen because one of the primary objectives of the scheduling system is to schedule as many products as possible. Figure 8 illustrates the performance of both systems based on the above metric. The total number of products scheduled with linear programming based system was 56 and in the case of the heuristic-based system, the total number of products scheduled was 47.

The following software metrics were also measured for both versions of the system [Cont86]: Lines of Code, Function Count, and Decision Count. The number of lines of code specifies the number of program lines excluding comment lines and blank lines [Cont86]. The function count specifies the number of individual functions in the program [Cont86]. The decision count specifies the total number of conditional and loop statements in the program such as If, While, and For [Cont86]. Figure 9 illustrates the comparison of the software metrics measured for both systems.

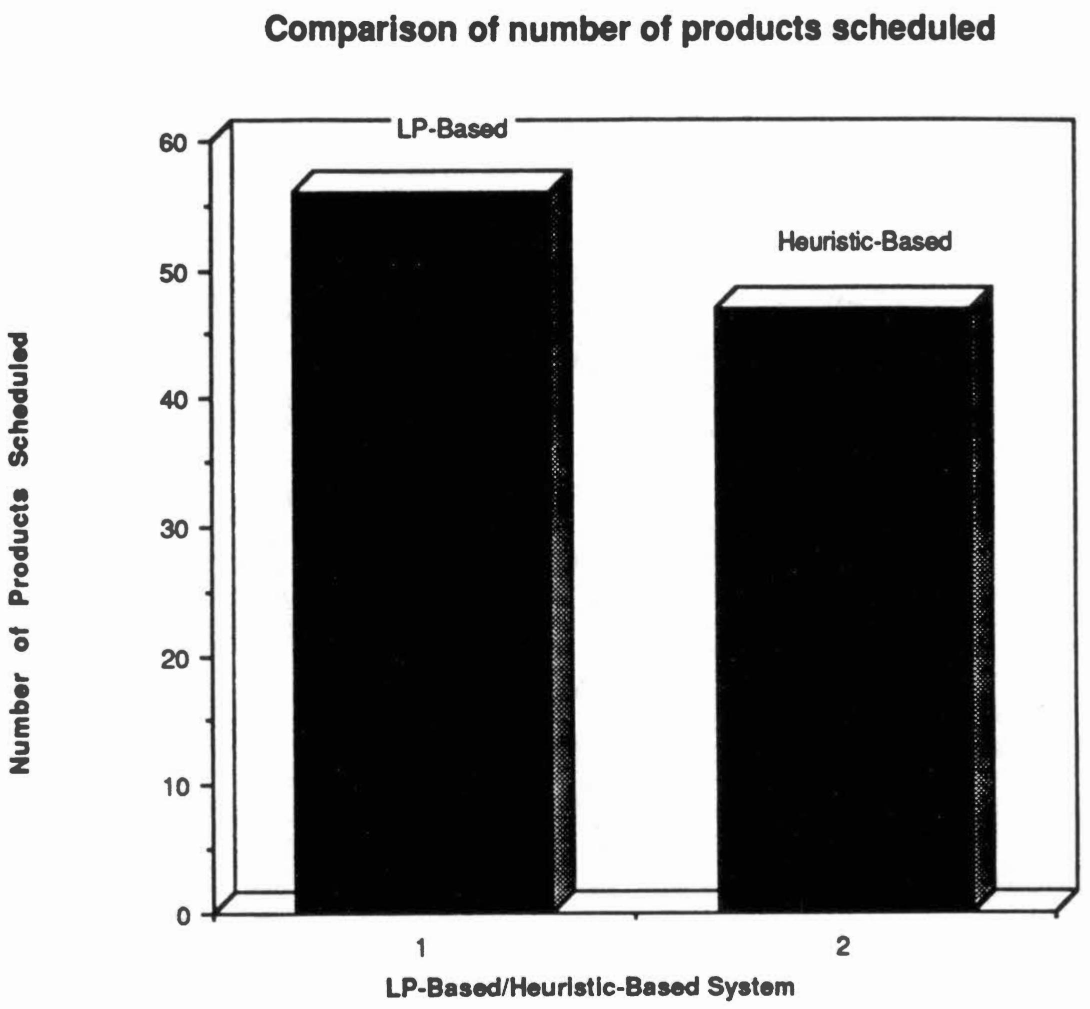


Figure 8. Comparison of Number of Products Scheduled for Both Systems

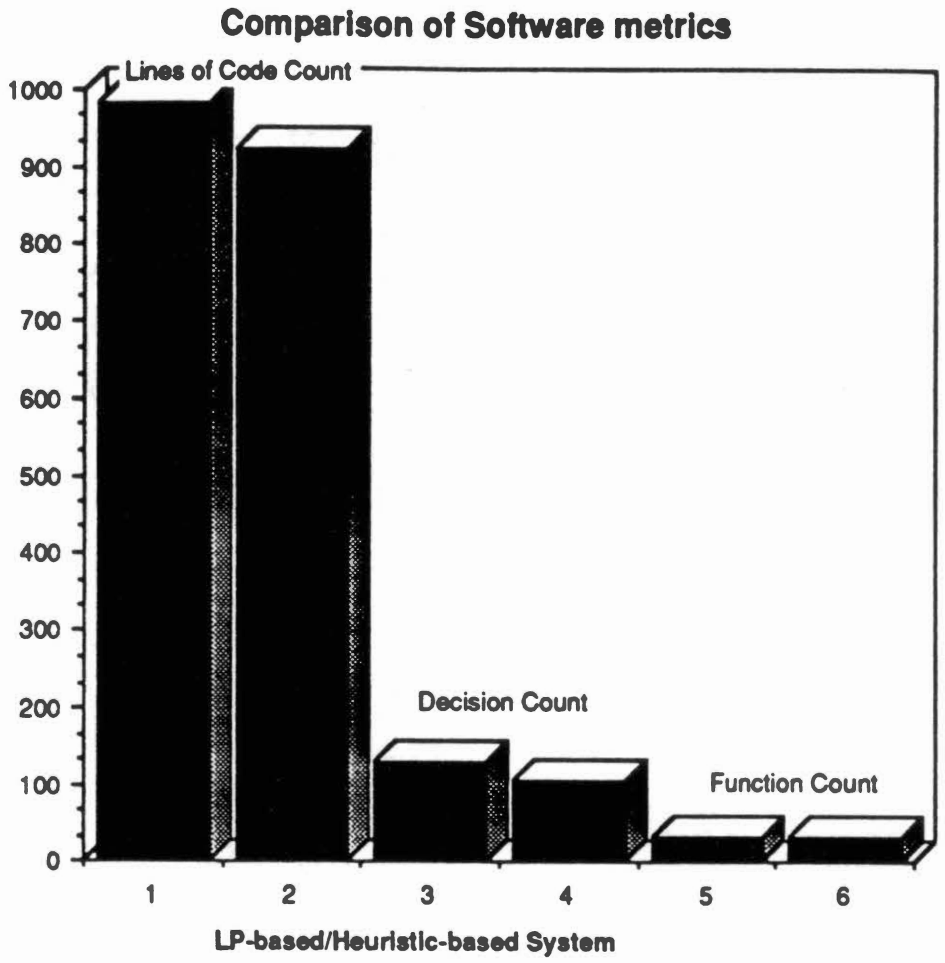


Figure 9. Comparison of Software Measures of Both Systems

	LP-based System	Heuristic-based System
Lines of Code:	987	929
Function Count:	32	30
Decision Count:	131	107

Conclusions

Linear programming-based techniques and heuristic-based techniques are two major techniques for solving scheduling problems. There have been quite a few expert scheduling systems implemented in the past using either of these techniques. Most of these systems were built based on a procedural programming approach.

This thesis has described the implementation of two versions of an expert scheduling system, one version using a linear programming technique and the other version using a heuristic-based technique for solving the scheduling problem. The expert scheduling system was built using an object-oriented programming approach so that the program is conceptually easier to visualize and much easier to expand and modify if required. In the version based on the linear programming technique, a linear programming model of the scheduling problem is formulated and solved to obtain an initial optimal solution and an expert system module operates on this optimal solution until an acceptable solution is reached. In the version based on the heuristic technique, scheduling heuristics are used to arrive at an initial optimal solution on which an expert system module operates to arrive at an acceptable solution. It is shown that linear programming techniques and heuristic-based techniques could be used with expert system techniques for solving complex scheduling problems. It is also shown that even though the heuristic-based approach produce acceptable solutions, the solutions are not always optimal. Linear programming-based approaches are truly optimal and can yield realistic solutions to problems such as scheduling. An integrated approach such as this could result in substantial savings in inventory costs and improvements in service level to customers. The expert scheduling system has been implemented for use at Purolator Products, Inc., Tulsa, Oklahoma.

Suggestions for Future Work

At present, the scheduling system does not have the capability to rerun the linear programming based optimization module after the user has chosen to modify the recommended schedule. In the future, the capability could be provided to rerun the linear programming based optimization module after the user has modified the priorities and inventory information. The scheduling system at present does not have a fully functional help capability. Context sensitive help capability could be added to the scheduling system to make it more easier for novice users of the system.

REFERENCES

[Arff90]
Stig Arff and Geir Hasle, "Synthesis of Schedules Using Heuristic, Constraint Guided Search", Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, pp. 111-124, 1990.

[Bake90]
Thomas E. Baker, "Integrating AI/OR/DataBase Technologies for Production Planning and Scheduling", Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, pp. 101-110, 1990.

[Brow90]
George H. Brown, "Commonizing Scheduling: Artificial Intelligence and Other Technologies in Generic Roles", Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, pp. 168-174, 1990.

[Cont86]
S.D. Conte, H.E. Dunsmore, and V.Y. Shen, Software Engineering Metrics and Models, The Benjamin/Cummings Publishing Company, Inc (1986).

[Czig90]
Martin H. Czigler and Clinton R. Whitaker, "A Hybrid Algorithmic and Knowledge Based Implementation for Workcenter Based Production Scheduling", Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, pp. 145-155, 1990.

[KAPP90]
KAPPA Users Guide, IntelliCorp, Inc.

[Kerr90]
R.M. Kerr and R.V. Ebsary, "Implementation of an Expert System for Production Scheduling", European Journal of Operations Research, Vol. 33, pp. 17-29, November 1988.

[Kowa90]
Bernadette Kowalski and Lori Stipp, "Object Processing for Knowledge-Based Systems", AI Expert, October 1990, pp. 34-41.

[Lee90]
Jae K. Lee, Sang B. Oh, Min S. Suh, Min Y. Kim, and Yong U. Song, "A Knowledge Network for Planning and Control of the Refinery Industry: The UNIK-R project Experience", Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, pp. 16-32, 1990.

[Mill88]
Richard K. Miller, and Terri C. Walker, Artificial Intelligence Applications in Manufacturing, SEAI Technical Publications (1988).

[Parg90]
Roy C. Pargas and John C. Peck, "Production Scheduling through Distributed Simulation", Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, pp. 90-94, 1990.

[Rauc87]
Wendy B. Rauchhindin, A Guide to Commercial Artificial Intelligence, Prentice-Hall Publications (1987).

[Rein90]
Kenneth F. Reinschmidt, John H. Slater, and Gavin A. Finn, "Expert Systems for Plant Scheduling using Linear Programming", Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, pp. 198-211, 1990.

[Rich87]
Elaine Rich, Artificial Intelligence, McGraw-Hill Publications (1987).

[Schr86]
Linus Schrage, Linear, Integer and Quadratic Programming with LINDO, The Scientific Press (1986).

[Sele90]
William J. Selen, Ruud M. Heuts and Willem Van Groenendaal, "Expert System and Operational Production Planning in the Manufacturing of Chemicals: A Hybrid Approach", Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, pp. 47-62, 1990.

[Sysw90]
Gilbert Syswerda and Daniel Cerys, "Knowledge-Based Genetic Search in Schedule Optimization", Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, pp. 125-133, 1990.

APPENDIXES

APPENDIX A

SOURCE CODE LISTING FOR USER DEFINED FUNCTIONS IN KAPPA

```
/********************************************************/
/**        ALL FUNCTIONS ARE LISTED BELOW              **/
/********************************************************/

     /*************************************
      **** FUNCTION: Readmainframedata
      *************************************/
MakeFunction( Readmainframedata, [],
   {
   Global:Temp1 = prodout1.c;
   PostBusy( ON, "Please Wait...Reading BOM Info" );
   OpenReadFile( Global:Temp1 );
   ReadWord( 144 );
   ReadWord( 144 );
   ClearList( Global:Parts );
   For x [1 305 ]
      {
      Global:Temp1 = ReadWord( );
      AppendToList( Global:Parts, Global:Temp1 );
      If Not( Instance?( Global:Temp1 ) )
        Then MakeInstance( Global:Temp1, Products );
      Global:Temp1:Partnumber = Global:Temp1;
      Global:Temp1:Familycode = ReadWord( );
      Global:Temp1:Abccode = ReadWord( );
      Global:Temp1:Ctndie = ReadWord( );
      Global:Temp1:Cartonnumber = ReadWord( );
      Global:Temp1:ShippingContainerNumber = ReadWord( );
      Global:Temp1:ContainerUsage = ReadWord( );
      Global:Temp1:Labelnumber = ReadWord( );
      Global:Temp1:AverageMonthlyForecast = ReadWord( );
      Global:Temp1:SafetyStock = ReadWord( );
      Global:Temp1:StandardCost = ReadWord( );
      Global:Temp1:PackingType = ReadWord( );
      };
   CloseReadFile( );
   PostBusy( OFF );
   } );

     /*************************************
      **** FUNCTION: Makeproductinstances
      *************************************/
MakeFunction( Makeproductinstances, [],
   {
   For x [1 305 ]
      {
      Global:Temp1 = P # x;
      MakeInstance( Global:Temp1, Products );
      };
   } );

     /*************************************
      **** FUNCTION: Readprodsizes
      *************************************/
MakeFunction( Readprodsizes, [],
   {
   Global:Temp1 = prodout2.c;
   OpenReadFile( Global:Temp1 );
   PostBusy( ON, "Please Wait... Transferring Packaging Sizes" );
   For x [1 305 ]
      {
      Global:Temp1 = ReadWord(  );
      Global:Temp1:ProductionSize = ReadWord(  );
      };
   CloseReadFile(  );
   PostBusy( OFF );
   } );

      /*************************************
      ****  FUNCTION: Showscheduledparts
      *************************************/
MakeFunction( Showscheduledparts, [],
   {
   ClearTranscriptImage( Transcript1 );
   ShowImage( Transcript1 );
   DisplayFile( Transcript1, finalsch.c );
   } );

      /*************************************
      ****  FUNCTION: Finishviewingschedule
      *************************************/
MakeFunction( Finishviewingschedule, [],
   {
   ClearTranscriptImage( Transcript1 );
   HideImage( Transcript1 );
   } );

      /*************************************
      ****  FUNCTION: Showunscheduledparts
      *************************************/
MakeFunction( Showunscheduledparts, [],
   {
   ShowImage( Transcript1 );
   DisplayFile( Transcript1, unschp.c );
   } );

      /*************************************
      ****  FUNCTION: Closeunscheduledlist
      *************************************/
MakeFunction( Closeunscheduledlist, [],
   {
   ClearTranscriptImage( Transcript1 );
   HideImage( Transcript1 );
   } );

      /*************************************
      ****  FUNCTION: Startscheduler
      *************************************/
MakeFunction( Startscheduler, [],
   {
   SetWindowTitle( SESSION, "Packaging Scheduler Prototype  :  Oklahoma State University May
1991" );
   SetWindowBackground( SESSION, 0, 0, 100 );
   ClearList( Global:HiddenImages );
   ClearList( Global:ShownImages );
   AppendToList( Global:HiddenImages, Button6, Button7, Button1,
               Button2, Button5, Button12, Button4 );
   AppendToList( Global:ShownImages, Button4, Button8, Button9,
               Text1, Button13, Button14, Button15, Button34,
               Button17, Button20, Button18, Button31 );
   DisplayImages(  );
   Formschpartsheader(  );
   Formunschpartsheader(  );
   MaximizeWindow( SESSION );
   } );

     /************************************
     ****  FUNCTION: Exitscheduler
     ************************************/
MakeFunction( Exitscheduler, [],
   {
   SetPostMenuPosition( 375, 75 );
   Global:Choice = PostMenu( "Confirm Exit?", YES, NO );
   If ( Global:Choice #= YES )
    Then {
        OpenWriteFile( pkgsch.out );
        SaveTranscriptImage( Transcript1 );
        CloseWriteFile(  );
        OpenWriteFile( pkgunsch.out );
        SaveTranscriptImage( Transcript24 );
        CloseWriteFile(  );
        ClearList( Global:HiddenImages );
        ClearList( Global:ShownImages );
        AppendToList( Global:HiddenImages, Button1, Button2,
                    Button4, Button5, Button9, Button10, Button12,
                    Text1, Text15, Button13, Button14, Button15,
                    Button16, Button17, Button18, Button19,
                    Button11, Button20, Text20, Text2, Button22,
                    Button23, Button24, Button28, Button29,
                    Text6, Transcript1, Transcript24, Button34,
                    Button31, Transcript31, Transcript3, Transcript4,
                    Transcript27 );
        AppendToList( Global:ShownImages, Bitmap1, Button7,
                    Button8 );
        DisplayImages(  );
        Exit(  );
        };
   } );

     /************************************
     ****  FUNCTION: DisplayImages
     ************************************/
MakeFunction( DisplayImages, [],
   {
   EnumList( Global:ShownImages, Images, ShowImage( Images ) );
   EnumList( Global:HiddenImages, Images, HideImage( Images ) );
   } );

      /************************************
       ****  FUNCTION: Restartscheduler
       ************************************/
MakeFunction( Restartscheduler, [],
   {
   SetPostMenuPosition( 75, 75 );
   Global:Temp1 = PostMenu( "Restart Scheduler?", YES, NO );
   If ( Global:Temp1 #= YES )
     Then {
         SetWindowBackground( SESSION, 0, 0, 100 );
         ClearList( Global:HiddenImages );
         ClearList( Global:ShownImages );
         AppendToList( Global:HiddenImages, Button1, Button2,
                    Button4, Button5, Button9, Button10, Button12,
                    Text1, Text15, Button13, Button14, Button15,
                    Button16, Button17, Button18, Button19,
                    Button11, Button20, Text20, Text2, Button22,
                    Button23, Button24, Button28, Button29,
                    Text6, Transcript1, Transcript24, Button34,
                    Button31, Transcript31, Transcript3, Transcript4,
                    Transcript27 );
         AppendToList( Global:ShownImages, Button7, Button8 );
         DisplayImages(  );
         MaximizeWindow( SESSION );
         };
   } );

      /************************************
       ****  FUNCTION: Deleteinstances
       ************************************/
MakeFunction( Deleteinstances, [],
   {
   For x [1 305 ]
      {
      Global:Temp1 = P # x;
      DeleteInstance( Global:Temp1 );
      };
   } );

      /************************************
       ****  FUNCTION: Readproductsinfo
       ************************************/
MakeFunction( Readproductsinfo, [],
   {
   ClearList( Global:HiddenImages );
   ClearList( Global:ShownImages );
   AppendToList( Global:HiddenImages, Button4, Text1, Button13,
             Button14, Button15, Button16, Button17, Button18 );
   AppendToList( Global:ShownImages, Button1, Button2, Button5,
            Button12, Text15, Button19 );
   DisplayImages( );
   } );

     /************************************
      **** FUNCTION: Displayproductinformation
      ************************************/
MakeFunction( Displayproductinformation, [],
   {
   Hidebeforedisplayprodinfo( );
   ShowImage( Text4 );
   Global:Temp1 = AA_AF299;
   Global:Temp1 = PostMenu( "Choose a part number", Global:Parts );
   HideImage( Text4 );
   Global:Fromwhere = SCR1;
   displayprodinfochoices( );
   } );

     /************************************
      **** FUNCTION: Displaypinfo
      ************************************/
MakeFunction( Displaypinfo, [],
   {
   Hidebeforeprodinfo( );
   PostInputForm( "Product Information", Global:Temp1:Partnumber,
            "Part Number ", Global:Temp1:Familycode, "Family Code",
            Global:Temp1:Abccode, "ABC Code", Global:Temp1:SafetyStock,
            "Safety Stock ", Global:Temp1:ProductionPlannedForDayT_1,
            "Yesterday's Package Size", Global:Temp1:PlannedUsageForDayT_1,
            "Yesterday's Shipments", Global:Temp1:ProjectedInventoryForDayT_1,
            "Projected Inventory", Global:Temp1:ProductionSize,
            "Proposed Package Size ", Global:Temp1:ScheduledYesorNo,
            "Scheduled Yes / No" );
   Showafterprodinfo( );
   } );

     /************************************
      **** FUNCTION: gettime
      ************************************/
MakeFunction( gettime, [],
   {
   Execute( winclock.exe );
   } );

     /************************************
      **** FUNCTION: showcalculator
      ************************************/
MakeFunction( showcalculator, [],
   {
   Execute( calc.exe );
   PostMessage( "Press ALT-TAB to use the Calculator" );
   } );

     /************************************
```

```
    **** FUNCTION: Browsefile
    *************************************/
MakeFunction( Browsefile, [],
   {
   PostInputForm( "Enter Filename ", Global:Browsefilename, " " );
   IconifyWindow( SESSION );
   Execute( notepad.exe, Global:Browsefilename );
   } );

   /*************************************
    **** FUNCTION: displaypartnumbers
    *************************************/
MakeFunction( displaypartnumbers, [],
   {
   ClearTranscriptImage( Transcript1 );
   For x [1 305 ]
      {
      Global:Temp1 = P # x;
      DisplayText( Transcript1, Global:Temp1:Partnumber );
      };
   } );

   /*************************************
    **** FUNCTION: Readprodinputfile
    *************************************/
MakeFunction( Readprodinputfile, [],
   {
   Global:Temp1 = inputf.c;
   PostBusy( ON, "Please Wait....Transferring Inventory " );
   OpenReadFile( Global:Temp1 );
   ReadWord( 50 );
   ReadWord( 50 );
   For x [1 305 ]
      {
      Global:Temp1 = ReadWord(  );
      Global:Temp1:ProductionPlannedForDayT_1 = ReadWord(  );
      Global:Temp1:PlannedUsageForDayT_1 = ReadWord(  );
      };
   CloseReadFile(  );
   PostBusy( OFF );
   } );

   /*************************************
    **** FUNCTION: Transferdata
    *************************************/
MakeFunction( Transferdata, [],
   {
   HideImage( Text20 );
   Readprodinputfile(  );
   Readprodsizes(  );
   Readinvt_1(  );
   Readresourcesinfo(  );
   Readbackorderquantity(  );
   Readschinfo(  );
   PostBusy( ON, "Please Wait...Processing Data" );
   Readdistanceinfo( );
   Formpriorities( );
   Updateshortages( );
   PostBusy( ON );
   Formscheduledlist( );
   Formunscheduledlist( );
   PostBusy( OFF );
   HideImage( Text20 );
   HideImage( Button11 );
   Showscr2( );
   } );

     /************************************
     ****  FUNCTION: Displaydetailedinformation
     ************************************/
MakeFunction( Displaydetailedinformation, [],
   {
   ClearList( Global:ShownImages );
   ClearList( Global:HiddenImages );
   AppendToList( Global:HiddenImages, Button1, Button2, Button5,
            Button12, Text15, Button9, Button8, Bitmap1 );
   DisplayImages( );
   Global:Temp1 = "AA AF299";
   Global:Temp1 = PostMenu( "Choose a Partnumber", "AA AF299", "AA AF300",
                    "AA AF602", "AA AF971", AF1149, AF3192,
                    AF3465, AF3471 );
   Displaydinfo( );
   } );

     /************************************
     ****  FUNCTION: Displaydinfo
     ************************************/
MakeFunction( Displaydinfo, [],
   {
   Hidebeforeprodinfo( );
   Global:Cnum = Global:Temp1:Cartonnumber;
   Global:Snum = Global:Temp1:ShippingContainerNumber;
   Global:Lnum = Global:Temp1:Labelnumber;
   If Not( Instance?( C # Global:Cnum ) )
     Then ResetValue( Global:Cnum );
   If Not( Instance?( S # Global:Snum ) )
     Then ResetValue( Global:Snum );
   If Not( Instance?( L # Global:Lnum ) )
     Then ResetValue( Global:Lnum );
   PostInputForm( "Product  BOM Information", Global:Temp1:Partnumber,
            "Part Number", Global:Temp1:AverageMonthlyForecast,
            "Average Mothly Forecast ", Global:Cnum, "Carton Number ",
            Global:Temp1:ContainerUsage, "Container Usage ",
            Global:Lnum, "Label Number ", Global:Temp1:PackingType,
            "Packing Type ", Global:Snum, "Shipping Container Number ",
            Global:Temp1:StandardCost, "Standard Cost " );
   Showafterprodinfo( );
   } );
```

```
     /************************************
      ****  FUNCTION: Onlinehelp1
      ************************************/
MakeFunction( Onlinehelp1, [],
   {
   Execute( "WINHELP.EXE WINHELP.HLP" );
   PostMessage( "Press ALT-TAB to use the Help System" );
   } );

     /************************************
      ****  FUNCTION: Closeonlinehelp1
      ************************************/
MakeFunction( Closeonlinehelp1, [],
   {
   ClearTranscriptImage( Transcript1 );
   HideImage( Transcript1 );
   } );

     /************************************
      ****  FUNCTION: Onlinehelp2
      ************************************/
MakeFunction( Onlinehelp2, [],
   {
   ClearTranscriptImage( Transcript1 );
   ShowImage( Transcript1 );
   DisplayFile( Transcript1, online2.c );
   } );

     /************************************
      ****  FUNCTION: Closeonlinehelp2
      ************************************/
MakeFunction( Closeonlinehelp2, [],
   {
   ClearTranscriptImage( Transcript1 );
   HideImage( Transcript1 );
   } );

     /************************************
      ****  FUNCTION: formpartslist
      ************************************/
MakeFunction( formpartslist, [],
   {
   ClearList( Global:Parts );
   For x [1 300 ]
      {
      Global:Temp1 = P # x;
      AppendToList( Global:Parts, Global:Temp1:Partnumber );
      };
   Global:Temp2 = PostMenu( "Choose a Part number", For x [1 10
                                             ]
                                          GetNthElem( Global:Parts,
                                                x ) );
   } );
```

```
    /***********************************
     ****  FUNCTION: findpart
     ***********************************/
MakeFunction( findpart, [],
   {
   For x [1 305 ]
      {
      Global:Temp3 = P # x;
      };
   } );

    /***********************************
     ****  FUNCTION: graphgenerate
     ***********************************/
MakeFunction( graphgenerate, [],
   {
   HideImage( Button19 );
   PostBusy( ON, "Please Wait...Generating Graph" );
   Global:dx = 0.25;
   Global:Num = 1;
   ClearList( Global:Xaxis1 );
   ClearList( Global:Xaxis2 );
   ClearList( Global:Xaxis3 );
   AppendToList( Global:Xaxis1, 0 );
   AppendToList( Global:Xaxis2, 0 );
   AppendToList( Global:Xaxis3, 0 );
   ClearList( Global:Bs1 );
   ClearList( Global:Bs2 );
   ClearList( Global:Bs3 );
   ClearList( Global:Bs4 );
   AppendToList( Global:Bs1, 0 );
   AppendToList( Global:Bs2, 0 );
   AppendToList( Global:Bs3, 0 );
   AppendToList( Global:Bs4, 0 );
   drawgraph(  );
   ClearTranscriptImage( Transcript2 );
   DisplayText( Transcript2, FormatValue( "%-13s  %13s", "Part Number :  ",
                                    Global:Temp1 ) );
   PostBusy( OFF );
   } );

    /***********************************
     ****  FUNCTION: drawgraph
     ***********************************/
MakeFunction( drawgraph, [],
   {
   AppendToList( Global:Xaxis1, Global:Num - Global:dx, Global:Num
                                          - Global:dx,
            Global:Num + Global:dx, Global:Num + Global:dx );
   AppendToList( Global:Xaxis2, 1.75, 1.75, 2.25, 2.25 );
   AppendToList( Global:Xaxis3, 2.75, 2.75, 3.25, 3.25 );
   AppendToList( Global:Bs1, 0, Global:Temp1:SafetyStock, Global:Temp1:SafetyStock,
            0 );
   AppendToList( Global:Bs2, 0, Global:Temp1:ProjectedInventoryForDayT_1,
            Global:Temp1:ProjectedInventoryForDayT_1, 0 );
   AppendToList( Global:Bs3, 0, Global:Temp1:ProductionSize, Global:Temp1:ProductionSize,
            0 );
   } );

      /************************************
      **** FUNCTION: maincontinue
      ************************************/
MakeFunction( maincontinue, [],
   {
   HideImage( Bitmap2 );
   ShowImage( Bitmap1 );
   ShowImage( Button7 );
   ShowImage( Button8 );
   } );

      /************************************
      **** FUNCTION: Readinvt_1
      ************************************/
MakeFunction( Readinvt_1, [],
   {
   Global:Temp1 = it_1.c;
   PostBusy( ON, "Please Wait...Transferring Inventory" );
   OpenReadFile( Global:Temp1 );
   ReadWord( 17 );
   ReadWord( 17 );
   For x [1 305 ]
      {
      Global:Temp1 = ReadWord( );
      Global:Temp1:ProjectedInventoryForDayT_1 = ReadWord( );
      };
   CloseReadFile( );
   PostBusy( OFF );
   } );

      /************************************
      **** FUNCTION: Readresourcesinfo
      ************************************/
MakeFunction( Readresourcesinfo, [],
   {
   Readcartonsinfo( );
   Readshipctninfo( );
   Readlabelsinfo( );
   Readfamilyinfo( );
   } );

      /************************************
      **** FUNCTION: displayprodinfochoices
      ************************************/
MakeFunction( displayprodinfochoices, [],
   {
   graphgenerate( );
   ClearList( Global:ShownImages );
   ClearList( Global:HiddenImages );
   AppendToList( Global:ShownImages, LinePlot1, Transcript2, Text3,
             Button19, Button21, Button35, Button3 );
   AppendToList( Global:HiddenImages, Button19 );
   DisplayImages(  );
   } );

     /************************************
     ****  FUNCTION: Continue
     *************************************/
MakeFunction( Continue, [],
   {
   PostBusy( ON, "Please Wait...Resetting Screens" );
   ClearList( Global:HiddenImages );
   ClearList( Global:ShownImages );
   AppendToList( Global:HiddenImages, LinePlot1, Button3, Text3,
             Button21, Transcript2, Button35 );
   If ( Global:Fromwhere #= SCR1 )
     Then {
          AppendToList( Global:ShownImages, Bitmap1, Button9,
                    Button8, Button17, Button20, Text6, Button12,
                    Button1, Button18, Button13, Button14,
                    Button15, Button31, Button34 );
          If ( Global:Seenschedule #= YES )
            Then {
                 ShowImage( Button28 );
                 ShowImage( Button22 );
                 ShowImage( Button5 );
                 };
          };
   DisplayImages(  );
   If ( Global:Fromwhere #= SCR2 )
     Then Showforceschedulechangescreen(  );
   PostBusy( OFF );
   } );

     /************************************
     ****  FUNCTION: Runcinitializer
     *************************************/
MakeFunction( Runcinitializer, [],
   {
   Execute( prod.exe );
   HideImage( Button16 );
   ShowImage( Button11 );
   } );

     /************************************
     ****  FUNCTION: Readschinfo
     *************************************/
MakeFunction( Readschinfo, [],
   {
   Global:Temp1 = schinfo.c;
   PostBusy( ON, "Please Wait...Transferring Schedule Info" );
   OpenReadFile( Global:Temp1 );
   ReadWord( 42 );
   ReadWord( 42 );
   For x [1 305 ]
      {
      Global:Temp1 = ReadWord( );
      Global:Temp2 = ReadWord( );
      If ( Global:Temp2 #= 1 )
        Then ( Global:Temp1:ScheduledYesorNo = YES )
        Else Global:Temp1:ScheduledYesorNo = NO;
      Global:Temp3 = ReadWord( );
      If ( Global:Temp2 #= 0 )
        Then Global:Temp1:ReasonForNotBeingScheduled = Global:Temp3;
      };
   CloseReadFile( );
   PostBusy( OFF );
   } );

      /************************************
      **** FUNCTION: Readcartonsinfo
      ************************************/
MakeFunction( Readcartonsinfo, [],
   {
   Global:Temp1 = cout.c;
   PostBusy( ON, "Please Wait...Reading Cartons Info" );
   OpenReadFile( Global:Temp1 );
   ReadWord( 85 );
   ReadWord( 85 );
   For x [1 182 ]
      {
      Global:Temp1 = ReadWord( );
      Global:Temp2 = C # Global:Temp1;
      If Not( Instance?( Global:Temp2 ) )
        Then MakeInstance( Global:Temp2, Cartons );
      Global:Temp2:Type = Global:Temp1;
      Global:Temp2:PlannedReceiptsForDayT_1 = ReadWord( );
      Global:Temp2:PlannedUsageForDayT_1 = ReadWord( );
      Global:Temp2:ProjectedInventoryForDayT_1 = ReadWord( );
      Global:Temp2:NumberOfPartsUsingThisType = ReadWord( );
      Global:Temp2:ProjectedUsage = ReadWord( );
      Global:Temp2:Available = ReadWord( );
      };
   CloseReadFile( );
   PostBusy( OFF );
   } );

      /************************************
      **** FUNCTION: Readshipctninfo
      ************************************/
MakeFunction( Readshipctninfo, [],
   {
   Global:Temp1 = sout.c;
   PostBusy( ON, "Please Wait....Transferring ShipCtn Info" );
   OpenReadFile( Global:Temp1 );
   ReadWord( 86 );
   ReadWord( 86 );
```

```
    For x [1 52 ]
       {
       Global:Temp1 = ReadWord( );
       Global:Temp2 = S # Global:Temp1;
       If Not( Instance?( Global:Temp2 ) )
         Then MakeInstance( Global:Temp2, ShippingContainers );
       Global:Temp2:Type = Global:Temp1;
       Global:Temp2:PlannedReceiptsForDayT_1 = ReadWord( );
       Global:Temp2:PlannedUsageForDayT_1 = ReadWord( );
       Global:Temp2:ProjectedInventoryForDayT_1 = ReadWord( );
       Global:Temp2:NumberOfPartsUsingThisType = ReadWord( );
       Global:Temp2:ProjectedUsage = ReadWord( );
       Global:Temp2:Available = ReadWord( );
       };
    CloseReadFile( );
    PostBusy( OFF );
    } );

       /***********************************
       ****  FUNCTION: Readlabelsinfo
       ***********************************/
 MakeFunction( Readlabelsinfo, [],
    {
    Global:Temp1 = lout.c;
    PostBusy( ON, "Please Wait...Transferring Labels Info" );
    OpenReadFile( Global:Temp1 );
    ReadWord( 84 );
    ReadWord( 84 );
    For x [1 20 ]
       {
       Global:Temp1 = ReadWord( );
       Global:Temp2 = L # Global:Temp1;
       If Not( Instance?( Global:Temp2 ) )
         Then MakeInstance( Global:Temp2, Labels );
       Global:Temp2:Type = Global:Temp1;
       Global:Temp2:PlannedReceiptsForDayT_1 = ReadWord( );
       Global:Temp2:PlannedUsageForDayT_1 = ReadWord( );
       Global:Temp2:ProjectedInventoryForDayT_1 = ReadWord( );
       Global:Temp2:NumberOfPartsUsingThisType = ReadWord( );
       Global:Temp2:ProjectedUsage = ReadWord( );
       Global:Temp2:Available = ReadWord( );
       };
    CloseReadFile( );
    PostBusy( OFF );
    } );

       /***********************************
       ****  FUNCTION: delpinst
       ***********************************/
 MakeFunction( delpinst, [],
    {
    ClearList( Global:Parts );
    GetInstanceList( Products, Global:Parts );
    Global:Num = LengthList( Global:Parts );
    PostMessage( Global:Num );
    For x [1 Global:Num ]
        {
        Global:Temp2 = GetNthElem( Global:Parts, x );
        DeleteInstance( Global:Temp2 );
        };
    } );

        /***********************************
        ****  FUNCTION: Formreslists
        ***********************************/
MakeFunction( Formreslists, [],
    {
    ClearList( Global:Cartonnums );
    ClearList( Global:Shipctnnums );
    ClearList( Global:Labelnums );
    GetInstanceList( Cartons, Global:Cartonnums );
    GetInstanceList( ShippingContainers, Global:Shipctnnums );
    GetInstanceList( Labels, Global:Labelnums );
    GetInstanceList( Family, Global:Familynums );
    } );

        /***********************************
        ****  FUNCTION: Clearreslists
        ***********************************/
MakeFunction( Clearreslists, [],
    {
    Global:Num = LengthList( Global:Cartonnums );
    For x [1 Global:Num ]
        {
        Global:Temp1 = GetNthElem( Global:Cartonnums, x );
        ClearList( Global:Temp1:PartnumbersUsingThisType );
        };
    Global:Num = LengthList( Global:Shipctnnums );
    For x [1 Global:Num ]
        {
        Global:Temp1 = GetNthElem( Global:Shipctnnums, x );
        ClearList( Global:Temp1:PartnumbersUsingThisType );
        };
    Global:Num = LengthList( Global:Labelnums );
    For x [1 Global:Num ]
        {
        Global:Temp1 = GetNthElem( Global:Labelnums, x );
        ClearList( Global:Temp1:PartnumbersUsingThisType );
        };
    Global:Num = LengthList( Global:Familynums );
    For x [1 Global:Num ]
        {
        Global:Temp1 = GetNthElem( Global:Familynums, x );
        ClearList( Global:Temp1:PartnumbersUsingThisType );
        };
    } );

        /***********************************
```

```
      **** FUNCTION: Aggregatereslists
      ************************************/
MakeFunction( Aggregatereslists, [],
   {
   Global:Num = LengthList( Global:Parts );
   For x [1 Global:Num ]
      {
      Global:Temp1 = GetNthElem( Global:Parts, x );
      Global:Temp2 = Global:Temp1:Cartonnumber;
      Global:Temp3 = C # Global:Temp2;
      If Instance?( Global:Temp3 )
        Then AppendToList( Global:Temp3:PartnumbersUsingThisType,
                           Global:Temp1 );
      Global:Temp2 = Global:Temp1:ShippingContainerNumber;
      Global:Temp3 = S # Global:Temp2;
      If Instance?( Global:Temp3 )
        Then AppendToList( Global:Temp3:PartnumbersUsingThisType,
                           Global:Temp1 );
      Global:Temp2 = Global:Temp1:Labelnumber;
      Global:Temp3 = L # Global:Temp2;
      If Instance?( Global:Temp3 )
        Then AppendToList( Global:Temp3:PartnumbersUsingThisType,
                           Global:Temp1 );
      Global:Temp2 = Global:Temp1:Familycode;
      Global:Temp3 = F # Global:Temp2;
      If Instance?( Global:Temp3 )
        Then AppendToList( Global:Temp3:PartnumbersUsingThisType,
                           Global:Temp1 );
      };
   } );

     /************************************
      **** FUNCTION: Updatebominformation
      ************************************/
MakeFunction( Updatebominformation, [],
   {
   PostMessage( "BOM Info Already Updated, This operation will take approximately 5 Minutes to
complete !" );
   SetPostMenuPosition( 200, 180 );
   Global:Choice = PostMenu( "Continue ?", YES, NO );
   If ( Global:Choice #= YES )
     Then Readmainframedata( );
   } );

     /************************************
      **** FUNCTION: Hidebeforeprodinfo
      ************************************/
MakeFunction( Hidebeforeprodinfo, [],
   {
   HideImage( Text3 );
   HideImage( LinePlot1 );
   HideImage( Transcript2 );
   HideImage( Button21 );
   HideImage( Button3 );
   HideImage( Button35 );
   } );

     /************************************
     **** FUNCTION: Showafterprodinfo
     ************************************/
MakeFunction( Showafterprodinfo, [],
   {
   ShowImage( Text3 );
   ShowImage( LinePlot1 );
   ShowImage( Transcript2 );
   ShowImage( Button21 );
   ShowImage( Button35 );
   ShowImage( Button3 );
   } );

     /************************************
     **** FUNCTION: Formscheduledlist
     ************************************/
MakeFunction( Formscheduledlist, [],
   {
   ClearTranscriptImage( Transcript1 );
   ClearList( Global:Scheduledparts );
   Global:Num = LengthList( Global:Parts );
   Global:Seqnum = 1;
   For x [1 Global:Num ]
      {
      Global:Temp1 = GetNthElem( Global:Parts, x );
      If Not( KnownValue?( Global:Temp1:Comment ) )
        Then ( Global:Temp2 = "          " )
        Else Global:Temp2 = Global:Temp1:Comment;
      If ( Global:Temp1:ScheduledYesorNo #= YES And Global:Temp1:ProductionSize
           > 0 )
        Then {
            AppendToList( Global:Scheduledparts, Global:Temp1 );
            DisplayText( Transcript1, FormatValue( " %3d\t%-14s%14s\t%-6d\t   %10s
\t%-14s%-6d%-14s\n",
                                        Global:Seqnum,
                                        Global:Temp1:Partnumber,
                                        Global:Temp1:Familycode,
                                        Global:Temp1:ProductionSize,
                                        Global:Temp1:PackingType,
                                        Global:Temp1:BackOrder?,
                                        Global:Temp1:BackOrderQuantity,
                                        Global:Temp2 ) );
            Global:Temp1:Seqnumber = Global:Seqnum;
            Global:Seqnum = Global:Seqnum + 1;
            };
      };
   Reformscheduledlist( );
   } );

     /************************************
     **** FUNCTION: Displayschedule
```

```
    ************************************/
MakeFunction( Displayschedule, [],
   {
   HideImage( Text6 );
   Global:Seenschedule = YES;
   SetPostMenuPosition( 230, 180 );
   Global:Temp1 = PostMenu( "Choose an Option", "Show Scheduled Parts",
                     "Show Unscheduled Parts" );
   If ( Global:Temp1 #= "Show Scheduled Parts" )
     Then {
         ShowImage( Transcript3 );
         ShowImage( Transcript1 );
         ShowImage( Button22 );
         ShowImage( Button5 );
         ShowImage( Button28 );
         }
     Else {
         ShowImage( Transcript4 );
         ShowImage( Transcript24 );
         ShowImage( Button22 );
         ShowImage( Button5 );
         ShowImage( Button28 );
         };
   } );

   /************************************
    **** FUNCTION: Hidebeforeshowschedulemenu
    ************************************/
MakeFunction( Hidebeforeshowschedulemenu, [],
   {
   HideImage( Button19 );
   HideImage( Text15 );
   HideImage( Button12 );
   HideImage( Button1 );
   HideImage( Button5 );
   HideImage( Button28 );
   HideImage( Button2 );
   } );

   /************************************
    **** FUNCTION: Viewhideschedule
    ************************************/
MakeFunction( Viewhideschedule, [],
   {
   SetPostMenuPosition( 200, 180 );
   Global:Temp1 = PostMenu( "Choose an Option", "Show Scheduled Parts",
                     "Show Unscheduled Parts", "Hide Schedule Window" );
   If ( Global:Temp1 #= "Show Scheduled Parts" )
     Then {
         HideImage( Transcript4 );
         HideImage( Transcript24 );
         ShowImage( Transcript3 );
         ShowImage( Transcript1 );
         };
    If ( Global:Temp1 #= "Show Unscheduled Parts" )
     Then {
          HideImage( Transcript3 );
          HideImage( Transcript1 );
          ShowImage( Transcript4 );
          ShowImage( Transcript24 );
          };
    If ( Global:Temp1 #= "Hide Schedule Window" )
     Then {
          HideImage( Transcript3 );
          HideImage( Transcript1 );
          HideImage( Transcript4 );
          HideImage( Transcript24 );
          };
    } );

      /************************************
       ****  FUNCTION: Viewhideunscheduledparts
       ************************************/
MakeFunction( Viewhideunscheduledparts, [],
    {
    If ( Global:Schshowing #= YES )
     Then HideImage( Transcript1 );
    If ( Global:Unschshowing #= YES )
     Then {
          HideImage( Transcript24 );
          Global:Unschshowing = NO;
          }
     Else {
          If ( Global:Unschshowing #= NO )
            Then {
                 ShowImage( Transcript24 );
                 Global:Unschshowing = YES;
                 };
          };
    } );

      /************************************
       ****  FUNCTION: Exittoscr2
       ************************************/
MakeFunction( Exittoscr2, [],
    {
    PostBusy( ON, "Please Wait...Resetting Screens" );
    ClearList( Global:HiddenImages );
    ClearList( Global:ShownImages );
    AppendToList( Global:HiddenImages, Text2, Button22, Button23,
                Button24 );
    AppendToList( Global:ShownImages, Bitmap1, Button9, Button8,
                 Text1, Button17, Button20, Text6, Button12, Button1,
                 Button5, Button28, Button18, Button13, Button14,
                 Button15 );
    DisplayImages( );
    PostBusy( OFF );
    } );
```

```
      /**********************************
       ****  FUNCTION: Formunscheduledlist
       **********************************/
MakeFunction( Formunscheduledlist, [],
   {
   ClearTranscriptImage( Transcript24 );
   ClearList( Global:Unscheduledparts );
   Global:Num = LengthList( Global:Parts );
   For x [1 Global:Num ]
      {
      Global:Temp1 = GetNthElem( Global:Parts, x );
      If Not( KnownValue?( Global:Temp1:Comment ) )
        Then ( Global:Temp2 = "              " )
        Else Global:Temp2 = Global:Temp1:Comment;
      If ( ( Global:Temp1:ScheduledYesorNo #= NO ) And ( Global:Temp1:ProductionSize
                                                   >
                                                   0 ) )
        Then {
            DisplayText( Transcript24, FormatValue( "%-18s%14s\t%-6d\t\t%-6d\t   %-6d    \t
%-6d\t %6d\t%14s\n",
                                         Global:Temp1:Partnumber,
                                         Global:Temp1:Familycode,
                                         Global:Temp1:ProjectedInventoryForDayT_1,
                                         Global:Temp1:SafetyStock,
                                         Global:Temp1:PlannedUsageForDayT_1,
                                         Global:Temp1:ProductionSize,
                                         Global:Temp1:BackOrderQuantity,
                                         Global:Temp1:Shortage ) );
            AppendToList( Global:Unscheduledparts, Global:Temp1 );
            };
      };
   ClearTranscriptImage( Transcript31 );
   ClearList( Global:Unscheduledparts );
   Global:Num = LengthList( Global:Parts );
   For x [1 Global:Num ]
      {
      Global:Temp1 = GetNthElem( Global:Parts, x );
      If Not( KnownValue?( Global:Temp1:Comment ) )
        Then ( Global:Temp2 = "              " )
        Else Global:Temp2 = Global:Temp1:Comment;
      If ( ( Global:Temp1:ScheduledYesorNo #= NO ) And ( Global:Temp1:ProductionSize
                                                   >
                                                   0 ) )
        Then {
            DisplayText( Transcript31, FormatValue( "%-18s%14s\t%-6d\t\t%-6d\t   %-6d    \t
%-6d\t %6d\t%14s\n",
                                         Global:Temp1:Partnumber,
                                         Global:Temp1:Familycode,
                                         Global:Temp1:ProjectedInventoryForDayT_1,
                                         Global:Temp1:SafetyStock,
                                         Global:Temp1:PlannedUsageForDayT_1,
                                         Global:Temp1:ProductionSize,
                                         Global:Temp1:BackOrderQuantity,
                                         Global:Temp1:Shortage ) );
            AppendToList( Global:Unscheduledparts, Global:Temp1 );
            };
      };
   } );

      /************************************
      ****  FUNCTION: Showschedulemenu
      ************************************/
MakeFunction( Showschedulemenu, [],
   {
   ShowImage( Text2 );
   ShowImage( Button22 );
   ShowImage( Button23 );
   ShowImage( Button24 );
   } );

      /************************************
      ****  FUNCTION: Forceschedulechange
      ************************************/
MakeFunction( Forceschedulechange, [],
   {
   Hidebeforedisplayprodinfo(  );
   Global:Changemade = NO;
   Showforceschedulechangescreen(  );
   } );

      /************************************
      ****  FUNCTION: Ok1
      ************************************/
MakeFunction( Ok1, [],
   {
   HideImage( Transcript27 );
   HideImage( Text32 );
   HideImage( Button32 );
   SetPostMenuPosition( 300, 300 );
   Global:Temp1 = PostMenu( "Choose an Unscheduled part number",
                        Global:Unscheduledparts );
   PostBusy( ON, "PleaseWait...Processing Query" );
   Global:Cnum = C # Global:Temp1:Cartonnumber;
   Global:Snum = S # Global:Temp1:ShippingContainerNumber;
   Global:Lnum = L # Global:Temp1:Labelnumber;
   Global:Fnum = F # Global:Temp1:Familycode;
   ClearTranscriptImage( Transcript27 );
   If ( Instance?( Global:Cnum ) And ( Global:Cnum:Available
                                 < Global:Temp1:ProductionSize ) )
     Then {
        DisplayText( Transcript27, FormatValue( "%-60s\n\n", Global:Temp1
                                                #
                                                "  is out of Cartons, Scheduled Parts using this
type :" ) );
        Global:Num = LengthList( Global:Cnum:PartnumbersUsingThisType );
        For x [1 Global:Num ]
           {
           Global:Temp3 = GetNthElem( Global:Cnum:PartnumbersUsingThisType,
                              x );
          If ( ( Global:Temp3:ScheduledYesorNo #= YES ) And
               ( Global:Temp3:ProductionSize > 0 ) )
           Then {
               DisplayText( Transcript27, FormatValue( "%-15s%-6d\n",
                                                       Global:Temp3,
                                                       Global:Temp3:ProductionSize ) );
               };
          };
       };
   Checkforsameshipctns( );
   Checkforsamelabels( );
   Checkforsamefamily( );
   PostBusy( OFF );
   ShowImage( Transcript27 );
   } );

     /************************************
      **** FUNCTION: Checkforsameshipctns
      ************************************/
MakeFunction( Checkforsameshipctns, [],
   {
   If Instance?( Global:Snum )
    Then {
       If ( ( Global:Temp1:ContainerUsage > 0 ) And ( Global:Temp1:ContainerUsage
                                               <
                                               1 ) )
         Then {
             Global:Num2 = Floor( 1 / Global:Temp1:ContainerUsage );
             Global:Num3 = Ceil( Global:Temp1:ProductionSize
                                 / Global:Num2 );
             If ( Global:Snum:Available < Global:Num3 )
              Then {
                 DisplayText( Transcript27, FormatValue( "\n\n%-60s\n\n",
                                                         Global:Temp1
                                                          #
                                                          " is out of Ship.Ctns, Scheduled Parts Using this
type :" ) );
                 Global:Num = LengthList( Global:Snum:PartnumbersUsingThisType );
                 For x [1 Global:Num ]
                    {
                    Global:Temp3 = GetNthElem( Global:Snum:PartnumbersUsingThisType,
                                               x );
                    If ( ( Global:Temp3:ScheduledYesorNo
                           #= Yes ) And ( Global:Temp3:ProductionSize
                                              > 0 ) )
                      Then DisplayText( Transcript27,
                                FormatValue( "%-15s%-6d\n",
                                             Global:Temp3,
                                             Global:Temp3:ProductionSize ) );
                    };
                 };
             };
       };
   } );

      /*************************************
       ****  FUNCTION: Checkforsamelabels
       *************************************/
MakeFunction( Checkforsamelabels, [],
   {
   If ( Instance?( Global:Lnum ) And ( Global:Lnum:Available
                                  < Global:Temp1:ProductionSize ) )
     Then {
         DisplayText( Transcript27, FormatValue( "\n\n%-60s\n\n",
                                          Global:Temp1
                                            # "  is out of Labels, Scheduled parts using this type :" ) );
         Global:Num = LengthList( Global:Lnum:PartnumbersUsingThisType );
         For x [1 Global:Num ]
            {
            Global:Temp3 = GetNthElem( Global:Lnum:PartnumbersUsingThisType,
                                   x );
            If ( ( Global:Temp3:ScheduledYesorNo #= YES ) And
                  ( Global:Temp3:ProductionSize > 0 ) )
              Then {
                  DisplayText( Transcript27, FormatValue( "%-15s%-6d\n",
                                                    Global:Temp3,
                                                    Global:Temp3:ProductionSize ) );
                  };
            };
         };
   } );

      /*************************************
       ****  FUNCTION: Closequerytranscript
       *************************************/
MakeFunction( Closequerytranscript, [],
   {
   HideImage( Transcript27 );
   } );

      /*************************************
       ****  FUNCTION: Exitfromforcesc
       *************************************/
MakeFunction( Exitfromforcesc, [],
   {
   HideImage( Transcript1 );
   HideImage( Transcript31 );
   HideImage( Transcript27 );
   HideImage( Transcript3 );
   HideImage( Transcript5 );
   If ( Global:Changemade #= YES )
     Then Reschedule(  );
   PostBusy( ON, "Please Wait...Resetting Screens" );
   ClearList( Global:HiddenImages );
   ClearList( Global:ShownImages );
   AppendToList( Global:HiddenImages, Transcript1, Transcript31,
              Transcript27, Transcript3, Transcript5, Button30,
               Button25, Button26, Button27 );
   AppendToList( Global:ShownImages, Bitmap1, Button8, Button9,
               Button17, Button20, Text6, Button12, Button1,
               Button5, Button28, Button18, Button13, Button14,
               Button15, Button22, Button31, Button34 );
   DisplayImages( );
   PostBusy( OFF );
   } );

     /************************************
      **** FUNCTION: Changeschedule
      *************************************/
MakeFunction( Changeschedule, [],
   {
   SetPostMenuPosition( 350, 75 );
   Global:Temp1 = PostMenu( "Choose Type of Change", "Change Scheduled Quantity",
                        "Increase Priority", "Increase Inventory",
                        "Replace Scheduled Part", CANCEL );
   If ( Global:Temp1 #= "Change Scheduled Quantity" )
     Then Changescheduledquantity( );
   If ( Global:Temp1 #= "Increase Priority" )
     Then Increasepriority( );
   If ( Global:Temp1 #= "Increase Inventory" )
     Then {
          Forceintoschedule( );
          };
   If ( Global:Temp1 #= "Replace Scheduled Part" )
     Then {
          Replacescheduledpart( );
          };
   } );

     /************************************
      **** FUNCTION: Hidebeforeforceintoschedule
      *************************************/
MakeFunction( Hidebeforeforceintoschedule, [],
   {
   HideImage( Transcript3 );
   HideImage( Transcript1 );
   HideImage( Transcript27 );
   HideImage( Transcript5 );
   HideImage( Transcript31 );
   HideImage( Button30 );
   HideImage( Button25 );
   HideImage( Button26 );
   HideImage( Button27 );
   } );

     /************************************
      **** FUNCTION: Showforceschedulechangescreen
      *************************************/
MakeFunction( Showforceschedulechangescreen, [],
   {
   ShowImage( Transcript3 );
   ShowImage( Transcript1 );
   ShowImage( Transcript5 );
   ShowImage( Transcript31 );
   ShowImage( Button30 );
   ShowImage( Button25 );
   ShowImage( Button26 );
   ShowImage( Button27 );
   } );

    /************************************
    ****  FUNCTION: Ok2
    ************************************/
MakeFunction( Ok2, [],
   {
   HideImage( Text33 );
   HideImage( Button33 );
   Global:Changemade = YES;
   Global:Temp1 = PostMenu( "Choose an Unscheduled Part Number",
                     Global:Unscheduledparts );
   Global:Cnum = C # Global:Temp1:Cartonnumber;
   Global:Snum = S # Global:Temp1:ShippingContainerNumber;
   Global:Lnum = L # Global:Temp1:Labelnumber;
   Global:Fnum = F # Global:Temp1:Familycode;
   If ( Instance?( Global:Cnum ) And ( Global:Cnum:Available
                                   < Global:Temp1:ProductionSize ) )
     Then {
         Global:Choice = PostMenu( "Part was out of " # Global:Cnum,
                             "Increase Inventory", CANCEL );
         If ( Global:Choice #= "Increase Inventory" )
           Then {
               ResetValue( Global:Num );
               PostInputForm( "Increase Inventory of " # Global:Cnum,
                         Global:Num, "Enter Amount to Increase by :" );
               Global:Cnum:ProjectedInventoryForDayT_1 += Global:Num;
               };
         };
   Increaseinvofshipctns( );
   Increaseinvoflabels( );
   Increaseinvoffamily( );
   } );

    /************************************
    ****  FUNCTION: Increaseinvandreschedule
    ************************************/
MakeFunction( Increaseinvandreschedule, [],
   {
   PostInputForm( "Increase Inventory of " # Global:Temp3, Global:Num,
              "Enter Amount to Increase by :" );
   Global:Temp5:Available = Global:Temp5:Available + Global:Num;
   Global:Temp5:ProjectedInventoryForDayT_1 = Global:Temp5:ProjectedInventoryForDayT_1
     + Global:Num;
   PostBusy( ON, "Inventory Increased..." );
   Wait( 1 );
   PostBusy( OFF );
   } );

      /************************************
      **** FUNCTION: Reformscheduledlist
      ************************************/
MakeFunction( Reformscheduledlist, [],
   {
   ClearTranscriptImage( Transcript1 );
   Formsequence( );
   Global:Num = LengthList( Global:Scheduledparts );
   For x [1 Global:Num ]
      {
      Global:Temp1 = GetNthElem( Global:Scheduledparts, x );
      If Not( KnownValue?( Global:Temp1:Comment ) )
        Then ( Global:Temp2 = "              " )
        Else Global:Temp2 = Global:Temp1:Comment;
      DisplayText( Transcript1, FormatValue( " %3d\t%-14s%14s\t%-6d\t   %10s
\t%-14s%-6d%-14s\n",
                                    x, Global:Temp1,
                                    Global:Temp1:Familycode,
                                    Global:Temp1:ProductionSize,
                                    Global:Temp1:PackingType,
                                    Global:Temp1:BackOrder?,
                                    Global:Temp1:BackOrderQuantity,
                                    Global:Temp2 ) );
      };
   } );

      /************************************
      **** FUNCTION: Removefromunschlist
      ************************************/
MakeFunction( Removefromunschlist, [],
   {
   RemoveFromList( Global:Unscheduledparts, Global:Temp1 );
   } );

      /************************************
      **** FUNCTION: Reformunscheduledlist
      ************************************/
MakeFunction( Reformunscheduledlist, [],
   {
   ClearTranscriptImage( Transcript24 );
   Global:Num = LengthList( Global:Unscheduledparts );
   For x [1 Global:Num ]
      {
      Global:Temp1 = GetNthElem( Global:Unscheduledparts, x );
      If Not( KnownValue?( Global:Temp1:Comment ) )
        Then ( Global:Temp2 = "              " )
        Else Global:Temp2 = Global:Temp1:Comment;
      DisplayText( Transcript24, FormatValue( "%-18s%14s\t%-6d\t\t%-6d\t    %-6d\t  %-6d\t
%6d\t%14s\n",
                                    Global:Temp1:Partnumber,
                                    Global:Temp1:Familycode,
                                    Global:Temp1:ProjectedInventoryForDayT_1,
                                        Global:Temp1:SafetyStock,
                                        Global:Temp1:PlannedUsageForDayT_1,
                                        Global:Temp1:ProductionSize,
                                        Global:Temp1:BackOrderQuantity,
                                        Global:Temp1:Shortage ) );
      };
   ClearTranscriptImage( Transcript31 );
   Global:Num = LengthList( Global:Unscheduledparts );
   For x [1 Global:Num ]
      {
      Global:Temp1 = GetNthElem( Global:Unscheduledparts, x );
      If Not( KnownValue?( Global:Temp1:Comment ) )
        Then ( Global:Temp2 = "            " )
        Else Global:Temp2 = Global:Temp1:Comment;
      DisplayText( Transcript31, FormatValue( "%-18s%14s\t%-6d\t\t%-6d\t    %-6d\t  %-6d\t
%6d\t%14s\n",
                                        Global:Temp1:Partnumber,
                                        Global:Temp1:Familycode,
                                        Global:Temp1:ProjectedInventoryForDayT_1,
                                        Global:Temp1:SafetyStock,
                                        Global:Temp1:PlannedUsageForDayT_1,
                                        Global:Temp1:ProductionSize,
                                        Global:Temp1:BackOrderQuantity,
                                        Global:Temp1:Shortage ) );
      };
   } );

     /************************************
      ****  FUNCTION: Acceptschedule
      ************************************/
MakeFunction( Acceptschedule, [],
   {
   SetPostMenuPosition( 440, 380 );
   Global:Temp1 = PostMenu( "Accept Schedule", YES, NO );
   If ( Global:Temp1 #= YES )
     Then {
          OpenWriteFile( pkgsch.out );
          SaveTranscriptImage( Transcript1 );
          CloseWriteFile( );
          OpenWriteFile( pkgunsch.out );
          SaveTranscriptImage( Transcript24 );
          CloseWriteFile( );
          Writeprodsizes( );
          Aggregatefamilyunits( );
          Writeresourcesusage( );
          ShowImage( Button29 );
          };
   } );

     /************************************
      ****  FUNCTION: Printschedule
      ************************************/
MakeFunction( Printschedule, [],
   {
    SetPostMenuPosition( 440, 395 );
    Global:Printer = PostMenu( "Choose Printer Type", PostScript,
                         "Non - PostScript", CANCEL );
    If Not( Global:Printer #= CANCEL )
      Then {
         If ( Global:Printer #= "Non - PostScript" )
           Then {
              Execute( "print.com pkgsch.out " );
              Execute( "print.com pkgunsch.out " );
              }
           Else {
              Execute( "psp -O lpt1 pkgsch.out" );
              Execute( "psp -O lpt1 pkgunsch.out" );
              };
         };
    } );

      /************************************
       **** FUNCTION: Runscheduler
       ************************************/
MakeFunction( Runscheduler, [],
    {
    Global:Seenschedule = NO;
    SetPostMenuPosition( 230, 180 );
    Global:Temp1 = PostMenu( "Run Scheduler?", YES, NO );
    If ( Global:Temp1 #= YES )
      Then {
         RunC( );
         ShowImage( Text20 );
         ShowImage( Button11 );
         HideImage( Text1 );
         };
    If ( Global:Temp1 #= NO )
      Then {
         ClearTranscriptImage( Transcript1 );
         ClearTranscriptImage( Transcript24 );
         DisplayFile( Transcript1, pkgsch.out );
         DisplayFile( Transcript24, pkgunsch.out );
         DisplayFile( Transcript31, pkgunsch.out );
         Showscr2( );
         HideImage( Text1 );
         };
    } );

      /************************************
       **** FUNCTION: Showscr2
       ************************************/
MakeFunction( Showscr2, [],
    {
    ShowImage( Text6 );
    ShowImage( Button12 );
    ShowImage( Button1 );
    } );
```

```
      /*************************************
       ****  FUNCTION: Hidebeforedisplayprodinfo
       *************************************/
MakeFunction( Hidebeforedisplayprodinfo, [],
   {
   ClearList( Global:HiddenImages );
   ClearList( Global:ShownImages );
   AppendToList( Global:HiddenImages, Bitmap1, Button9, Button8,
              Text1, Button17, Button20, Text6, Button12, Button1,
              Button5, Button28, Button29, Button18, Button13,
              Button14, Button15, Button31, Button22, Button34,
              Transcript1, Transcript24, Transcript3, Transcript4 );
   DisplayImages(  );
   } );

      /*************************************
       ****  FUNCTION: Displayschpartnuminfo
       *************************************/
MakeFunction( Displayschpartnuminfo, [],
   {
   Global:Fromwhere = SCR2;
   Global:Temp1 = PostMenu( "Choose a Scheduled Part Number", Global:Scheduledparts );
   Hidebeforeforceintoschedule(  );
   displayprodinfochoices(  );
   } );

      /*************************************
       ****  FUNCTION: Displayunschpartnuminfo
       *************************************/
MakeFunction( Displayunschpartnuminfo, [],
   {
   Global:Fromwhere = SCR2;
   Global:Temp1 = PostMenu( "Choose a Unscheduled Part Number",
                     Global:Unscheduledparts );
   Hidebeforeforceintoschedule(  );
   displayprodinfochoices(  );
   } );

      /*************************************
       ****  FUNCTION: Replacescheduledpart
       *************************************/
MakeFunction( Replacescheduledpart, [],
   {
   Global:Changemade = YES;
   Global:Temp1 = PostMenu( "Choose a Scheduled Part Number to be Replaced",
                     Global:Scheduledparts );
   Global:Rpartnum = PostMenu( "Choose a Unscheduled Part to Replace "
                        # Global:Temp1, Global:Unscheduledparts );
   SetPostMenuPosition( 200, 180 );
   Global:Temp4 = PostMenu( "Replace " # Global:Temp1 # " with "
                      # Global:Rpartnum, "Continue with Replacement",
                     CANCEL );
   If ( Global:Temp4 #= "Continue with Replacement" )
     Then {
        Continuewithreplacement( );
        };
   } );

     /***********************************
      ****  FUNCTION: Continuewithreplacement
      ************************************/
MakeFunction( Continuewithreplacement, [],
   {
   PostBusy( ON, "Please Wait...Processing Replacement" );
   Global:Num1 = GetElemPos( Global:Scheduledparts, Global:Temp1 );
   Global:Num2 = GetElemPos( Global:Unscheduledparts, Global:Rpartnum );
   RemoveFromList( Global:Scheduledparts, Global:Temp1 );
   RemoveFromList( Global:Unscheduledparts, Global:Rpartnum );
   InsertNthElem( Global:Scheduledparts, Global:Num1, Global:Rpartnum );
   InsertNthElem( Global:Unscheduledparts, Global:Num2, Global:Temp1 );
   Wait( 1 );
   PostBusy( OFF );
   } );

     /***********************************
      ****  FUNCTION: Checkandproceedwithreplacement
      ************************************/
MakeFunction( Checkandproceedwithreplacement, [],
   {
   If ( ( Global:Cavailable #= 1 ) And ( Global:Savailable #=
                                   1 ) And ( Global:Lavailable
                                            #= 1 ) )
     Then {
        Global:Cnum:Available = Global:Cnum:Available - Global:Temp1:ProductionSize;
        Global:Snum:Available = Global:Snum:Available - Global:Temp1:ProductionSize;
        Global:Lnum:Available = Global:Lnum:Available - Global:Temp1:ProductionSize;
        Global:Temp1:ScheduledYesorNo = NO;
        Global:Temp1:ReasonForNotBeingScheduled =
Global:Rpartnum:ReasonForNotBeingScheduled;
        Global:Rpartnum:ScheduledYesorNo = YES;
        Global:Rpartnum:ReasonForNotBeingScheduled = " ";
        RemoveFromList( Global:Scheduledparts, Global:Temp1 );
        InsertNthElem( Global:Unscheduledparts, Global:Num,
                  Global:Temp1 );
        RemoveFromList( Global:Unscheduledparts, Global:Rpartnum );
        InsertNthElem( Global:Scheduledparts, Global:Num2, Global:Rpartnum );
        Reformscheduledlist( );
        Reformunscheduledlist( );
        PostBusy( OFF );
        Beep( );
        PostBusy( ON, "Replacement Successful" );
        Wait( 4 );
        Showforceschedulechangescreen( );
        PostBusy( OFF );
        }
     Else Resetlists( );
   } );
```

```
    /***********************************
     ****  FUNCTION: Resetlists
     ************************************/
MakeFunction( Resetlists, [],
  {
  PostBusy( OFF );
  Beep(  );
  PostBusy( ON, "Resources Released Not Enough " );
  Wait( 5 );
  PostBusy( OFF );
  Showforceschedulechangescreen(  );
  } );

    /***********************************
     ****  FUNCTION: Displaypartinfo
     ************************************/
MakeFunction( Displaypartinfo, [],
  {
  SetPostMenuPosition( 25, 75 );
  Global:Choice = PostMenu( "Choose an Option", "Scheduled Product Info",
                      "Unscheduled Product Info", CANCEL );
  If ( Global:Choice #= "Scheduled Product Info" )
    Then Displayschpartnuminfo(  );
  If ( Global:Choice #= "Unscheduled Product Info" )
    Then Displayunschpartnuminfo(  );
  } );

    /***********************************
     ****  FUNCTION: Dosshell
     ************************************/
MakeFunction( Dosshell, [],
  {
  Execute( dos.pif );
  } );

    /***********************************
     ****  FUNCTION: Queryunscheduledparts
     ************************************/
MakeFunction( Queryunscheduledparts, [],
  {
  ShowImage( Text32 );
  ShowImage( Button32 );
  } );

    /***********************************
     ****  FUNCTION: Forceintoschedule
     ************************************/
MakeFunction( Forceintoschedule, [],
  {
  ShowImage( Text33 );
  ShowImage( Button33 );
  } );

    /***********************************
```

```
     **** FUNCTION: RunParadox
     ************************************/
MakeFunction( RunParadox, [],
   {
   Execute( pdox.pif );
   } );

    /************************************
     **** FUNCTION: RunC
     ************************************/
MakeFunction( RunC, [],
   Execute( Prod.exe ) );

    /************************************
     **** FUNCTION: Readfamilyinfo
     ************************************/
MakeFunction( Readfamilyinfo, [],
   {
   Global:Temp1 = fout.c;
   PostBusy( ON, "Please Wait...Transferring Family Info" );
   OpenReadFile( Global:Temp1 );
   ReadWord( 84 );
   ReadWord( 84 );
   For x [1 13 ]
      {
      Global:Temp1 = ReadWord( );
      Global:Temp2 = F # Global:Temp1;
      If Not( Instance?( Global:Temp2 ) )
        Then MakeInstance( Global:Temp2, Family );
      Global:Temp2:Type = Global:Temp1;
      Global:Temp2:PlannedReceiptsForDayT_1 = ReadWord( );
      Global:Temp2:PlannedUsageForDayT_1 = ReadWord( );
      Global:Temp2:ProjectedInventoryForDayT_1 = ReadWord( );
      Global:Temp2:NumberOfPartsUsingThisType = ReadWord( );
      Global:Temp2:ProjectedUsage = ReadWord( );
      Global:Temp2:Available = ReadWord( );
      };
   CloseReadFile( );
   PostBusy( OFF );
   } );

    /************************************
     **** FUNCTION: Writeprodsizes
     ************************************/
MakeFunction( Writeprodsizes, [],
   {
   Global:Temp1 = newprod.c;
   OpenWriteFile( Global:Temp1 );
   PostBusy( ON, "Please Wait...Writing Package Sizes" );
   ForAll [ x|Products ]
         {
         Global:Temp2 = x:Partnumber # "      " # x:ProductionSize;
         WriteLine( Global:Temp2 );
         };
    CloseWriteFile( );
    PostBusy( OFF );
    } );


      /************************************
       ****  FUNCTION: Readbackorderquantity
       ************************************/
MakeFunction( Readbackorderquantity, [],
    {
    PostBusy( ON, "Please Wait...Transferring BackOrder Info" );
    Global:Temp1 = backorder.c;
    OpenReadFile( Global:Temp1 );
    ReadWord( 24 );
    ReadWord( 24 );
    For x [1 305 ]
       {
       Global:Temp2 = ReadWord( );
       Global:Temp2:BackOrderQuantity = ReadWord( );
       If ( Global:Temp2:BackOrderQuantity > 0 )
         Then ( Global:Temp2:BackOrder? = YES )
         Else Global:Temp2:BackOrder? = NO;
       };
    CloseReadFile( );
    PostBusy( OFF );
    } );

      /************************************
       ****  FUNCTION: Formschpartsheader
       ************************************/
MakeFunction( Formschpartsheader, [],
    {
    ClearTranscriptImage( Transcript3 );
    DisplayText( Transcript3, FormatValue( "\t\tPackaging Schedule\n" ),
              FormatValue( "Date : %8s\n", Date( ) ), FormatValue( "%-8s %-14s   %-14s   %-10s
%-10s   %-6s   %-10s   %-14s",
               "Seq#", "Part Number", "Equiv. Type", Sch.Qty, Pkg.Type,
               " BO ", "BO. Qty", Comment ) );
    } );

      /************************************
       ****  FUNCTION: Formunschpartsheader
       ************************************/
MakeFunction( Formunschpartsheader, [],
    {
    ClearTranscriptImage( Transcript4 );
    ClearTranscriptImage( Transcript5 );
    DisplayText( Transcript4, FormatValue( "\t\t Unscheduled Parts\n" ),
              FormatValue( "Date : %8s \n", Date( ) ), FormatValue( "%-14s   %-14s   %-14s
%-14s%-14s %-10s   %-10s   %-10s",
               "Part Number", "Equiv. Type", "On Hand", Safety.Stk,
               Cust.Orders, "Sch. Qty.", "BO. Qty", Reason ) );
    DisplayText( Transcript5, FormatValue( "\t\t Unscheduled Parts\n" ),
              FormatValue( "Date : %8s \n", Date( ) ), FormatValue( "%-14s   %-14s   %-14s
%-14s%-14s %-10s   %-10s   %-10s",
                "Part Number", "Equiv. Type", "On Hand", Safety.Stk,
                Cust.Orders, "Sch. Qty.", "BO. Qty", Reason ) );
   } );


      /***********************************
       ****  FUNCTION: Checkforsamefamily
       ***********************************/
MakeFunction( Checkforsamefamily, [],
   {
   If ( Instance?( Global:Fnum ) And ( Global:Fnum:Available
                               < Global:Temp1:ProductionSize ) )
     Then {
         DisplayText( Transcript27, FormatValue( "\n\n%-60s\n\n",
                                         "Part : " # Global:Temp1
                                           # "  is out of Equiv. Type Family Units, Scheduled Parts
Using this type :" ) );
         Global:Num = LengthList( Global:Fnum:PartnumbersUsingThisType );
         For x [1 Global:Num ]
             {
             Global:Temp3 = GetNthElem( Global:Fnum:PartnumbersUsingThisType,
                                  x );
             If ( ( Global:Temp3:ScheduledYesorNo #= YES ) And
                    ( Global:Temp3:ProductionSize > 0 ) )
               Then {
                    DisplayText( Transcript27, FormatValue( "%-15s%-6d\n",
                                                  Global:Temp3,
                                                  Global:Temp3:ProductionSize ) );
                    };
             };
         };
   } );

      /***********************************
       ****  FUNCTION: Changescheduledquantity
       ***********************************/
MakeFunction( Changescheduledquantity, [],
   {
   SetPostMenuPosition( 200, 180 );
   Global:Changemade = YES;
   Global:Temp1 = PostMenu( "Change Scheduled Quantity Of :", "Scheduled Part",
                     "Unscheduled Part", CANCEL );
   If ( Global:Temp1 #= "Scheduled Part" )
     Then Changeqtyofscheduledpart(  );
   If ( Global:Temp1 #= "Unscheduled Part" )
     Then Changeqtyofunscheduledpart(  );
   } );

      /***********************************
       ****  FUNCTION: Changeqtyofscheduledpart
       ***********************************/
MakeFunction( Changeqtyofscheduledpart, [],
   {
    Global:Temp1 = PostMenu( "Choose a Scheduled Part", Global:Scheduledparts );
    Global:Num = Global:Temp1:ProductionSize;
    Global:Num2 = Global:Num;
    PostInputForm( "Change Scheduled Quantity Of : " # Global:Temp1,
                Global:Num2, "Enter New Quantity :" );
    If ( Global:Num2 > Global:Num )
      Then {
          Beep( );
          PostBusy( ON, "Quantity entered was more than Sch. Qty..." );
          Wait( 1 );
          PostBusy( OFF );
          Global:Choice = PostMenu( "Increase Scheduled Quantity?",
                                  YES, NO );
          If ( Global:Choice #= YES )
            Then {
                Global:Temp1:ProductionSize = Global:Num2;
                PostBusy( ON, "Scheduled Quantity Changed..." );
                Wait( 1 );
                PostBusy( OFF );
                };
          };
    If ( Global:Num2 < Global:Num )
      Then {
          Global:Temp1:ProductionSize = Global:Num2;
          PostBusy( ON, "Scheduled Quantity Changed..." );
          Wait( 1 );
          PostBusy( OFF );
          };
    } );

      /************************************
       ****  FUNCTION: Proceedwithdecqtyofschpart
       ************************************/
MakeFunction( Proceedwithdecqtyofschpart, [],
    {
    If KnownValue?( Global:Temp1:ContainerUsage )
      Then {
          If ( ( Global:Temp1:ContainerUsage > 0 ) And ( Global:Temp1:ContainerUsage
                                              <
                                              1 ) )
            Then {
                Global:Num4 = Floor( 1 / Global:Temp1:ContainerUsage );
                If Instance?( Global:Snum )
                  Then {
                      Global:Num5 = Floor( Global:Num3 / Global:Num4 );
                      Global:Snum:Available = Global:Snum:Available
                        + Global:Num5;
                      };
                };
          };
    Global:Temp1:ProductionSize = Global:Num2;
    If Instance?( Global:Cnum )
      Then Global:Cnum:Available = Global:Cnum:Available + Global:Num3;
    If Instance?( Global:Lnum )
    Then Global:Lnum:Available = Global:Lnum:Available + Global:Num3;
   If Instance?( Global:Fnum )
    Then Global:Fnum:Available = Global:Fnum:Available + Global:Num3;
   Beep( );
   PostBusy( ON, "Scheduled Quantity Decreased..." );
   Wait( 2 );
   PostBusy( OFF );
   } );

     /************************************
      **** FUNCTION: Changeqtyofunscheduledpart
      ************************************/
MakeFunction( Changeqtyofunscheduledpart, [],
   {
   Global:Temp1 = PostMenu( "Select an Unscheduled Part", Global:Unscheduledparts );
   Global:Num = Global:Temp1:ProductionSize;
   Global:Num2 = Global:Num;
   PostInputForm( "Decrease Scheduled Qty. Of : " # Global:Temp1,
              Global:Num2, "Enter New Quantity :" );
   If ( Global:Num2 > Global:Num )
    Then {
        Beep( );
        PostBusy( ON, "Quantity entered was more than Sch. Qty..." );
        Wait( 2 );
        PostBusy( OFF );
        };
   If ( Global:Num2 < Global:Num )
    Then {
        Global:Temp1:ProductionSize = Global:Num2;
        Beep( );
        PostBusy( ON, "Scheduled Quantity Decreased..." );
        Wait( 2 );
        PostBusy( OFF );
        };
   } );

     /************************************
      **** FUNCTION: Readdistanceinfo
      ************************************/
MakeFunction( Readdistanceinfo, [],
   {
   Global:Temp1 = distance.c;
   OpenReadFile( Global:Temp1 );
   ReadWord( 30 );
   ReadWord( 30 );
   For x [1 305 ]
      {
      Global:Temp2 = ReadWord( );
      Global:Temp2:ModifiedDistanceawayfromSS = ReadWord( );
      If ( x = = 1 )
       Then Global:Maxdistance = Global:Temp2:ModifiedDistanceawayfromSS;
      };
   CloseReadFile( );
   } );
```

```
      /************************************
       **** FUNCTION: Increasepriority
       ************************************/
MakeFunction( Increasepriority, [],
   {
   Global:Temp1 = PostMenu( "Increase Priority : Choose an Unscheduled Part",
                        Global:Unscheduledparts );
   ResetValue( Global:Num );
   Global:Num = Global:Temp1:Priority;
   PostInputForm( "Increase Priority of  " # Global:Temp1, Global:Num,
              "Enter Priority between 1 to 100 :" );
   If ( Global:Num > Global:Temp1:Priority )
     Then {
         Global:Changemade = YES;
         Global:Temp1:Priority = Global:Num;
         PostBusy( ON, "Please Wait...Processing Priority Change" );
         Insertincorrectplace(  );
         PostBusy( OFF );
         }
     Else {
         PostBusy( ON, "Priority Not Greater than Original Priority..." );
         Wait( 1 );
         PostBusy( OFF );
         };
   } );

      /************************************
       **** FUNCTION: Formpriorities
       ************************************/
MakeFunction( Formpriorities, [],
   {
   ForAll [ x|Products ]
         {
         x:Priority = Ceil( ( x:ModifiedDistanceawayfromSS /
                           Global:Maxdistance ) * 100 );
         };
   } );

      /************************************
       **** FUNCTION: Reassignavailable
       ************************************/
MakeFunction( Reassignavailable, [],
   {
   ForAll [ x|Cartons ]
         {
         x:Available = x:ProjectedInventoryForDayT_1;
         };
   ForAll [ x|ShippingContainers ]
         {
         x:Available = x:ProjectedInventoryForDayT_1;
         };
   ForAll [ x|Labels ]
         {
         x:Available = x:ProjectedInventoryForDayT_1;
```

```
        };
    ForAll [ x|Family ]
        {
        x:Available = x:ProjectedInventoryForDayT_1;
        };
    } );

      /************************************
      **** FUNCTION: Sortparts
      ************************************/
MakeFunction( Sortparts, [],
    {
    PostBusy( ON, "Please Wait...Sorting Parts" );
    ClearList( Global:Sortedparts );
    For Num [100 1 -1]
      {
      ForAll [ x|Products ]
            {
            If ( x:Priority == Num )
              Then AppendToList( Global:Sortedparts, x );
            };
      };
    PostBusy( OFF );
    } );

      /************************************
      **** FUNCTION: Getmaxmdindex
      ************************************/
MakeFunction( Getmaxmdindex, [],
    {
    Global:Temp1 = GetNthElem( Global:Parts, Global:Num );
    Global:Maxd = Global:Temp1:Priority;
    } );

      /************************************
      **** FUNCTION: Initialize
      ************************************/
MakeFunction( Initialize, [],
    {
    Global:Cavailable = 0;
    Global:Savailable = 0;
    Global:Lavailable = 0;
    Global:Favailable = 0;
    } );

      /************************************
      **** FUNCTION: Updateshortages
      ************************************/
MakeFunction( Updateshortages, [],
    {
    PostBusy( OFF );
    PostBusy( ON, "Please Wait...Updating Shortages" );
    ForAll [ x|Products ]
        {
          If ( x:ScheduledYesorNo #= NO )
            Then {
                Initialize( );
                Global:Cnum = C # x:Cartonnumber;
                Global:Snum = S # x:ShippingContainerNumber;
                Global:Lnum = L # x:Labelnumber;
                Global:Fnum = F # x:Familycode;
                If Instance?( Global:Cnum )
                  Then {
                      If ( Global:Cnum:Available >= x:ProductionSize )
                        Then Global:Cavailable = 1;
                      }
                  Else Global:Cavailable = N;
                If Instance?( Global:Snum )
                  Then {
                      If ( ( x:ContainerUsage > 0 ) And ( x:ContainerUsage
                                                          <
                                                          1 ) )
                        Then {
                            Global:Num3 = Floor( 1 / x:ContainerUsage );
                            Global:Num4 = Ceil( x:ProductionSize
                                                / Global:Num3 );
                            If ( Global:Snum:Available >=
                                   Global:Num4 )
                              Then Global:Savailable = 1;
                            };
                      }
                  Else Global:Savailable = N;
                If Instance?( Global:Lnum )
                  Then {
                      If ( Global:Lnum:Available >= x:ProductionSize )
                        Then Global:Lavailable = 1;
                      }
                  Else Global:Lavailable = N;
                If Instance?( Global:Fnum )
                  Then {
                      If ( Global:Fnum:Available >= x:ProductionSize )
                        Then Global:Favailable = 1;
                      }
                  Else Global:Favailable = N;
                Formshortage( );
                x:Shortage = Global:Temp4;
                };
          };
   PostBusy( OFF );
   } );

      /************************************
       ****  FUNCTION: Formshortage
       ************************************/
MakeFunction( Formshortage, [],
   {
   ResetValue( Global:Temp4 );
   If ( Global:Cavailable #= 0 )
```

```
     Then Global:Temp4 = Global:Temp4 # C;
   If ( Global:Savailable #= 0 )
     Then Global:Temp4 = Global:Temp4 # S;
   If ( Global:Lavailable #= 0 )
     Then Global:Temp4 = Global:Temp4 # L;
   If ( Global:Favailable #= 0 )
     Then Global:Temp4 = Global:Temp4 # F;
   } );

     /************************************
      **** FUNCTION: Insertincorrectplace
      ************************************/
MakeFunction( Insertincorrectplace, [],
   {
   RemoveFromList( Global:Unscheduledparts, Global:Temp1 );
   Global:Num1 = LengthList( Global:Scheduledparts );
   Global:Choice = NOTFOUND;
   Global:Num = 1;
   While  (( Not( Global:Choice #= FOUND ) And ( Global:Num <=
                                                Global:Num1 ) ))
        {
        Global:Temp2 = GetNthElem( Global:Scheduledparts, Global:Num );
        If ( Global:Temp2:Priority <= Global:Temp1:Priority )
          Then {
              InsertNthElem( Global:Scheduledparts, Global:Num,
                          Global:Temp1 );
              Global:Choice = FOUND;
              }
          Else Global:Num = Global:Num + 1;
        };
   } );

     /************************************
      **** FUNCTION: Increaseinvofshipctns
      ************************************/
MakeFunction( Increaseinvofshipctns, [],
   {
   If Instance?( Global:Snum )
     Then {
        If ( ( Global:Temp1:ContainerUsage > 0 ) And ( Global:Temp1:ContainerUsage
                                              <
                                              1 ) )
          Then {
              Global:Num2 = Floor( 1 / Global:Temp1:ContainerUsage );
              Global:Num3 = Ceil( Global:Temp1:ProductionSize
                            / Global:Num2 );
              If ( Global:Snum:Available < Global:Num3 )
                Then {
                    Global:Choice = PostMenu( "Part was out of "
                                         # Global:Snum,
                                       "Increase Inventory",
                                       CANCEL );
                    If ( Global:Choice #= "Increase Inventory" )
                      Then {
                        ResetValue( Global:Num );
                        PostInputForm( "Increase Inventory of "
                               # Global:Snum,
                              Global:Num, "Enter Amount to Increase by :" );
                        Global:Snum:ProjectedInventoryForDayT_1
                          += Global:Num;
                        };
                    };
                };
            };
    } );

        /**************************************
         ****  FUNCTION: Increaseinvoflabels
         **************************************/
MakeFunction( Increaseinvoflabels, [],
    {
    If ( Instance?( Global:Lnum ) And ( Global:Lnum:Available
                                    < Global:Temp1:ProductionSize ) )
     Then {
        Global:Choice = PostMenu( "Part was out of " # Global:Lnum,
                            "Increase Inventory", CANCEL );
        If ( Global:Choice #= "Increase Inventory" )
          Then {
             ResetValue( Global:Num );
             PostInputForm( "Increase Inventory of " # Global:Lnum,
                      Global:Num, "Enter Amount to Increase by :" );
             Global:Lnum:ProjectedInventoryForDayT_1 += Global:Num;
             };
        };
    } );

        /**************************************
         ****  FUNCTION: Increaseinvoffamily
         **************************************/
MakeFunction( Increaseinvoffamily, [],
    {
    If ( Instance?( Global:Fnum ) And ( Global:Fnum:Available
                                    < Global:Temp1:ProductionSize ) )
     Then {
        Global:Choice = PostMenu( "Part was out of " # Global:Fnum,
                            "Increase Inventory", CANCEL );
        If ( Global:Choice #= "Increase Inventory" )
          Then {
             ResetValue( Global:Num );
             PostInputForm( "Increase Inventory of " # Global:Fnum,
                      Global:Num, "Enter Amount to Increase by :" );
             Global:Fnum:ProjectedInventoryForDayT_1 += Global:Num;
             };
        };
    } );

        /**************************************
         ****  FUNCTION: Trycartons
     ************************************/
MakeFunction( Trycartons, [],
   {
   If Not( Instance?( Global:Cnum ) )
     Then ( Global:Cavailable = 1 )
     Else {
         If ( Global:Cnum:Available >= Global:Temp1:ProductionSize )
           Then Global:Cavailable = 1;
         };
   } );

    /************************************
     **** FUNCTION: Reschedule
     ************************************/
MakeFunction( Reschedule, [],
   {
   PostBusy( ON, "PleaseWait...Processing Schedule Change" );
   Reassignavailable( );
   Global:Num = LengthList( Global:Scheduledparts );
   ClearList( Global:Tomove );
   For x [1 Global:Num ]
      {
      Global:Temp1 = GetNthElem( Global:Scheduledparts, x );
      Initialize( );
      Initresnames( );
      Trycartons( );
      Tryshipctns( );
      Trylabels( );
      Tryfamily( );
      If ( ( Global:Cavailable #= 1 ) And ( Global:Savailable
                                       #= 1 ) And ( Global:Lavailable
                                                    #=
                                                    1 )
           And ( Global:Favailable #= 1 ) )
        Then {
           Global:Temp1:ScheduledYesorNo = YES;
           Updateres( );
           }
        Else {
           Global:Temp1:ScheduledYesorNo = NO;
           AppendToList( Global:Tomove, Global:Temp1 );
           };
      };
   CheckTomovelist( );
   Tryunscheduledlist( );
   Updateshortages( );
   Reformscheduledlist( );
   Reformunscheduledlist( );
   PostBusy( OFF );
   } );

    /************************************
     **** FUNCTION: Initresnames
     ************************************/
```

```
MakeFunction( Initresnames, [],
   {
   Global:Cnum = C # Global:Temp1:Cartonnumber;
   Global:Snum = S # Global:Temp1:ShippingContainerNumber;
   Global:Lnum = L # Global:Temp1:Labelnumber;
   Global:Fnum = F # Global:Temp1:Familycode;
   } );

     /***********************************
      **** FUNCTION: Updateres
      ***********************************/
MakeFunction( Updateres, [],
   {
   If Instance?( Global:Cnum )
     Then Global:Cnum:Available = Global:Cnum:Available - Global:Temp1:ProductionSize;
   If Instance?( Global:Snum )
     Then {
         If ( ( Global:Temp1:ContainerUsage > 0 ) And ( Global:Temp1:ContainerUsage
                                                   <
                                                   1 ) )
           Then {
               Global:Num2 = Floor( 1 / Global:Temp1:ContainerUsage );
               Global:Num3 = Ceil( Global:Temp1:ProductionSize
                                 / Global:Num2 );
               Global:Snum:Available = Global:Snum:Available
                 - Global:Num3;
               };
         };
   If Instance?( Global:Lnum )
     Then Global:Lnum:Available = Global:Lnum:Available - Global:Temp1:ProductionSize;
   If Instance?( Global:Fnum )
     Then Global:Fnum:Available = Global:Fnum:Available - Global:Temp1:ProductionSize;
   } );

     /***********************************
      **** FUNCTION: Trylabels
      ***********************************/
MakeFunction( Trylabels, [],
   {
   If Not( Instance?( Global:Lnum ) )
     Then ( Global:Lavailable = 1 )
     Else {
         If ( Global:Lnum:Available >= Global:Temp1:ProductionSize )
           Then Global:Lavailable = 1;
         };
   } );

     /***********************************
      **** FUNCTION: Tryfamily
      ***********************************/
MakeFunction( Tryfamily, [],
   {
   If Not( Instance?( Global:Fnum ) )
     Then ( Global:Favailable = 1 )
```

```
      Else {
          If ( Global:Fnum:Available >= Global:Temp1:ProductionSize )
            Then Global:Favailable = 1;
          };
    } );

      /***********************************
       ****  FUNCTION: Tryshipctns
       ***********************************/
MakeFunction( Tryshipctns, [],
    {
    If Not( Instance?( Global:Snum ) )
      Then Global:Savailable = 1;
    If Instance?( Global:Snum )
      Then {
          If ( ( Global:Temp1:ContainerUsage > 0 ) And ( Global:Temp1:ContainerUsage
                                                     <
                                                     1 ) )
            Then {
                Global:Num2 = Floor( 1 / Global:Temp1:ContainerUsage );
                Global:Num3 = Ceil( Global:Temp1:ProductionSize
                                    / Global:Num2 );
                If ( Global:Snum:Available >= Global:Num3 )
                  Then Global:Savailable = 1;
                };
          };
    } );

      /***********************************
       ****  FUNCTION: Tryunscheduledlist
       ***********************************/
MakeFunction( Tryunscheduledlist, [],
    {
    ClearList( Global:Tomove );
    Global:Num = LengthList( Global:Unscheduledparts );
    For x [1 Global:Num ]
        {
        Global:Temp1 = GetNthElem( Global:Unscheduledparts, x );
        Initialize( );
        Initresnames( );
        Trycartons( );
        Tryshipctns( );
        Trylabels( );
        Tryfamily( );
        If ( ( Global:Cavailable #= 1 ) And ( Global:Savailable
                                        #= 1 ) And ( Global:Lavailable
                                                     #=
                                                     1 )
               And ( Global:Favailable #= 1 ) )
          Then {
               Global:Temp1:ScheduledYesorNo = YES;
               Updateres( );
               AppendToList( Global:Tomove, Global:Temp1 );
               }
          Else {
              Global:Temp1:ScheduledYesorNo = NO;
              };
        };
   Checkmovelist( );
   } );

     /***********************************
     ****  FUNCTION: CheckTomovelist
     ***********************************/
MakeFunction( CheckTomovelist, [],
   {
   Global:Num = LengthList( Global:Tomove );
   If ( Global:Num != 0 )
     Then {
         For x [1 Global:Num ]
             {
             Global:Temp1 = GetNthElem( Global:Tomove, x );
             RemoveFromList( Global:Scheduledparts, Global:Temp1 );
             InsertNthElem( Global:Unscheduledparts, 1, Global:Temp1 );
             };
         };
   } );

     /***********************************
     ****  FUNCTION: Checkmovelist
     ***********************************/
MakeFunction( Checkmovelist, [],
   {
   Global:Num = LengthList( Global:Tomove );
   If ( Global:Num != 0 )
     Then {
         For x [1 Global:Num ]
             {
             Global:Temp1 = GetNthElem( Global:Tomove, x );
             RemoveFromList( Global:Unscheduledparts, Global:Temp1 );
             AppendToList( Global:Scheduledparts, Global:Temp1 );
             };
         };
   } );

     /***********************************
     ****  FUNCTION: Resetdrawsize
     ***********************************/
MakeFunction( Resetdrawsize, [],
   {
   ForAll [ x|Family ]
         {
         x:Drawsize = 0;
         };
   } );

     /***********************************
     ****  FUNCTION: Aggregatefamilyunits
```

```
    ***********************************/
MakeFunction( Aggregatefamilyunits, [],
   {
   PostBusy( ON, "Please Wait...Writing Draw Sizes" );
   Resetdrawsize( );
   ForAll [ x|Products ]
        {
        If ( x:ScheduledYesorNo #= YES )
          Then {
              Global:Fnum = F # x:Familycode;
              Global:Fnum:Drawsize = Global:Fnum:Drawsize
                + x:ProductionSize;
              };
        };
   OpenWriteFile( "c:\panelsch\draw.dat" );
   WriteLine( "Family                Draw" );
   ForAll [ x|Family ]
        {
        Global:Temp1 = x:Type # "  " # x:Drawsize;
        WriteLine( Global:Temp1 );
        };
   CloseWriteFile( );
   PostBusy( OFF );
   } );

    /***********************************
    **** FUNCTION: Writecartonsusage
    ***********************************/
MakeFunction( Writecartonsusage, [],
   {
   OpenWriteFile( cusage.c );
   WriteLine( "Cartons  Usage" );
   ForAll [ x|Cartons ]
        {
        Global:Num = x:ProjectedInventoryForDayT_1 - x:Available;
        Global:Temp1 = x:Type # "       " # Global:Num;
        WriteLine( Global:Temp1 );
        };
   CloseWriteFile( );
   } );

    /***********************************
    **** FUNCTION: Writeshipctnsusage
    ***********************************/
MakeFunction( Writeshipctnsusage, [],
   {
   OpenWriteFile( susage.c );
   WriteLine( "Shipctns  Usage" );
   ForAll [ x|ShippingContainers ]
        {
        Global:Num = x:ProjectedInventoryForDayT_1 - x:Available;
        Global:Temp1 = x:Type # "       " # Global:Num;
        WriteLine( Global:Temp1 );
        };
   CloseWriteFile( );
   } );

     /************************************
     ****  FUNCTION: Writelabelsusage
     ************************************/
MakeFunction( Writelabelsusage, [],
   {
   OpenWriteFile( lusage.c );
   WriteLine( "Labels  Usage" );
   ForAll [ x|Labels ]
         {
         Global:Num = x:ProjectedInventoryForDayT_1 - x:Available;
         Global:Temp1 = x:Type # "        " # Global:Num;
         WriteLine( Global:Temp1 );
         };
   CloseWriteFile( );
   } );

     /************************************
     ****  FUNCTION: Writeresourcesusage
     ************************************/
MakeFunction( Writeresourcesusage, [],
   {
   PostBusy( ON, "Please Wait...Writing Resources Usage" );
   Writecartonsusage( );
   Writeshipctnsusage( );
   Writelabelsusage( );
   PostBusy( OFF );
   } );

     /************************************
     ****  FUNCTION: Formsequence
     ************************************/
MakeFunction( Formsequence, [],
   {
   ClearList( Global:Sortedparts );
   EnumList( Global:Scheduledparts, x,
           {
           If ( x:PackingType #= CTN And x:ContainerUsage <
                2 )
             Then AppendToList( Global:Sortedparts, x );
           } );
   EnumList( Global:Scheduledparts, x,
           {
           If ( x:PackingType #= CTN_LBL And x:ContainerUsage
                < 2 )
             Then AppendToList( Global:Sortedparts, x );
           } );
   EnumList( Global:Scheduledparts, x,
           {
           If ( x:PackingType #= CTN And x:ContainerUsage >
                2 And x:ContainerUsage < 4 )
             Then AppendToList( Global:Sortedparts, x );
```

```
        } );
  EnumList( Global:Scheduledparts, x,
        {
        If ( x:PackingType #= CTN_LBL And x:ContainerUsage
             > 2 And x:ContainerUsage < 4 )
          Then AppendToList( Global:Sortedparts, x );
        } );
  EnumList( Global:Scheduledparts, x,
        {
        If ( x:PackingType #= BULK )
          Then AppendToList( Global:Sortedparts, x );
        } );
  EnumList( Global:Scheduledparts, x,
        {
        If ( x:PackingType #= CTN_PRT )
          Then AppendToList( Global:Sortedparts, x );
        } );
  Reform( );
  } );

    /************************************
     **** FUNCTION: Reform
     ************************************/
MakeFunction( Reform, [],
  {
  Global:Num = LengthList( Global:Sortedparts );
  ClearList( Global:Scheduledparts );
  For x [1 Global:Num ]
     {
     Global:Temp1 = GetNthElem( Global:Sortedparts, x );
     AppendToList( Global:Scheduledparts, Global:Temp1 );
     };
  ClearList( Global:Sortedparts );
  } );
```

APPENDIX B

LINEAR PROGRAMMING FORMULATION OF THE SCHEDULING PROBLEM

```
MAX     32 P6 + 3 P8 + 18 P10 + 40 P12 + 20 P13 + 6 P15 + 62 P16
     + 58 P18 + 24 P22 + 34 P31 + 22 P32 + 83 P34 + 52 P38 + 150 P42
     + 77 P49 + 150 P53 + 100 P55 + 100 P56 + 100 P62 + 1631 P65 + 96 P66
     + 199 P74 + 102 P76 + 68 P78 + 276 P84 + 100 P89 + 18 P92 + 100 P95
     + 66 P102 + 446 P114 + 279 P116 + 183 P118 + 86 P128 + 100 P135
     + 42 P139 + 30 P143 + 200 P146 + 100 P151 + 32 P155 + 116 P156
     + 10 P161 + 88 P163 + 100 P165 + 100 P168 + 150 P174 + 150 P175
     + 100 P176 + 143 P179 + 24 P182 + 31 P189 + 1810 P193 + 35 P206
     + 656 P226 + 376 P229 + 558 P234 + 205 P235 + 108 P285 + 114 P292
     + 68 P293 + 1351 P296 + 40 P298
SUBJECT TO
 FAF1149)  P1 + P5 + P23 + P31 + P52 + P65 + P77 + P86 + P98 + P110 + P123
     + P136 + P149 + P156 + P179 + P203 + P231 + P248 + P268 + P284 + P292
     <=   4638
 FAF3192)  P3 + P6 + P25 + P32 + P43 + P49 + P51 + P54 + P55 + P66 + P80
     + P87 + P97 + P100 + P112 + P125 + P133 + P137 + P148 + P155 + P165
     + P169 + P174 + P187 + P192 + P195 + P204 + P216 + P227 + P233 + P243
     + P251 + P252 + P270 + P279 + P285 + P293 + P294 <=   6000
 FAF3465)  P2 + P7 + P22 + P24 + P33 + P44 + P50 + P53 + P56 + P67 + P68
     + P78 + P99 + P111 + P124 + P138 + P150 + P161 + P166 + P170 + P175
     + P180 + P196 + P205 + P217 + P232 + P247 + P249 + P250 + P269 + P280
     + P283 + P295 + P296 <=   3850
 FAF3471)  P8 + P57 + P89 + P114 + P188 + P194 + P198 + P206 + P239 + P255
     <=   4500
 FAF3472)  P4 + P9 + P27 + P34 + P45 + P58 + P69 + P70 + P82 + P88 + P101
     + P113 + P126 + P134 + P139 + P158 + P167 + P171 + P176 + P190 + P193
     + P197 + P207 + P218 + P234 + P240 + P253 + P254 + P271 + P282 + P286
     + P297 <=   6050
 FAF3590)  P10 + P28 + P35 + P46 + P59 + P71 + P83 + P96 + P109 + P122
     + P132 + P140 + P152 + P160 + P172 + P177 + P191 + P199 + P209 + P219
     + P230 + P235 + P242 + P258 + P259 + P274 + P289 + P298 <=   5024
 FAF3592)  P11 + P29 + P36 + P47 + P60 + P72 + P76 + P90 + P102 + P115
     + P127 + P135 + P141 + P153 + P157 + P173 + P178 + P181 + P200 + P211
     + P220 + P226 + P237 + P244 + P260 + P261 + P277 + P281 + P288 + P299
     <=   4005
 FAF3593)  P12 + P18 + P37 + P79 + P91 + P103 + P116 + P128 + P142 + P182
     + P212 + P221 + P241 + P265 + P276 + P300 <=   1200
 FAF4339)  P13 + P20 + P38 + P61 + P92 + P104 + P117 + P143 + P183 + P213
     + P222 + P229 + P266 + P273 + P301 <=   6496
 FAF4343)  P14 + P30 + P39 + P62 + P73 + P84 + P93 + P105 + P118 + P129
     + P144 + P154 + P162 + P184 + P201 + P210 + P223 + P225 + P236 + P245
     + P262 + P263 + P275 + P287 + P302 <=   1561
 FAF4346)  P15 + P26 + P40 + P48 + P63 + P74 + P81 + P95 + P108 + P121
     + P131 + P145 + P151 + P159 + P168 + P189 + P208 + P228 + P256 + P257
     + P272 + P290 + P303 <=   5653
 FAF4372)  P16 + P21 + P41 + P106 + P119 + P146 + P163 + P185 + P214 + P267
     + P304 <=   1234
 FAF4378)  P17 + P19 + P42 + P64 + P75 + P85 + P94 + P107 + P120 + P130
     + P147 + P164 + P186 + P202 + P215 + P224 + P238 + P246 + P264 + P278
     + P291 + P305 <=   1111
 C485305)  P1 + P2 <=   1480
 C485311)  P3 + P4 <=   1556
 C440006)  P5 <=   1646
 C440029)  P6 <=   1445
 C440051)  P7 <=  1267
 C440055)  P8 <=  1557
 C440056)  P9 <=  1677
 C440073)  P10 <=  1246
 C440075)  P11 <=  1768
 C440076)  P12 <=  666
 C440110)  P13 <=  1666
 C440113)  P14 <=  1467
 C440114)  P15 <=  1175
 C440125)  P16 <=  1024
 C440127)  P17 <=  1556
C6914930)  P18 <=  1567
C6914932)  P19 <=  1567
C6914933)  P20 <=  1656
 C493028)  P21 <=  1746
 C493030)  P22 <=  701
C6720829)  P23 <=  1557
C6720830)  P24 <=  1157
C6720826)  P25 <=  1457
 C494134)  P26 <=  1357
 C494127)  P27 <=  1477
 C494115)  P28 <=  1678
 C494128)  P29 <=  1657
C6850305)  P30 <=  1628
 C483929)  P31 <=  1083
 C483936)  P32 <=  442
 C480386)  P33 <=  1499
 C484088)  P34 <=  1479
 C498683)  P35 <=  1269
 C498680)  P36 <=  1269
 C450428)  P37 <=  1267
 C500274)  P38 <=  157
 C498682)  P39 <=  1155
 C498679)  P40 <=  1156
 C450429)  P41 <=  1168
 C450409)  P42 <=  2158
C4826321)  P43 <=  1278
 C482632)  P44 + P48 + P55 + P56 + P63 + P78 + P81 + P89 + P95 + P96 + P133
     <=  1248
 C482638)  P45 + P46 + P58 + P59 + P61 + P80 + P82 + P83 + P87 + P88 + P92
     + P134 <=  1277
C6836904)  P47 <=  1276
C6725070)  P49 <=  1168
 C486131)  P52 <=  1168
 C486133)  P53 <=  1168
 C486144)  P54 <=  1136
 C482649)  P60 + P76 + P135 <=  1168
 C482633)  P62 + P84 + P90 + P93 + P94 <=  1162
 C482653)  P64 + P85 <=  1478
 C480358)  P67 <=  1475
 C480359)  P69 <=  1475
 C492591)  P77 + P86 <=  1508
 C482634)  P79 + P91 <=  1496
 C490056)  P97 <=  1478
```

```
C492696)  P98 <=  1467
C486963)  P99 <=  1479
C484298)  P100 <=  1455
C488749)  P101 <=  1455
C503263)  P102 <=  829
C503268)  P103 <=  1666
C503303)  P104 <=  1556
C489466)  P105 <=  1668
C503304)  P106 <=  1656
C503266)  P107 <=  1545
C489463)  P108 <=  1645
C489464)  P109 <=  1667
C488641)  P123 <=  1645
C488642)  P124 <=  1668
C488648)  P125 <=  1678
C488649)  P126 <=  1668
C488892)  P127 <=  1657
C488897)  P128 <=  1531
C489752)  P129 <=  1678
C488894)  P130 <=  1668
C489748)  P131 <=  1668
C489750)  P132 <=  1656
C485290)  P148 + P152 <=  1679
C485298)  P149 <=  1668
C485299)  P150 <=  5176
C450436)  P151 <=  1532
C450437)  P153 <=  1610
C450438)  P154 <=  1608
C503533)  P155 <=  1598
C503537)  P156 <=  1589
C503539)  P157 <=  1557
C503538)  P158 <=  1534
C503535)  P159 <=  1524
C503536)  P160 <=  1624
C503532)  P161 <=  1326
C503540)  P162 <=  1592
C503541)  P163 <=  1571
C503542)  P164 <=  1390
C450216)  P165 <=  1482
C450217)  P166 <=  1471
C450219)  P167 <=  4170
C450221)  P168 <=  3133
C450396)  P169 + P171 + P172 <=  1490
C450397)  P170 + P173 <=  1493
C487188)  P192 <=  1484
C487170)  P193 <=  773
C487071)  P194 + P198 <=  1488
C487049)  P195 <=  4189
C487074)  P196 <=  4476
C487070)  P197 <=  4287
C489293)  P199 <=  3397
C489294)  P200 <=  3296
C489295)  P201 <=  1387
C489296)  P202 <=  1378
```

```
 C497887)  P203 <=  1378
 C498113)  P204 <=  1378
 C498112)  P205 <=  1357
 C498114)  P206 <=  1401
 C488733)  P207 <=  1289
 C488042)  P208 <=  1286
 C488041)  P209 <=  1295
 C488045)  P210 <=  1296
 C488044)  P211 <=  1275
 C498116)  P212 <=  1275
 C498499)  P213 <=  1275
 C498698)  P214 <=  1265
 C498118)  P215 <=  1290
 C498261)  P216 <=  1290
C6913733)  P217 <=  1290
C6913726)  P218 <=  1470
C6913728)  P219 <=  1470
C6913730)  P220 <=  1470
C6929114)  P221 <=  1470
C6913731)  P222 <=  1491
C6913732)  P223 <=  1492
C6929116)  P224 <=  1491
 C498844)  P225 <=  1459
 C498845)  P226 <=  1141
 C498812)  P227 <=  4836
 C498837)  P228 <=  4823
 C498840)  P229 <=  4823
 C498843)  P230 <=  4483
 C481375)  P231 + P233 <=  4823
 C481974)  P232 <=  4823
 C483078)  P234 <=  4565
 C481405)  P235 + P236 + P237 <=  4272
 C481401)  P238 <=  2823
 C488799)  P249 <=  8222
 C488797)  P251 <=  2821
 C488577)  P253 <=  2821
 C488584)  P256 <=  2821
 C488585)  P258 <=  2821
 C488586)  P260 <=  2281
 C488587)  P262 <=  2281
 C494149)  P268 <=  2281
C6918873)  P269 <=  2281
C6805187)  P270 <=  2281
 C494161)  P271 <=  2821
 C494158)  P272 <=  2821
 C494166)  P273 <=  2821
 C494159)  P274 <=  2281
 C494167)  P275 <=  2281
 C494163)  P276 <=  2282
 C494162)  P277 <=  292
 C494168)  P278 <=  2289
 C410032)  P279 <=  2289
 C410037)  P280 <=  2289
 C410050)  P281 <=  2290
```

C410051) P282 <= 2289
C500062) P283 + P290 <= 2289
C500056) P284 + P285 + P289 <= 2073
C503970) P286 <= 2289
C500060) P287 + P291 <= 2291
C500070) P288 <= 2290
C6930092) P292 <= 1191
C498227) P293 <= 114
C6913692) P295 <= 1113
C498230) P297 <= 1134
C6913697) P298 <= 1114
C6913699) P299 <= 1134
C6913700) P300 <= 1134
C6913711) P301 <= 1134
C6913712) P302 <= 1134
C6913713) P303 <= 1134
C6926529) P304 <= 1134
C6926527) P305 <= 1134
S463827) 0.1666 P1 + 0.1666 P2 + 0.1666 P52 + 0.1666 P77 + 0.1666 P98
+ 0.1666 P123 + 0.1666 P149 + 0.1666 P203 + 0.1666 P231 + 0.1666 P233
+ 0.1666 P292 <= 1476
S497665) 0.1666 P3 + 0.1666 P4 + 0.1666 P25 + 0.1666 P27 + 0.1666 P28
+ 0.3333 P32 + 0.1666 P45 + 0.1666 P46 + 0.1666 P49 + 0.1666 P54
+ 0.1666 P58 + 0.1666 P59 + 0.1666 P61 + 0.1666 P69 + 0.1666 P82
+ 0.1666 P83 + 0.1666 P101 + 0.1666 P109 + 0.1666 P126 + 0.1666 P132
+ 0.1666 P134 + 0.1666 P148 + 0.1666 P152 + 0.1666 P165 + 0.1666 P167
+ 0.1666 P169 + 0.1666 P171 + 0.1666 P172 + 0.1666 P193 + 0.1666 P197
+ 0.1666 P199 + 0.1666 P209 + 0.1666 P218 + 0.1666 P219 + 0.1666 P230
+ 0.1666 P234 + 0.1666 P253 + 0.1666 P270 + 0.1666 P271 + 0.1666 P282
+ 0.1666 P297 + 0.1666 P298 <= 2144
S63827) 0.1666 P5 <= 2476
S97655) 0.1666 P6 <= 276
S97666) 0.1666 P7 <= 2476
S491959) 0.1666 P8 + 0.1666 P194 + 0.1666 P198 + 0.1666 P206 <= 4276
S97665) 0.1666 P9 + 0.1666 P10 <= 2398
S97671) 0.1666 P11 <= 2485
S6792017) 0.1666 P12 <= 2352
S91964) 0.1666 P13 <= 2485
S97673) 0.1666 P14 <= 2485
S97672) 0.1666 P15 <= 2475
S91965) 0.1666 P16 + 0.1666 P17 <= 2401
S491962) 0.1666 P18 + 0.1666 P103 + 0.1666 P128 + 0.3333 P212
+ 0.1666 P221 + 0.1666 P276 + 0.1666 P300 <= 2456
S491965) 0.1666 P19 + 0.1666 P21 + 0.1666 P106 + 0.1666 P107
+ 0.1666 P130 + 0.1666 P202 + 0.3333 P214 + 0.3333 P215 + 0.1666 P224
+ 0.1666 P238 + 0.1666 P304 + 0.1666 P305 <= 2475
S491964) 0.1666 P20 + 0.1666 P104 + 0.3333 P163 + 0.3333 P164
+ 0.3333 P213 + 0.1666 P222 + 0.1666 P229 + 0.1666 P301 <= 2475
S497666) 0.1666 P22 + 0.1666 P24 + 0.1666 P43 + 0.1666 P44 + 0.1666 P48
+ 0.1666 P53 + 0.1666 P55 + 0.1666 P56 + 0.1666 P63 + 0.1666 P67
+ 0.1666 P78 + 0.1666 P81 + 0.1666 P99 + 0.1666 P108 + 0.1666 P124
+ 0.1666 P131 + 0.1666 P133 + 0.1666 P150 + 0.1666 P151 + 0.1666 P196
+ 0.1666 P205 + 0.1666 P208 + 0.1666 P217 + 0.1666 P232 + 0.1666 P249
+ 0.1666 P269 + 0.1666 P272 + 0.1666 P280 + 0.1666 P295 <= 2293

```
S6703457)  0.1666 P23 < =   2475
 S497672)  0.1666 P26 + 0.1666 P168 + 0.1666 P228 + 0.1666 P256
     + 0.1666 P303 < =   2302
 S497671)  0.1666 P29 + 0.1666 P60 + 0.1666 P76 + 0.1666 P102 + 0.1666 P127
     + 0.1666 P135 + 0.1666 P153 + 0.1666 P200 + 0.1666 P211 + 0.1666 P220
     + 0.1666 P226 + 0.1666 P260 + 0.1666 P281 + 0.1666 P299 < =   2284
 S497673)  0.1666 P30 + 0.1666 P73 + 0.1666 P144 + 0.1666 P154
     + 0.1666 P170 + 0.1666 P173 + 0.1666 P184 + 0.1666 P201 + 0.1666 P223
     + 0.1666 P225 + 0.1666 P235 + 0.1666 P236 + 0.1666 P237 + 0.1666 P245
     + 0.1666 P262 + 0.1666 P263 + 0.1666 P273 + 0.1666 P275 + 0.1666 P277
     + 0.1666 P302 < =   2383
 S492126)  0.3333 P31 + 0.3333 P86 + 0.3333 P156 + 0.3333 P284
     + 0.3333 P285 + 0.3333 P289 < =   2342
 S492116)  0.3333 P33 + 0.3333 P95 + 0.3333 P96 + 0.3333 P290 < =   2475
 S492145)  0.3333 P34 + 0.3333 P35 + 0.3333 P87 + 0.3333 P88 + 0.3333 P92
     + 0.3333 P158 + 0.3333 P160 + 0.3333 P286 < =   2465
 S492135)  0.3333 P36 + 0.3333 P157 + 0.3333 P288 < =   2463
 S492143)  0.3333 P37 < =   2463
 S492101)  0.3333 P38 < =   23307
S6792362)  0.3333 P39 + 0.3333 P162 < =   2373
S6784902)  0.3333 P40 + 0.3333 P159 < =   2373
S6724263)  0.1666 P41 + 0.1666 P64 + 0.3333 P85 + 0.1666 P268 + 0.1666 P278
     < =   2463
S6828149)  0.3333 P42 < =   2463
S6912055)  0.1666 P47 < =   263
S6744503)  0.1666 P51 + 0.1666 P66 + 0.1666 P112 + 0.1666 P137
     + 0.1666 P187 + 0.1666 P243 + 0.1666 P252 + 0.1666 P294 < =   2258
 S497664)  0.1666 P62 + 0.3333 P84 + 0.1666 P105 + 0.1666 P129
     + 0.1666 P210 < =   2257
S6699416)  0.1666 P65 + 0.1666 P110 + 0.1666 P136 + 0.1666 P179
     + 0.1666 P248 < =   2058
S6721754)  0.1666 P68 + 0.1666 P111 + 0.1666 P138 + 0.1666 P180
     + 0.1666 P247 + 0.1666 P250 + 0.1666 P296 < =   2203
S6723369)  0.1666 P70 + 0.1666 P113 + 0.1666 P139 + 0.1666 P190
     + 0.1666 P240 + 0.1666 P254 < =   2734
S6709153)  0.1666 P71 + 0.1666 P122 + 0.1666 P140 + 0.1666 P191
     + 0.1666 P242 + 0.1666 P259 < =   248
S6695369)  0.1666 P72 + 0.1666 P115 + 0.1666 P141 + 0.1666 P181
     + 0.1666 P244 + 0.1666 P261 < =   1158
S6757005)  0.1666 P74 + 0.1666 P121 + 0.1666 P145 + 0.1666 P189
     + 0.1666 P257 < =   973
S6757050)  0.1666 P75 + 0.1666 P120 + 0.1666 P147 + 0.1666 P186
     + 0.1666 P246 + 0.1666 P264 < =   1158
 S492017)  0.1666 P79 < =   2145
 S497655)  0.1666 P80 + 0.1666 P97 + 0.1666 P100 + 0.1666 P125
     + 0.3333 P155 + 0.1666 P192 + 0.1666 P195 + 0.1666 P204 + 0.1666 P207
     + 0.1666 P216 + 0.1666 P227 + 0.1666 P251 + 0.1666 P258 + 0.1666 P274
     + 0.1666 P279 + 0.1666 P293 < =   992
S6828112)  0.3333 P89 + 0.3333 P161 + 0.3333 P283 < =   1116
 S492131)  0.3333 P90 + 0.3333 P93 + 0.3333 P94 + 0.3333 P287 + 0.3333 P291
     < =   58
 S492134)  0.3333 P91 < =   58
S6757004)  0.1666 P114 + 0.1666 P188 + 0.1666 P239 + 0.1666 P255 < =   2258
S6757051)  0.1666 P116 + 0.1666 P142 + 0.1666 P182 + 0.1666 P241
```

```
    + 0.1666 P265 <=   14
S6709762)  0.1666 P117 + 0.1666 P143 + 0.1666 P183 + 0.1666 P266 <=   58
 S497633)  0.1666 P118 <=   58
S6792304)  0.1666 P119 + 0.1666 P146 + 0.1666 P185 + 0.1666 P267 <=   58
 S475010)  0.1666 P166 <=   58
L6700719)  P23 + P24 + P25 + P30 <=   22538
 L475007)  P43 + P44 + P46 + P48 <=   3438
  L75007)  P45 <=   2168
L6912017)  P47 <=   2438
 L499526)  P51 <=   2438
 L200751)  P55 <=   2438
 L200752)  P56 <=   238
 L200755)  P58 <=   2438
 L200756)  P59 <=   2438
 L200757)  P60 <=   2440
 L200758)  P61 <=   2440
 L200760)  P62 <=   1434
 L200761)  P63 <=   1440
 L200762)  P64 <=   1440
 L245201)  P65 + P66 + P68 + P71 + P72 + P73 + P74 + P75 <=   1310
 L499527)  P110 + P111 + P112 + P114 + P115 + P117 + P118 + P120 + P121
    <=   7220
 L475008)  P133 + P134 + P135 <=   8540
 L484536)  P179 + P180 + P181 + P183 + P186 + P187 + P188 + P189 + P239
    <=   5830
 L484543)  P241 + P243 + P244 + P245 + P246 + P247 <=   8140
 L484541)  P248 + P250 + P252 + P255 + P257 + P261 + P263 + P264 + P266
    <=   1910
   PROD1)  P1 <=   0
   PROD2)  P2 <=   0
   PROD3)  P3 <=   0
   PROD4)  P4 <=   0
   PROD5)  P5 <=   0
   PROD6)  P6 <=   756
   PROD7)  P7 <=   0
   PROD8)  P8 <=   12
   PROD9)  P9 <=   0
  PROD10)  P10 <=   192
  PROD11)  P11 <=   0
  PROD12)  P12 <=   395
  PROD13)  P13 <=   240
  PROD14)  P14 <=   0
  PROD15)  P15 <=   96
  PROD16)  P16 <=   804
  PROD17)  P17 <=   0
  PROD18)  P18 <=   402
  PROD19)  P19 <=   0
  PROD20)  P20 <=   0
  PROD21)  P21 <=   0
  PROD22)  P22 <=   264
  PROD23)  P23 <=   0
  PROD24)  P24 <=   0
  PROD25)  P25 <=   0
  PROD26)  P26 <=   0
```

```
PROD27)   P27 < =   0
PROD28)   P28 < =   0
PROD29)   P29 < =   0
PROD30)   P30 < =   0
PROD31)   P31 < =   195
PROD32)   P32 < =   396
PROD33)   P33 < =   0
PROD34)   P34 < =   801
PROD35)   P35 < =   0
PROD36)   P36 < =   0
PROD37)   P37 < =   0
PROD38)   P38 < =   396
PROD39)   P39 < =   0
PROD40)   P40 < =   0
PROD41)   P41 < =   0
PROD42)   P42 < =   171
PROD43)   P43 < =   0
PROD44)   P44 < =   0
PROD45)   P45 < =   0
PROD46)   P46 < =   0
PROD47)   P47 < =   0
PROD48)   P48 < =   0
PROD49)   P49 < =   216
PROD50)   P50 < =   0
PROD51)   P51 < =   0
PROD52)   P52 < =   0
PROD53)   P53 < =   300
PROD54)   P54 < =   0
PROD55)   P55 < =   798
PROD56)   P56 < =   798
PROD57)   P57 < =   0
PROD58)   P58 < =   0
PROD59)   P59 < =   0
PROD60)   P60 < =   0
PROD61)   P61 < =   0
PROD62)   P62 < =   396
PROD63)   P63 < =   0
PROD64)   P64 < =   0
PROD65)   P65 < =   804
PROD66)   P66 < =   804
PROD67)   P67 < =   0
PROD68)   P68 < =   0
PROD69)   P69 < =   0
PROD70)   P70 < =   0
PROD71)   P71 < =   0
PROD72)   P72 < =   0
PROD73)   P73 < =   0
PROD74)   P74 < =   798
PROD75)   P75 < =   0
PROD76)   P76 < =   798
PROD77)   P77 < =   0
PROD78)   P78 < =   402
PROD79)   P79 < =   0
PROD80)   P80 < =   0
```

```
 PROD81)   P81 <=   0
 PROD82)   P82 <=   0
 PROD83)   P83 <=   0
 PROD84)   P84 <=   801
 PROD85)   P85 <=   0
 PROD86)   P86 <=   0
 PROD87)   P87 <=   0
 PROD88)   P88 <=   0
 PROD89)   P89 <=   396
 PROD90)   P90 <=   0
 PROD91)   P91 <=   0
 PROD92)   P92 <=   246
 PROD93)   P93 <=   0
 PROD94)   P94 <=   0
 PROD95)   P95 <=   795
 PROD96)   P96 <=   0
 PROD97)   P97 <=   0
 PROD98)   P98 <=   0
 PROD99)   P99 <=   0
PROD100)   P100 <=   0
PROD101)   P101 <=   0
PROD102)   P102 <=   396
PROD103)   P103 <=   0
PROD104)   P104 <=   0
PROD105)   P105 <=   0
PROD106)   P106 <=   0
PROD107)   P107 <=   0
PROD108)   P108 <=   0
PROD109)   P109 <=   0
PROD110)   P110 <=   0
PROD111)   P111 <=   0
PROD112)   P112 <=   0
PROD113)   P113 <=   0
PROD114)   P114 <=   294
PROD115)   P115 <=   0
PROD116)   P116 <=   804
PROD117)   P117 <=   0
PROD118)   P118 <=   402
PROD119)   P119 <=   0
PROD120)   P120 <=   0
PROD121)   P121 <=   0
PROD122)   P122 <=   0
PROD123)   P123 <=   0
PROD124)   P124 <=   0
PROD125)   P125 <=   0
PROD126)   P126 <=   0
PROD127)   P127 <=   0
PROD128)   P128 <=   396
PROD129)   P129 <=   0
PROD130)   P130 <=   0
PROD131)   P131 <=   0
PROD132)   P132 <=   0
PROD133)   P133 <=   0
PROD134)   P134 <=   0
```

```
PROD135)   P135 <=   618
PROD136)   P136 <=   0
PROD137)   P137 <=   0
PROD138)   P138 <=   0
PROD139)   P139 <=   390
PROD140)   P140 <=   0
PROD141)   P141 <=   0
PROD142)   P142 <=   0
PROD143)   P143 <=   300
PROD144)   P144 <=   0
PROD145)   P145 <=   0
PROD146)   P146 <=   792
PROD147)   P147 <=   0
PROD148)   P148 <=   0
PROD149)   P149 <=   0
PROD150)   P150 <=   0
PROD151)   P151 <=   198
PROD152)   P152 <=   0
PROD153)   P153 <=   0
PROD154)   P154 <=   0
PROD155)   P155 <=   795
PROD156)   P156 <=   801
PROD157)   P157 <=   0
PROD158)   P158 <=   0
PROD159)   P159 <=   0
PROD160)   P160 <=   0
PROD161)   P161 <=   204
PROD162)   P162 <=   0
PROD163)   P163 <=   600
PROD164)   P164 <=   0
PROD165)   P165 <=   258
PROD166)   P166 <=   0
PROD167)   P167 <=   0
PROD168)   P168 <=   396
PROD169)   P169 <=   0
PROD170)   P170 <=   0
PROD171)   P171 <=   0
PROD172)   P172 <=   0
PROD173)   P173 <=   0
PROD174)   P174 <=   800
PROD175)   P175 <=   800
PROD176)   P176 <=   550
PROD177)   P177 <=   0
PROD178)   P178 <=   0
PROD179)   P179 <=   396
PROD180)   P180 <=   0
PROD181)   P181 <=   0
PROD182)   P182 <=   210
PROD183)   P183 <=   0
PROD184)   P184 <=   0
PROD185)   P185 <=   0
PROD186)   P186 <=   0
PROD187)   P187 <=   0
PROD188)   P188 <=   0
```

```
PROD189)  P189 <=  396
PROD190)  P190 <=  0
PROD191)  P191 <=  0
PROD192)  P192 <=  0
PROD193)  P193 <=  600
PROD194)  P194 <=  0
PROD195)  P195 <=  0
PROD196)  P196 <=  0
PROD197)  P197 <=  0
PROD198)  P198 <=  0
PROD199)  P199 <=  0
PROD200)  P200 <=  0
PROD201)  P201 <=  0
PROD202)  P202 <=  0
PROD203)  P203 <=  0
PROD204)  P204 <=  0
PROD205)  P205 <=  0
PROD206)  P206 <=  48
PROD207)  P207 <=  0
PROD208)  P208 <=  0
PROD209)  P209 <=  0
PROD210)  P210 <=  0
PROD211)  P211 <=  0
PROD212)  P212 <=  0
PROD213)  P213 <=  0
PROD214)  P214 <=  0
PROD215)  P215 <=  0
PROD216)  P216 <=  0
PROD217)  P217 <=  0
PROD218)  P218 <=  0
PROD219)  P219 <=  0
PROD220)  P220 <=  0
PROD221)  P221 <=  0
PROD222)  P222 <=  0
PROD223)  P223 <=  0
PROD224)  P224 <=  0
PROD225)  P225 <=  0
PROD226)  P226 <=  804
PROD227)  P227 <=  0
PROD228)  P228 <=  0
PROD229)  P229 <=  390
PROD230)  P230 <=  0
PROD231)  P231 <=  0
PROD232)  P232 <=  0
PROD233)  P233 <=  0
PROD234)  P234 <=  798
PROD235)  P235 <=  798
PROD236)  P236 <=  0
PROD237)  P237 <=  0
PROD238)  P238 <=  0
PROD239)  P239 <=  0
PROD240)  P240 <=  0
PROD241)  P241 <=  0
PROD242)  P242 <=  0
```

```
PROD243)  P243 <=  0
PROD244)  P244 <=  0
PROD245)  P245 <=  0
PROD246)  P246 <=  0
PROD247)  P247 <=  0
PROD248)  P248 <=  0
PROD249)  P249 <=  0
PROD250)  P250 <=  0
PROD251)  P251 <=  0
PROD252)  P252 <=  0
PROD253)  P253 <=  0
PROD254)  P254 <=  0
PROD255)  P255 <=  0
PROD256)  P256 <=  0
PROD257)  P257 <=  0
PROD258)  P258 <=  0
PROD259)  P259 <=  0
PROD260)  P260 <=  0
PROD261)  P261 <=  0
PROD262)  P262 <=  0
PROD263)  P263 <=  0
PROD264)  P264 <=  0
PROD265)  P265 <=  0
PROD266)  P266 <=  0
PROD267)  P267 <=  0
PROD268)  P268 <=  0
PROD269)  P269 <=  0
PROD270)  P270 <=  0
PROD271)  P271 <=  0
PROD272)  P272 <=  0
PROD273)  P273 <=  0
PROD274)  P274 <=  0
PROD275)  P275 <=  0
PROD276)  P276 <=  0
PROD277)  P277 <=  0
PROD278)  P278 <=  0
PROD279)  P279 <=  0
PROD280)  P280 <=  0
PROD281)  P281 <=  0
PROD282)  P282 <=  0
PROD283)  P283 <=  0
PROD284)  P284 <=  0
PROD285)  P285 <=  654
PROD286)  P286 <=  0
PROD287)  P287 <=  0
PROD288)  P288 <=  0
PROD289)  P289 <=  0
PROD290)  P290 <=  0
PROD291)  P291 <=  0
PROD292)  P292 <=  390
PROD293)  P293 <=  600
PROD294)  P294 <=  0
PROD295)  P295 <=  0
PROD296)  P296 <=  798
```

```
 PROD297)  P297 <=  0
 PROD298)  P298 <=  312
 PROD299)  P299 <=  0
 PROD300)  P300 <=  0
 PROD301)  P301 <=  0
 PROD302)  P302 <=  0
 PROD303)  P303 <=  0
 PROD304)  P304 <=  0
 PROD305)  P305 <=  0
END
```

VITA [illegible]

Krishnaswami Ravi

Candidate for the Degree of

Master of Science

Thesis: IMPLEMENTATION OF AN EXPERT SYSTEM FOR PRODUCTION SCHEDULING USING OBJECT-ORIENTED TECHNIQUES

Major Field: Computer Science

Biographical:

Personal Data: Born in Madras, India, June 18, 1968, the son of Krishnaswamy, N.S. and Kalyani, S.

Education: Graduated from Santhome Higher Secondary School, Madras, India, in 1985; received Bachelor of Engineering Degree in Computer Engineering from Madurai Kamaraj University, India in June, 1989; completed requirements for the Master of Science degree at Oklahoma State University in May, 1992.

Professional Experience: Research Assistant, College of Business Administration, Oklahoma Stata University, January 1990, to January, 1992.