AN ABSTRACT OF THE THESIS OF

Darwin Sitompul for the degree of Doctor of Philosophy in Industrial and Manufacturing Engineering presented on October 16, 1991.

Title: Design and Implementation of a Heuristic-based Decision Support System for Nurse Scheduling

Abstract approved: _____

Sabah Randhawa

A decision support system (DSS) for nurse scheduling in hospitals is developed and implemented on microcomputer. The system includes algorithms and databases for developing weekly work and shift patterns and combining these into working schedules for nurses for a specified time horizon, and interface modules for the user to interact with the system. The system combines heuristic modeling with decision analysis concepts to generate nurse schedules. A heuristic best-first search technique is used in implementing pattern generation and screening process to satisfy both nurses and hospital's objectives.

Emphasis in the design of the DSS has been on computational efficiency and user acceptability. The system is flexible so that it can be implemented in different hospital environments, and incorporates a wide range of hospital and nurses' objectives.

Design and Implementation of a Heuristic-based Decision Support System for Nurse Scheduling

by

Darwin Sitompul

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of
Doctor of Philosophy

Completed October 16, 1991

Commencement June 1992

APPROVED:

Redacted for Privacy

Associate Professor of Industrial and Manufacturing Engineering in charge of major

Redacted for Privacy

Head of department of Industrial and Manufacturing Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented October 16, 1991

Typed by Darwin Sitompul for Darwin Sitompul

ACKNOWLEDGEMENTS

I would like to acknowledge my appreciation to many individuals who have been helpful to me for completing this work and have made my stay at Oregon State University a fruitful learning experience.

I am most thankful to my major professor, Dr. Sabah Randhawa. Without his continuous support, guidance and encouragement this thesis would not have been possible.

Thanks are also given to all my committee members, Dr. Robert R. Safford, Dr. Rasaratnam Logen Logendran, Dr. John Sessions, Dr. Eugene A. Abrassart and Dr. David Birkes for their support, suggestion, and guidance regarding my thesis.

Special thanks are also extended to Mr. Jim Beecroft of Good Samaritan Hospital, Corvallis, Oregon for his assistance during the research.

I also want to express my appreciation to my family. My mother passed away four years ago when this work was about to begin, and my father passed away four months ago when this work was about to complete. They wanted so much to see me finish this work. I have been deeply obliged to them for completing this work, which now I do.

Semoga Allah menerima mereka di tempat yang sebaik-baiknya, Amin.

And finally, I want to express my deep appreciation and

thanks to my wife Jenny, and my two children, Dany and Dyta, for their support, help, encouragement and sacrifice.

My study at Oregon State University was funded by PIU-ADB-USU, Medan, Indonesia and Proyek Pengembangan Perguruan Tinggi Bantuan Luar Negeri Bank Dunia XXI, Departemen Pendidikan dan Kebudayaan Republik Indonesia.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDIX FIGURES

DESIGN AND IMPLEMENTATION OF A HEURISTIC-BASED DECISION SUPPORT SYSTEM FOR NURSE SCHEDULING

CHAPTER 1
INTRODUCTION

The problem of manpower scheduling exists in almost all industries including manufacturing, service and transportation. Each industry has its own specific manpower scheduling problems. These problems are a function of the type of the industry and services provided. For example, the objectives and constraints of scheduling telephone operators may not be the same as that of hospital nurses, or airline crews. However, some general characteristics are common in manpower scheduling problems. These include:

1. The operation usually exceeds the normal 8 working hours per day and 5 working days a week. In some industries, once a process is started, it must be continued until completed, and the time needed to complete the process may exceed the normal working hours. Some operations work 24 hours around the clock, 7 days a week (e.g. telephone operators, nurses and police), with more than one shift every day.

2. Demand for services tends to fluctuate over the course of the day, and over the day of the week, as for example in telephone service, fire department, ambulance and nurses in hospitals. This means that the number of employees needed for each shift also varies.

3. There are some regulations that must be followed (e.g., labor and state laws) due to the specific nature of the

operations and physical limitations of humans. People cannot work continuously for more than a certain length of time; individuals need rest, sleep, days off, recreation, etc., and human effort cannot be inventoried.

All these characteristics, plus several other constraints, such as cost, special preference of the employees, differences in skill of the employees (which causes differences in labor costs), differences in skill needed to perform a certain job or service, and emergency requests, make the problem of manpower scheduling complicated, difficult, frustrating and time consuming [Howell, 1966; Henderson and Berry, 1976; and Smith and Wiggins, 1977].

1.1 Nurse Scheduling

1.1.1 Problem

Scheduling nurses in hospitals is a problem that has attracted the attention of many researchers. Ballantyne [1979] points out that personnel costs make up approximately 70 percent of the hospital expenses, and most of those expenses are attributable to the nurses. In other words, nursing cost is the largest single cost element in the hospital [Maier-Rothe and Wolfe, 1973; Ahuja and Sheppard, 1975; Kao and Queyranne, 1985; and Smith-Daniels, Schweikhart, and Smith-

Daniels, 1988]. Warner [1976], and Warner, Holloway and Grazier [1984] estimated that nursing personnel accounts for about one-third of the hospital budget. Professor Karyn Holm of University of Illinois, Chicago, estimated that this figure is now approximately 25 percent, due to the improvement of technology and management method [Arbeiter, 1988b], but is still the largest single cost element in the hospital.

Smith-Daniels, Schweikhart, and Smith-Daniels [1988] point out that health care expenditures in the United States have rapidly increased from 5 percent of 1965 GNP to a level of nearly 11 percent of 1986 GNP. Therefore, in order to minimize costs while still keeping a high quality of services, scheduling nurses is critical to the hospital.

Nursing shortage in the United States is also still an important consideration. Many hospitals are forced to hire foreign nurses [Arbeiter, 1988a] and marketing companies to help them in doing nurse recruitment [Perry, 1989]. Boston and Karzel [1987] wrote that in December 1986, the American Organization of Nurse Executives reported the national vacancy rate for RNs was 13.6 percent, nearly double the rate in 1985. Perry [1989] shows that based on a survey by ECS, a Fort Lee, New Jersey-based firm, the nurse vacancy rate in 1988 is 10.5 percent nationally. Although this rate is slightly lower than 1986 rates, ECS forecasts that the rate will increase till the year 2000.

Besides the high percentage of nursing shortage, hospi-

tals also face the high rates of nurse turnover. According to Perry [1989], American Hospital Association (AHA) reported that the turnover rate is 20 percent nationally, and in some hospitals more than half of the nursing staff quits each year. Arbeiter [1988b] points out that better pay, **flexible scheduling**, greater control over practices, **sufficient staffing**, recognition, respect, and tuition reimbursement are several factors that affect the recruitment strategy. The author shows some reports of successful hospitals practicing flexible schedules for nurses, in keeping the turnover rate low.

These facts indicate that a better method of scheduling nurses will be a significant contribution to solving the problem. With a good schedule, nurses can work efficiently and the turnover rate can be reduced. Better schedules can benefit hospitals because it will reduce the operation costs. Adequate staffing because of a good schedule can improve and guarantee high quality service. An easy and computerized schedule can free nursing personnel who otherwise have to spend time making manual schedules to give more attention to the patient care instead of doing administrative work.

Although the main objective of scheduling nurses in hospital can be simply stated as provision of adequate staffing to meet patient needs at all times [Frances, 1966], this objective is very difficult to accomplish. This is because there are other related hospital and nurse's objectives linked directly or indirectly to the main objective,

with each objective having its own constraints.

The complexity and difficulty in nurse scheduling is due to several reasons including the possible conflict of the hospital and nurse's objectives, the different levels of nurses, and the different requirements of individual nurses.

Livengood [1965] asserts that adequate scheduling is not wholly a mathematical problem, it is a human one as well. Morale of the nursing staff often depends on equitable sharing of working time and acceptable shift rotation policies. A planned schedule is preferred by the nurses because they like to be able to plan their days off ahead of time for personal and family activities.

It is also claimed that a planned schedule will benefit the hospital as well. The administrator, head nurse, or other individuals responsible for work schedules can see the exact needs for personnel in advance, and the part time nurse can also know when their services are needed [Livengood, 1965].

1.1.2 Scheduling Practices

Livengood [1965] describes three basic steps for scheduling nurses:

1. Determining actual work load for nursing personnel for each division and each shift within the division.
2. Developing a staff pattern to answer the work load

requirements.

3. Establishing the work schedules.

In general, there are three basic scheduling arrangements [Livengood, 1965]:

1. **Straight shift** - each nurse or team works straight day, evening, or night shift.
2. **Rotating shift** - each nurse or team works a certain period on the day shift, then similarly on the other two shifts.
3. **Alternating shift** - each nurse or team alternates between two shifts.

Some hospitals use a combination of these three arrangements [Livengood, 1965].

In most hospitals, scheduling is usually done on a ward basis by a head nurse due to the particular specialization of the nurses on a ward and the fact that the number of nurses needed to work a particular shift will often vary from day to day [Rosenbloom and Goertzen, 1987]. This task, until now, is mostly performed manually [Ozkarahan, 1987]. Mathematical models, heuristic methods and some other approaches have been proposed and applied to the nurse scheduling problem. Computers have been used to help the scheduler to generate schedule. Sitompul and Randhawa [1990] point out that in general there are four categories of nurse scheduling models which are a

combination of scheduling arrangement (cyclic and non-cyclic), and solution methodology (optimization and heuristic).

In spite of the fact that many models have already been proposed to this problem, ranging from completely decentralized and manual to centralized computer controlled, no one method is widely accepted. Warner [1976] showed that from 13 cyclical scheduling systems surveyed, only one had survived over one year. Rosenbloom and Goertzen [1987] point out that the previous approaches have not been widely implemented due to two primary reasons:

1. Some models need enormous amount of computer resources, therefore they are impractical for use on a ward basis.
2. Some other models are problem specific, therefore inflexible. Any change in the system may result in the method being scrapped.

Ozkarahan and Bailey [1988] also conclude that the present nurse scheduling models do not suffice the need. They argue that some models previously presented have neglected many constraints to obtain practical solutions; others are very inflexible and user interaction is almost impossible. The other problem they found was that none of the previous approaches considered integration of the time of day, and day of week problems.

1.2 Research Objectives

There were two objectives of this research. The first objective was to develop a new structure of decision support system (DSS) framework for nurse scheduling in hospitals; the second objective was to develop and implement a new heuristic procedure of generating schedules for nurses under the proposed DSS framework.

1.3 Methodology

The research was implemented in two steps. The first step consisted of developing the framework for the new DSS model; in the second step, a new heuristic procedure of generating schedules for nurses that considers both the nurses and hospital requirements was developed, implemented and validated.

1.4 Scope

The research assumes that the hospital will have a specified operating policy concerning the proportion of nurses working in different shifts. One example of shift assignment is a 3:2:1 policy, meaning that the number of nurses working in the day shift is three times the number of nurses working in the night shift, and the number of nurses working in the

evening shift is twice as much as the number of nurses working in the night shift. It is also assumed that a shift is assigned for a full one week period, meaning that the shift change can occur only at the beginning of the week and remains unchanged during the week.

This thesis limits the scope to full time nurses only. However, some suggestions for using this model for part time nurses are provided at the end of the thesis.

1.5 Contribution of this thesis

The contributions of this thesis are as follows:

1. The development of a Decision Support System framework for solving the nurse scheduling problem in hospitals that can be applied in different hospital environments.
2. The development and implementation of a microcomputer-based heuristic system for solving the multiple objectives nurse scheduling problem. The model incorporates a wide range of hospital and nurses' objectives, and represents a computationally attractive alternative to solving this complex problem.

1.6 Potential benefits of the research

The research will result in several potential benefits:

1. The use of heuristic methods will eliminate the need to

use more complex mathematical or other analytical models. Nurse schedules for up to 12 week scheduling periods can be generated in a very short computational time using a microcomputer.

2. The microcomputer implementation of the heuristic model is an important consideration. This will enhance the usefulness of the model even for small hospitals due to the smaller hardware cost associated with the personal computer. The model also has the potential of being used in hospitals in developing countries which otherwise have to do nurse scheduling manually because they cannot afford mainframe or minicomputers.
3. The use of this model will promote "objectivity" in developing nurse schedules, thus eliminating the personal bias of nurse administrator that may be introduced in scheduling. This has often resulted in nurses' dissatisfaction with schedules generated by the hospital's staff.
4. The relative ease of use of the model and the use of computer will free nurses from doing administrative jobs such as filling the job pattern request which is used by some hospital for scheduling nurses. The flexibility of the model and the ability of the model in handling hospital and nurse constraints will reduce complaints from both hospital and nurses. This will result in reduction of nurse turn over rate, and eventually will reduce costs and increase quality of health care.

5. The DSS framework and heuristic implementation can be modified to be used for manpower scheduling in other application areas.

The next chapter provides a comprehensive literature review of nurse scheduling in hospital. Chapter 3 describes the conceptual framework of the model. Chapter 4 describes the pattern generation procedure and the concept of the best-first search method used in the model. Chapter 5 discusses implementation and evaluation issues; finally Chapter 6 presents conclusions of this research and provides some suggestions for further research.

CHAPTER 2
LITERATURE REVIEW

Manpower scheduling presents a considerable problem in most organizations. Manpower scheduling problems also have some distinct characteristics. First, the demand tends to fluctuate widely, particularly over the short term, and may occur seven days a week. Second, human effort cannot be inventoried. Whereas in manufacturing, production requirements can be balanced by building inventories of materials, a nurse cannot perform a service before the demand occurs. The third attribute is that customer convenience is critical in hospitals. In addition to these three general attributes, there may be constraints that are specifically imposed by the system or the system environment, such as hospital personnel policies and nursing staff preferences for distribution of days off. All these characteristics make the manpower scheduling problem difficult to model and solve.

2.1 Scheduling Objectives

The objective of manpower scheduling in hospitals is to develop a systematic procedure for allocating nurses to work shifts and work days to ensure continuous high quality service. Most of the research in hospital operations has focused on scheduling nurses, since the amount paid in nursing staff salaries is the largest single component in the hospital

budget (Maier-Rothe and Wolfe, 1973; Ahuja and Sheppard, 1975; and Ballantyne, 1979), and because nurse scheduling directly affects the quality of patient care.

The nurse scheduling problem may be considered a multi-stage problem, even though most of the reported research has treated it as a single-stage decision problem. The stages involved are: (1) determining a set of feasible schedules that satisfy the system constraints for a specified time horizon, (2) selection of the best schedule in terms of cost, coverage and/or other criteria, (3) fine tuning the schedule to accommodate changes in staffing levels and variations in patient demand, and (4) making specific shift assignments.

The objectives in nurse scheduling are multiple. These include: use minimum staffing to avoid wasted manpower, provide adequate patient care, ensure continuity in service, and satisfy organizational scheduling policies such as work patterns. The problems is further complicated by such factors as:

- The patient demand varies 24 hours a day, seven days a week, and difficult to forecast. Thus, for example, the number of personnel needed on a particular shift may vary from day to day.
- The allocation decision must consider overall staffing levels, and qualifications and specialization of nurses available.
- Individual patients have different nursing care re-

quirements. These requirements, for example, depend on the acuity of illness of the patient. A variable mix of personnel is required to satisfy this diversity of nursing-care demand.

- Organizational structure and characteristics need to be satisfied. These include minimum required coverage, days-off policies and work patterns, admission policies, communication systems, and information and structural design of the units.
- Abnormal demand for service and unpredictable absenteeism.
- Requests from personnel including vacation, work stretch and work patterns.

Additionally, some of these considerations may be in conflict with others, such as employee requests and the need to balance work load.

Different criteria have been proposed for evaluating scheduling methodologies in hospitals. The more important criteria are (Warner, 1976c):

- Coverage, or the extent to which a schedule meets the minimum coverage requirements, and provides balanced coverage.
- Quality, or the perceived value of the schedule to the nurses in terms of work-stretch length, days off, weekends, equalization of rotation, etc.

- Flexibility, or the extent to which the scheduling system can adapt to changes in staffing level environment.
- Cost, in terms of resources consumed in making the scheduling decision.

2.2 Scheduling Approaches

The scheduling research in health care systems may be classified based on the type of scheduling and the scheduling procedure used. There are basically two types of scheduling - cyclical and non-cyclical, and two types of solution techniques - optimizing and heuristic. Some of the available scheduling models have been classified according to this schema (see Table 2.1). The names in Table 2.1 refer to the references for this thesis. Some scheduling models (for example, Arthur and Ravindran, 1981) use more than one approach for modelling different components of the problem, and may thus be classified into more than one cell of Table 2.1. However, Table 2.1 is based on the primary methodology used by the authors.

2.2.1 Cyclical Scheduling

In cyclical scheduling, each nurse work pattern is repeated in a cycle of "n" weeks indefinitely into the future.

The major advantage of this technique is that the schedules are easily generated on a consistent policy basis, and a new schedule needs to be produced only when there is a change in staffing requirements.

Type of Scheduling	Scheduling Procedure	
	Heuristic	Optimizing
Cyclical	Ahuja and Sheppard (1975) Alivizatos (1981) Arnold and Mills (1983) Frances (1966) Howell (1966) Maier-Rothe and Wolfe (1973) Megeath (1978) Morrish and O'Connor (1970) Murray (1971) Smith (1976)	Rosenbloom and Goertzen (1987)
Non-Cyclical	Arthur and Ravindran (1981) Jelinek, Zinn and Brya (1973) Smith and Wiggins (1977)	Abernathy et al. (1973) Miller, Pierskalla and Rath (1976) Musa and Saxena (1984) Ozkarahan (1987a, 1987b) Ozkarahan and Bailey (1988) Rothstein (1973) Tobon Perez (1984) Warner (1976a, 1976b, and 1976c) Warner and Prawda (1972)

Table 2.1 Classification of Scheduling Techniques

The major disadvantage of cyclical scheduling is that even though it guarantees a certain minimum coverage, it lacks flexibility in dealing with supply and demand fluctuations, and in meeting individual needs for nurses.

2.2.2 Non-Cyclical Scheduling

Non-cyclical scheduling generates a new schedule for each scheduling period with available resources and policies that attempt to satisfy a given set of constraints. The biggest advantage of non-cyclical scheduling is flexibility. However, it is time-consuming, and typically produces unstable and suboptimal schedules.

2.2.3 Optimizing Solution Techniques

These techniques employ a structure in which some objective function is optimized subject to a set of constraints.

2.2.4 Heuristic Techniques

A heuristic procedure finds a good solution, but one that is not guaranteed to be the best solution. These techniques avoid the excessive cost of using an optimizing approach. Most of the heuristic models start off by generating an initial schedule to satisfy predicted workload, and then refine the initial schedule to satisfy hospital policy constraints and accommodate individual work preferences.

As shown in Table 2.1, most of the heuristic models have focused on solving the cyclical scheduling problem whereas the

optimizing models have attempted to solve the non-cyclical problem. A brief discussion of the models available in each of the four cells in Table 1 follows.

2.3 Heuristic Approach for Cyclical Scheduling

Howell (1966) outlined the steps necessary to develop a cyclical schedule and then presented examples of how a working pattern can be developed based on this procedure. Howell's method is a step-by-step procedure for accommodating the work patterns and individual preferences of nurses given a certain service level.

Maier-Rothe and Wolfe (1973) developed a cyclical scheduling procedure that assigns different types of nurses to each unit based on average patient care requirements, hospital personnel policies, and nursing staff preferences.

Ahuja and Sheppard (1975) described a scheduling system that consists of four components: work pattern selector that selects work patterns to be repeated indefinitely for a group of nurses; work schedule assembler that converts the work patterns into work schedules for different nurse types and accommodates operating constraints; work schedule projector that allows the user to project the work schedule for a unit (or individual) over a predetermined period of time; and work load allocator that assigns nurses to units based on work load index. A work load index representing the direct and indirect

care required for a unit is computed for each unit daily. Comparing available and required staffing levels drives the reallocation decision.

Smith (1976) demonstrated the use of an interactive algorithm to help a scheduler focus on the tradeoffs necessary to produce an acceptable cyclical rotational schedule. The algorithm starts off by determining the number of full-time personnel equivalents required, develops an initial schedule, and then successively refines it to accommodate precedence violations, work stretches, split vacation days, and some other considerations.

Use of heuristic in cyclic scheduling has also been reported by Megeath (1978), Morrish and O'Connor (1970), Frances (1966), Arnold and Mills (1983) and Murray (1971). The overall concept remains the same: an initial schedule is developed and then refined based on hospital and nurse requirements.

2.4 Heuristic Approach for Non-cyclical Scheduling

Smith and Wiggins (1977) developed a computer-based system for non-cyclical scheduling to generate monthly schedules for several nurse categories. The system considers individual preferences for shifts and days off, includes part-time employees, accommodates special request for days off or particular shift assignments, and provides an interface for

scheduling clerks to make final adjustments to computer-generated schedules. Jelinek, Zinn and Brya (1973) also described a computer-based scheduling system that predicts workload on daily basis for each patient unit and schedules personnel to match the forecasted demand subject to hospital policy constraints.

2.5 Optimizing Approach for Cyclical Scheduling

Rosenbloom and Goertzen (1987) presented an integer programming (IP) based three-stage algorithm for cyclic nurse scheduling. In the first stage, a set of possible schedules is generated. This set of schedules is then screened against labor and work-pattern constraints. In stage two, the resulting subset of possible schedules are formulated as an integer program. The objective function in IP is to minimize the maximum daily coverage constraints. In the third and final stage of the scheduling algorithm, the solution from IP is converted into work patterns for each individual nurse.

2.6 Optimizing Approach for Non-cyclical Scheduling

Various optimization approaches have been used for non-cyclical scheduling. These include integer programming (Rothstein, 1973 and Tobon Perez, 1984), goal programming (Ozkarahan, 1987a, 1987b; Ozkarahan and Bailey, 1988; Musa and

Saxena, 1984; and Arthur and Ravindran, 1981), stochastic programming (Abernathy, et al., 1973), and non-linear programming (Warner, 1976b, 1976c; and Warner and Prawda, 1972). All these approaches except the goal programming approach attempt to optimize an objective function subject to a set of constraints.

The Rothstein (1973) model was specifically formulated for manpower allocation in the housekeeping operations of hospitals. The objective function for this model maximizes the number of weekly assignments with consecutively paired days off. Due to the nature of the constraints formulated in the model, the integer requirement on the variables was dropped. Then the integer program was reduced to a continuous linear program.

Tobon Perez (1984) formulated the scheduling problem as a zero-one IP problem. The objective function consists of two components. The primary objective reflects the quality of nursing care. The secondary objective reflects the nurses' combined satisfaction with a particular schedule. A heuristic approach is used to solve the resulting formulation. The primary objective is optimized first. If this optimization results in alternative optimal solutions, then the secondary objective is considered.

Abernathy, et al. (1973) formulated the nurse scheduling problem as a stochastic programming problem. They developed a non-iterative solution procedure for the formulation that

considers alternative operating procedures and source criteria, and allows for the inclusion of statistically dependent demand. Warner (1976b and 1976c) described a nurse-scheduling system formulated as a multiple-choice programming problem whose objective function quantifies preferences of individual nursing personnel concerning length of work stretch, rotation patterns and request for days off. The constraints provide for minimum numbers of nurses from each skill class to be assigned to each day and shift of a four- or six-week scheduling pattern. Warner and Prawda (1972) developed a mixed-integer quadratic programming formulation for the problem, but suggested that due to the complexity of the non-linear formulation, a linear programming formulation with post-optimality analysis may be substituted as the solution approach. Miller, Pierskalla and Roth (1976) formulated the nurse scheduling problem as a trade-off between staffing coverage and individual preferences. The problem was then solved using a cyclic coordinate descent algorithm that seeks a near-optimal solution.

The optimizing approach that has received the most attention recently is goal programming. The techniques discussed so far attempt to optimize one criterion only. In goal programming, all the objectives are assigned target levels for achievements and a relative priority for achieving these goals. Goal programming attempts to find an optimal solution that comes as close as possible to the targets in the

order of specified priorities.

The Arthur and Ravindran (1981) goal programming formulation used the following set of goals: minimum staffing requirements, desired staffing requirements, nurse preferences, and nurse special requests. The constraints corresponding to each goal are generated and added to the problem, and zero-one goal programming is used to solve the resulting problem for assigning days on and off to nurses. A heuristic procedure is then used to assign specific shifts to nurses.

Musa and Saxena (1984) use zero-one goal programming to schedule eleven nurses in one unit for the day shift over a two-week period. The goals represent the scheduling policies of the hospital and nurse preferences for weekends.

Ozkarahan (1986 and 1987a) and Ozkarahan and Bailey (1988) use goal programming as the key underlying mathematical model for a proposed nurse-scheduling decision support system. The objectives of this model are to maximize utilization of full-time staff, minimize understaffing and overstaffing costs, minimize payroll costs, and minimize part-time staff costs.

2.7 Discussion

The solution to the staffing problem requires consideration of many interrelated factors, including those dealing with organizational structure and characteristics, qualifica-

tions of personnel, and overall staffing levels. The problem is further complicated by the diversity of working preferences. What is needed is a system for scheduling nurses that is able to operate from the perspective of the entire hospital; that is, it accounts for patient needs and requirements, hospital policies, particular circumstances of each nursing unit, and preferences of individual nurses.

Most of the optimization approaches consider only a small subset of constraints. The authors generally impose their own priority structure on their models, or they use suboptimal heuristics. The mathematical models are inflexible, they fail to accommodate the dynamics involved in scheduling, and user interaction is insufficient. In addition, mathematical models that do include a large number of objectives and constraints to represent a real-time operation require large computational resources, which further limit their use in a dynamic decision-making environment.

Much of the heuristic work related to nurse scheduling has been concerned with cyclic scheduling. Cyclical schedules are easily generated, but they are rigid in the face of variation in demand. Several computerized nurse scheduling systems have been developed (Murray, 1971; Ballantyne, 1979; Blau and Sear, 1983; Veranth and Cheson, 1984; and Flanders and Lutgen, 1984). These systems are generally computer versions of scheduling heuristic, including the traditional approach where the head nurses works out a new schedule for

each scheduling period in a trial-and-error fashion.

Scheduling is a component of the nurse management system that include staffing, scheduling, and budgeting, where staffing refers to the mix (number and skill level) of nurses required for each unit to satisfy daily and shift demand, and scheduling refers to nurse allocation. Even though these three decisions deal with problems having distinct time dimensions, they are highly interdependent. Most researchers have concentrated on the scheduling problem. Where multiple decisions have been analyzed, they have been treated separately. The use of different solution approaches for each of the three decisions makes it easy to analyze each decision independent of the other two components. However, such solution approaches may lead to suboptimal results.

2.8 Use of Artificial Intelligence in Nurse Scheduling

Lukman (1986) used an artificial intelligence (AI) methodology to investigate the nurse scheduling problem. The AI-based computer system uses an abstraction of human schedulers' knowledge and information in formulating the schedule and monitoring its execution. The reasoning processes that are used in scheduling are abstracted into a series of decision rules of the form:

IF condition satisfied THEN action.

An example of such a rule is:

IF a nurse worked last weekend,

THEN she is entitled to have this weekend off.

The computer system consists of three programs. The schedule program generates unit-schedules. The tasks included in this component are nurses' requests, continuity of previous schedules, minimum coverage level, and the working pattern of nurses. The tasks are processed sequentially and all possible schedules are generated. The merge program merges all unit-schedules to form an overall schedule. Finally, the maintenance program updates daily staffing.

The use of a decision-rule structure allows the inclusion of qualitative considerations that are difficult to quantify mathematically. However, Lukman's model does not use strategic planning to predict or to determine the number of required personnel from existing or past data. Also, the model does not include any mechanism to identify or generate new rules or to modify existing rules in order to improve the schedule's quality.

There are some additional issues that need to be addressed if the rule-based concept is to be effectively used for nurse scheduling. First, procedures need to be developed to resolve the conflict that may arise if the rules are derived from multiple experts. Second, with rules generated at a specific hospital location, there is the question of the validity of the system in a different setting. Developing a rule-based system that can be used in a variety of hospital

environments would require including scheduling knowledge from experts at different hospitals in developing the decision rules, and extensive validation tests.

2.9 Decision Support Systems

Most of the research reported on nurse scheduling has tackled isolated elements of nurse scheduling problem. However, in order to be effective, nurse scheduling requires a comprehensive methodology encompassing all of the elements of the problem. One approach to developing a user-interactive, integrated approach to nurse scheduling is that of decision support system (DSS). This approach has been proposed by Nutt (1984), Ozkarahan (1987b), Ozkarahan and Bailey (1988), Tobon Perez (1984), and Bell, Hay and Liang (1986). A DSS focuses on supporting decision making and shifts attention from the operational level toward the issues of managerial problem-solving. The nurse-scheduling problem has components that can be structured, and others that require subjective assessments. The key characteristics of DSS for nurse scheduling include:

- A tendency to aim at less well-structured problem
- An attempt to combine the use of models or analytic techniques with database access and retrieval functions
- A focus on features which make the DSS easy to use in an interactive mode

- Flexibility and adaptability to accommodate changes in the environment and the decision making approach of the user

Such a DSS would draw upon the benefits of the previously proposed approaches, such as the optimizing characteristics of mathematical programming models, the speed of certain heuristic, and the representation procedures of artificial intelligence. The specifics of the nurse-scheduling problem are often undefined at the outset, whereas the use of analytic models requires a rather rigid structure to be determined beforehand. The rigid framework in which the algorithm resides cannot be altered easily to match the user's changing perception of the problem. In addition, the information requirements of analytical models often far exceed the user's patience for entering data. This can be a problem even if the model is sufficiently good representation of the system, whether it is analytic or heuristic. It is envisioned that a decision support system would (1) include both nurses' and hospitals' objectives, (2) contain a "model base" for solving specific problems, (3) modify the results from these solutions to accommodate changing requirements of individual and environment, (4) integrate the subcomponents into a system decision-making tool, and (5) provide some measure of utility of the solution generated. Such a system would immensely enhance the decision-making effectiveness and, ultimately, the quality of hospital operation.

2.10 Conclusions

The objective of nurse scheduling is to develop a systematic procedure for allocating nurses to work-shifts and work-days to ensure high-quality service. Several approaches have been proposed to accomplish this objective, including mathematical programming, heuristic, and artificial intelligence concepts. The use of these approaches has been effective in modelling specific components of the nurse scheduling problem. However, the resulting models have seen limited use because of the complexity and information intensive nature of nurse scheduling. It may be possible to utilize the concepts of decision support systems to combine analytic and heuristic approaches with information-based techniques in order to develop more adaptable and flexible nurse scheduling systems.

One way to reduce the size of the integer problem in the nurse scheduling model is to first generate some possible patterns and then screen the patterns against some common constraints. With this method, several infeasible patterns can be thrown away so that the size of mathematical formulation becomes smaller. This method has been proposed and used by Rosenbloom and Goertzen [1987].

Rosenbloom and Goertzen [1987] in their algorithm first generate a possible pattern based on a heuristic rule. This pattern is then screened in two phases. In the first phase, which they call as static elimination procedure, the pattern

is screened against some constraints such as the maximum consecutive days off. The second phase, which is called the dynamic elimination procedure, the screening process continues to check the possibility that a pattern can be combined with another pattern without violation of any constraint.

CHAPTER 3
CONCEPTUAL FRAMEWORK AND OVERALL SYSTEM DESCRIPTION

This research has two objectives. The first objective is to develop a conceptual framework for a microcomputer-based decision support system (DSS) that can be used by management of hospitals for scheduling nurses. The DSS will be interactive, easy to use and user friendly, so that it can be effectively used by a non-programmer user. For the purpose of building the DSS framework, some methods that have been proposed earlier to handle individual aspects of the scheduling problem, will be modified and integrated.

The second objective of this study is to implement the schedule generation component of proposed DSS. An heuristic approach is used for the schedule generation and schedule assignment.

3.1 The Conceptual Model

The conceptual framework of the proposed DSS model is shown in Figure 3.1. The model consists of several important components as described below:

1. **Pattern Generator.** This program generates work and shifts patterns needed for developing the schedules. **Work pattern** is a combination of '0' and '1' showing the nurses' daily work routine. The

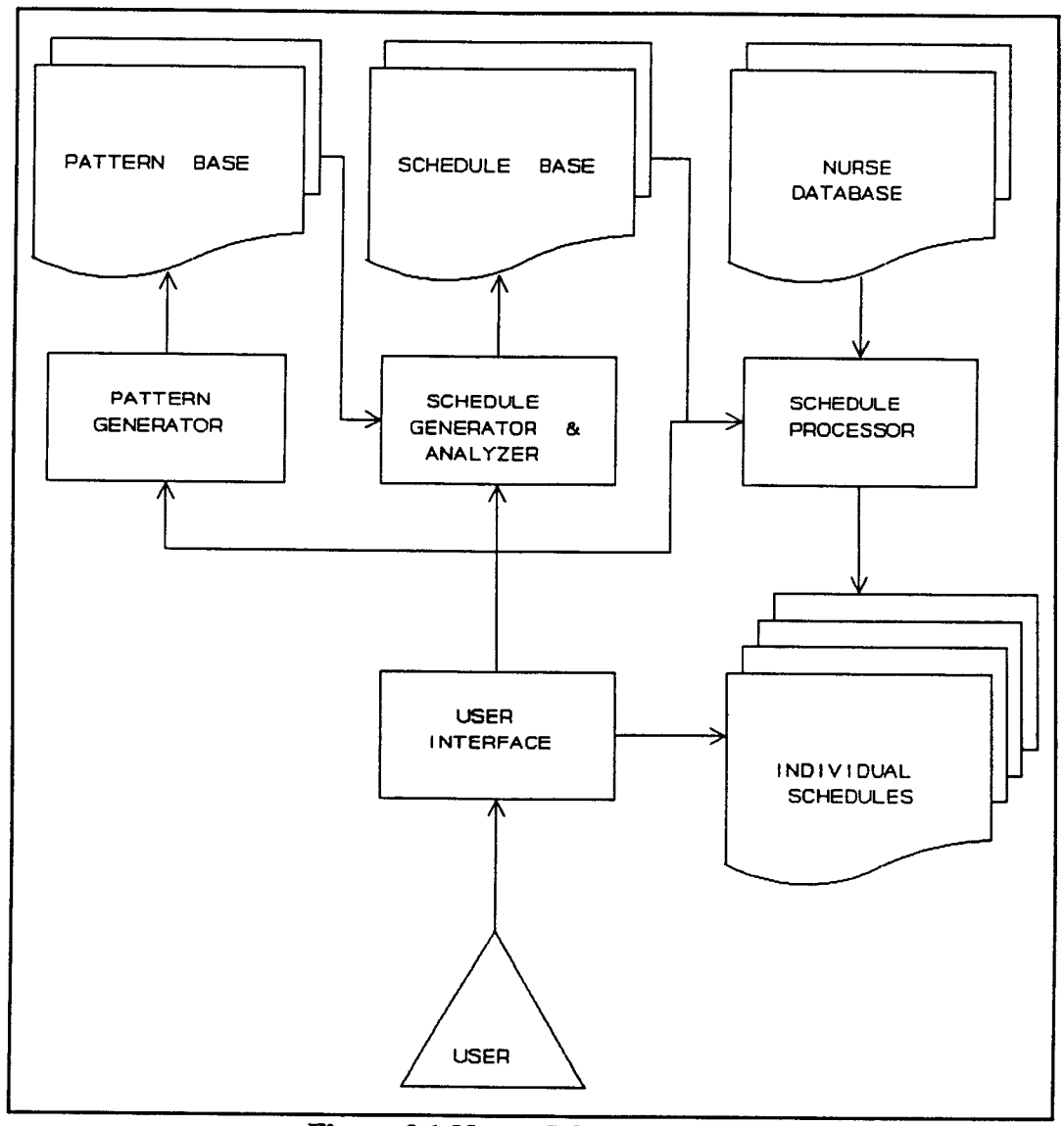

Figure 3.1 Nurse Scheduling DSS

digit '0' represents day-off, while digit '1' means day-on. For example, a one-week work pattern can be represented as '1111100' which means that the nurse works from Monday to Friday (the first five days) and has Saturday and Sunday off (the sixth and seventh day). **Shift**

pattern, is a combination of the shift for each week. Here, letters 'D', 'E' and 'N' are used to represents day, evening and night shifts, respectively. For example, for a 4-week schedule, the pattern DDEN means that the nurse works in day shift in the first and second weeks, evening shift in the third week, and night shift in the fourth week.

2. Schedule Generator and Analyzer. This program combines the shift and work patterns to form the schedule. This program will also handle the modification and extension of patterns. Once patterns have been generated, they can be modified, extended, and revised. For example, 4-week patterns can be extended to 8-week or 12-week planning horizon.

3. Schedule Processor. This program assigns schedules to the nurses.

4. Pattern Base. Pattern base is the database where the work and shift patterns are stored.

5. Schedule Base. This is the database which contains ready-to-use schedules.

6. Nurse Database. This database contains the necessary information pertaining to the nurses. The process of assigning schedule to nurses involves matching the schedule with the nurse database.

7. User Interface. This is a front-end program to aid the user in using the system.

The system developed in this research focuses on pattern and schedule generator, and associated data bases. The specific assignment of nurses to schedules (Schedule Processor) is not addressed here since this phase is hospital-specific.

3.2 Schedule Generation Process

The schedule generation process is shown in Figure 3.2 and is as follows:

1. Generate sets of work patterns in terms of n-tuple binary (**n** can be 7, 14, 21, 28 and so on, depending on the schedule period), so that different schedule lengths can be covered.

   For example, a pattern of four-week period of schedule can be represented as:

M	U	W	T	F	S	S	M	U	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S
1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0

   where 1 represents working day while 0 represents day off.

2. Generate shift patterns in term of D-E-N combination, where D stands for day shift, E for evening shifts and N for night shift. Example of four-week shifts pattern is:

   D D N E

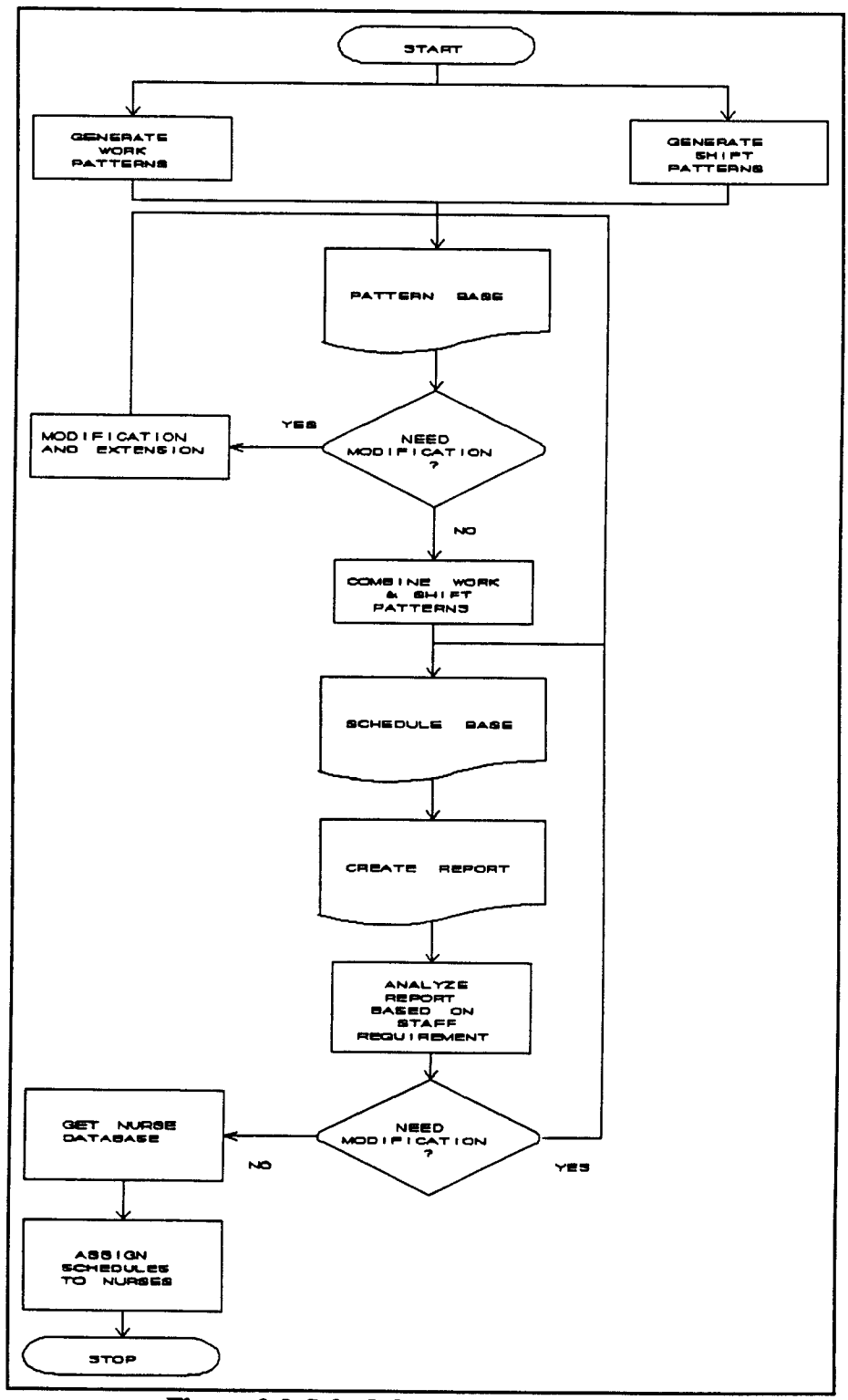

Figure 3-2 Schedule Generation Process

that is, the nurse works in the day shift for the first two weeks, night shift in the third week, and evening shift on the fourth week.

3. The work pattern and shift pattern are then combined to form a complete scheduling pattern as follows:

Day :		M	T	W	T	F	S	S
Week : 1	D:	1	1	1	1	1	0	0
Week : 2	D:	1	1	1	1	1	0	0
Week : 3	N:	1	1	1	1	1	0	0
Week : 4	E:	1	1	1	1	1	0	0

4. Modification and extension. If necessary, the patterns or the schedules can be modified or extended so that the length of schedule can be extended and the variation and number of schedules can be increased.

5. Generate scheduling reports and evaluate the schedule against the staffing requirements. If the deviation between the generated schedule and staffing requirements is beyond acceptable limits, then an alternative schedule can be evaluated. If all schedules in the schedule base have higher than acceptable deviations, then the system can back-

track to the pattern base, and generate a new set of patterns.

6. Schedule assignment. If an acceptable schedule is identified, then it is combined with nurse data (from the nurse database) to assign a specific working schedule to each nurse.

3.3 Screening Process.

During the process of pattern generation (work and shift patterns) all patterns are screened against a set of common constraints to obtain feasible scheduling patterns. To illustrate, consider a two-week planning period. The number of different patterns that can be formed for a two-week period are 16384 (2^{14}). However, because of constraints, this number can be significantly reduced. For example, consider a specific nurse that is scheduled to work 5 days a week and Friday and Saturday before the two-week period begins, has the "middle" weekend off and then must work the "end" weekend. Then there are 25 possible patterns that can be assigned to the nurse (Figure 3.3). This set of patterns can be further reduced if other constraints were to be considered. The screening method that is used in this research is a heuristic search technique called the best-first heuristic mechanism (Rich and Knight, 1991). The performance measure used by this

heuristic is the penalty cost, that is, the cost associated with the violation of the constraint. In other

Pattern Number	Previous Week F	Previous Week S	S	M	U	W	T	F	S	S	M	U	W	T	F	S	Next Week S
1.	1	1	1	0	1	1	1	1	0	0	0	1	1	1	1	1	1
2.	1	1	1	1	0	1	1	1	0	0	0	1	1	1	1	1	1
3.	1	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	1
4.	1	1	1	1	1	1	0	1	0	0	0	1	1	1	1	1	1
5.	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1
6.	1	1	1	0	1	1	1	1	0	0	1	0	1	1	1	1	1
7.	1	1	1	1	0	1	1	1	0	0	1	0	1	1	1	1	1
8.	1	1	1	1	1	0	1	1	0	0	1	0	1	1	1	1	1
9.	1	1	1	1	1	1	0	1	0	0	1	0	1	1	1	1	1
10.	1	1	1	1	1	1	1	0	0	0	1	0	1	1	1	1	1
11.	1	1	1	0	1	1	1	1	0	0	1	1	0	1	1	1	1
12.	1	1	1	1	0	1	1	1	0	0	1	1	0	1	1	1	1
13.	1	1	1	1	1	0	1	1	0	0	1	1	0	1	1	1	1
14.	1	1	1	1	1	1	0	1	0	0	1	1	0	1	1	1	1
15.	1	1	1	1	1	1	1	0	0	0	1	1	0	1	1	1	1
16.	1	1	1	0	1	1	1	1	0	0	1	1	1	0	1	1	1
17.	1	1	1	1	0	1	1	1	0	0	1	1	1	0	1	1	1
18.	1	1	1	1	1	0	1	1	0	0	1	1	1	0	1	1	1
19.	1	1	1	1	1	1	0	1	0	0	1	1	1	0	1	1	1
20.	1	1	1	1	1	1	1	0	0	0	1	1	1	0	1	1	1
21.	1	1	1	0	1	1	1	1	0	0	1	1	1	1	0	1	1
22.	1	1	1	1	0	1	1	1	0	0	1	1	1	1	0	1	1
23.	1	1	1	1	1	0	1	1	0	0	1	1	1	1	0	1	1
24.	1	1	1	1	1	1	0	1	0	0	1	1	1	1	0	1	1
25.	1	1	1	1	1	1	1	0	0	0	1	1	1	1	0	1	1

Figure 3.3 Example of a set of two-week schedule patterns

words, the penalty cost measures the degree of dissatisfaction with the schedule. Examples of constraints include avoidance of [Warner, Holloway and Grazier, 1984]:

- a single day on

- a single day off
- a stretch of 6 working days
- a stretch of 7 working days
- a stretch of 8 working days
- a stretch of 6 days that includes a 3-day weekend
- a stretch of 7 days that includes a 3-day weekend
- a stretch of 8 days that includes a 3-day weekend

Each of these constraints (undesirable situations) have different penalty costs associated with them. The result from this screening process is stored in a "pattern base" that contains several set of feasible patterns. Feasible patterns are patterns that have total penalty cost less than or equal to maximum penalty cost which is set by the hospital.

3.4 Matching Process.

To form the schedules, work patterns must be matched with the shift patterns. There are two different matching processes:

1. Matching to generate schedules. In this process, work patterns are combined or matched with the shift patterns. Combining these two patterns can be done randomly or using any other method. Pattern base allows the user to develop a large base of

patterns. To illustrate, consider four-week work and shift patterns. If the number of patterns generated during the generation and screening process is, say, 1000 each, then random ordering of these patterns results in a very large set of scheduling patterns. The first set of 1000 randomly ordered work pattern can be combined with the first set of 1000 randomly ordered shift patterns to form 1000 schedule patterns. Then, either work patterns or shift patterns can be randomly reordered to get a new set of 1000 schedule patterns. This process can be repeated multiple times depending on the number of scheduling patterns desired and the acceptability requirements for the patterns.

The schedule patterns can be extended to obtain patterns for longer planning periods. For example, using the four-week patterns, eight-week or twelve-week patterns can be easily generated by simply combining the four-week pattern with itself or any reordered set. This ability makes the system more flexible and powerful.

The schedules generated by this method are saved in the Schedule Base.

2. Nurse assignment, or the process of assigning the schedules to the nurses. The process of schedule

assignment is carried out by matching the schedules from the schedule base with the nurse database. Here again, the schedule base provides flexibility and extendibility due to the large number of schedules that can be mixed and matched. The pattern and schedule bases represent data bases that are designed to provide the scheduler with a large set of alternatives, thus eliminating the need to generate schedules every time a new schedule is needed. Also, these two supporting databases can reduce the screening process resulting in a reduction of computer time and memory requirements. In other words, if the model does not give feasible solution, the scheduler does not need to start from the beginning; they just need to pick another set of patterns from the pattern base.

CHAPTER 4
PATTERN GENERATOR AND PATTERN BASE

The pattern generator produces work and shift patterns that cover all possible schedules with different shifts. It is assumed that the hospital will have a specific operating policy concerning the proportion of nurses working in different shifts. One example is a 3:2:1 policy, meaning that the number of nurses work in the day shift is three times the number of nurses working in the night shift, and the number of nurses work in the evening shift is twice as much as the number of nurses work in the night shift. Other examples of nurses' working policies include 4:1:1, 4:2:1 and 1:1:1. The ratio of number of nurses working in the three shifts and the length of schedule period are management specified parameters. Examples of the length of the schedule are 4-week, 8-week or 12-week; this length represents the time for which the schedule is to be generated.

This chapter first describes the general algorithm for pattern generation; each major components of this algorithm are then explained in detail.

4.1 General Procedure

The general steps in the pattern generation are shown in the flowchart of Figure 4-1.

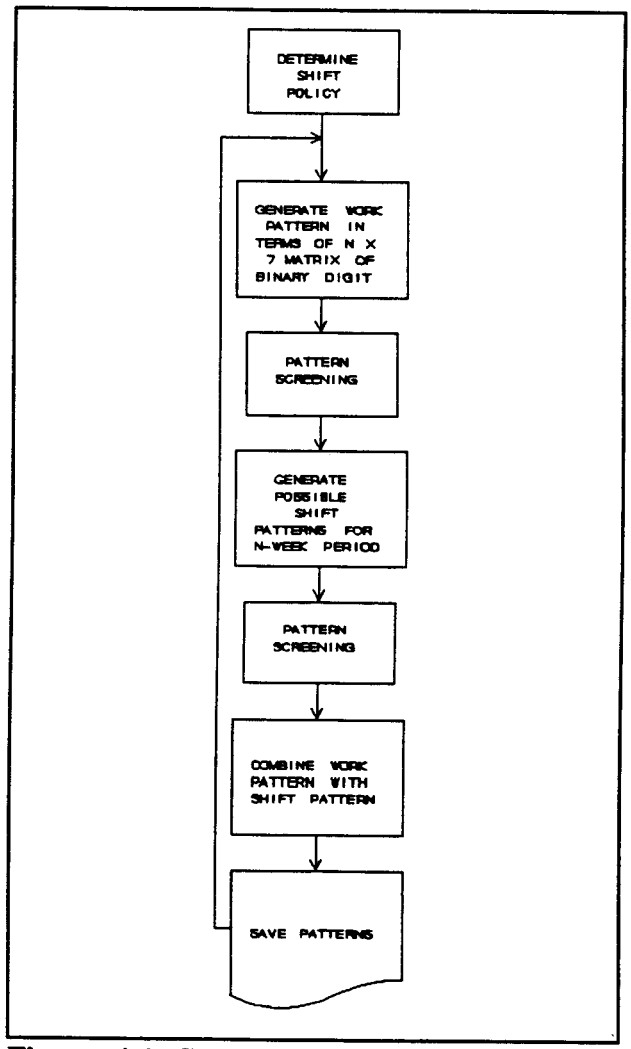

Figure 4-1 General Algorithm for Pattern Generation

The process of pattern generation begins with generating the **work patterns**. A work pattern is a daily pattern that must be followed by a nurse. The pattern shows whether a nurse will work on a particular day or not.

An example of the work pattern schedule of a nurse for a 4-week period is shown in Figure 4-2, where 1 represents the

day when the nurse works and 0 represents the day off.

S	M	U	W	T	F	S	S	M	U	W	T	F	S	S	M	U	W	T	F	S	S	M	U	W	T	F	S
1	1	1	1	1	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-2. An example of a 4-week work pattern

The number of possible work patterns that can be generated is very large. For example, for a 12-week period there will be 2^{84} different patterns. However, some of these patterns are not feasible. The number of work patterns can be significantly reduced if they are screened against constraints before combining them with the shift-patterns. The constraints that will be considered in the screening process include:

1. Minimize working on 6 consecutive days.
2. Minimize working on 7 consecutive days.
3. Avoid working on 8 consecutive days.
4. Minimize single days off and on.
5. A maximum of 4 days off.
6. Minimize working on weekend.
7. Minimize split weekends.

Following the work pattern generation is the process of generating possible **shift-patterns** for a specified hospital policy. A shift-pattern is a pattern showing the shift that a nurse must work in a particular week during a certain period

(the schedule period). For n-week period with k shifts, the number of possible shift-pattern can be calculated as

$$N = \left[n_1 \; n_2 \; \ldots n_k \right]^{n} = \frac{n\,!}{\prod_{i=1}^{k} n_k}$$

For example, for 12-week period with three shifts a day and a 3:2:1 ratio of working policy, the number of possible combinations will be as many as

$$N = \frac{12\,!}{6!\;4!\;2!} = 13860$$

This is also a very large number. Therefore, given a specific hospital policy, the first task is to generate possible shift patterns for that policy followed by randomly selecting some of these shift patterns and combining them with work patterns. In this model, the length of schedule period can be specified as either 4-week, 8-week or 12-week period.

Like work patterns, shift patterns need to be screened against policy constraints. These include:

1. Pattern with 4 consecutive night-shift must be eliminated.

2. Pattern with 4 consecutive evening shift must be eliminated.
3. For a 4-week period, pattern with three night-shifts must be eliminated.
4. For a 4-week period, pattern with three evening shifts must be eliminated.

Example of shift patterns, assuming a scheduling period of 4 weeks and ratio of nurses working in day, evening and night shift to be 2:1:1, are shown in Figure 4-3.

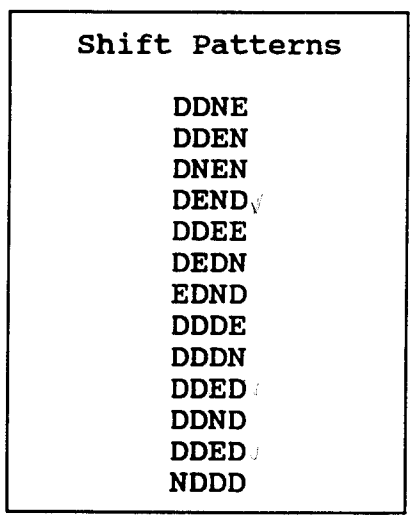

Figure 4-3 Example of Shift Patterns

The combination of the work patterns and the shift patterns is therefore a complete schedule for the hospitals. An example of a complete schedule for one nurse for 4-week period and shift policy of 2:1:1 is shown in Figure 4-4. In this schedule, the nurse works two weeks in the day shift, one

week in the evening shift and one week in the night shift.

Schedule No. : 1								
Day :			M	T	W	T	F	S	S
Week :	1	D:	1	1	1	1	1	0	0
Week :	2	D:	1	1	1	1	1	0	0
Week :	3	N:	0	1	1	1	1	1	0
Week :	4	E:	1	1	1	1	1	0	0

Figure 4-4. An example of a complete schedule of 4-week period with 2:1:1 shift policy for one nurse

4.2 Work Pattern Generation

The first step in pattern generation is the generation of work patterns. This pattern is an array with (7 x n) elements or slots. Each element is symbolized by a **'zero'** or a **'one'** representing day-off and day-on. For 4-week period there will be an array of 28 (7 x 4) slots which will be filled with either zero or one, while for 12-week period the number of slots will be 84 (7 x 12). For a specified planning period, **m** working patterns are required, where **m** is the number of nurses who will be working during this scheduling period.

The procedure for generating work patterns is shown in Figure 4-5. The process starts off by first generating a set of one-week work patterns. The one-week work patterns are then combined with themselves to generate two-week patterns. These two-week patterns are combined with the previous one-week

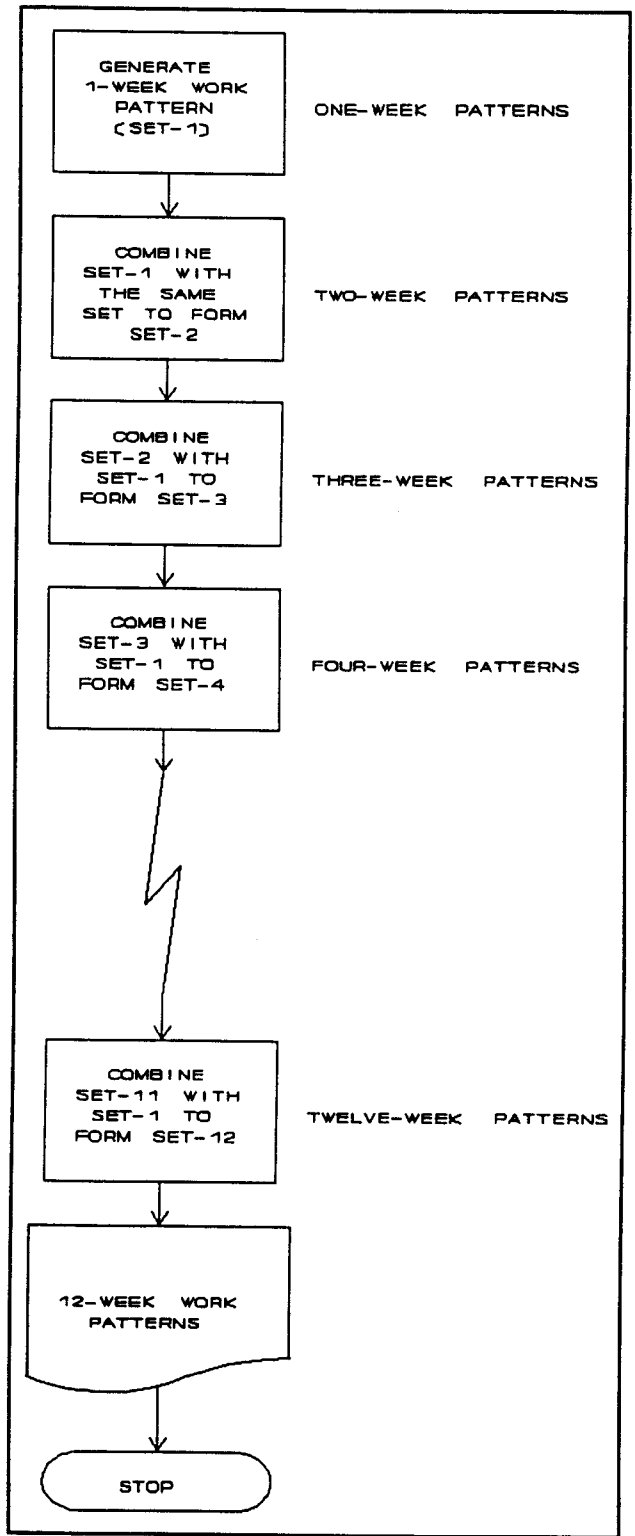

Figure 4-5 General Algorithm of Work Pattern Generation

patterns to create three-week patterns, and so forth. This method, without any screening procedure will create a very large number of patterns. The longer the length of the schedule, the larger the number of possible patterns that can be generated. Even if the system starts with seven one-week patterns, for 4-week period there will be $7^4 = 2401$ different possible patterns and for 12 week there will be more than one billion possible patterns.

Therefore, to make the method more efficient, the pattern generation process is combined with the pattern screening process using **the best-first heuristic technique**. Only patterns that pass the screening process are stored in the database to be combined with the shift patterns.

In the **best-first search algorithm** application used in this research, after a set of one-week work patterns have been generated, a penalty cost is calculated for every pattern in the set. Penalty cost is a cost associated with the violation of constraints. The more the constraints violated, the higher the penalty cost. The penalty costs associated with constraints are different from each other, reflecting the relative importance of constraints to hospital administration and/or nurses.

After calculating the penalty cost, the program sorts the patterns in ascending order of penalty cost; thus the pattern at the top of the queue is the minimum cost pattern. This pattern then forms the basis for generating two-week patterns.

The method consists of combining the minimum cost pattern with the previous set of one-week patterns resulting in a set of two week patterns. After the two-week patterns have been generated, the penalty costs associated with these patterns are calculated. The two-week patterns are again sorted based on the least penalty cost.

The process continues until patterns with the desired length of period have been generated. At this stage, a subset of patterns whose penalty costs are less than or equal to the maximum penalty cost is selected as a feasible set. The maximum penalty cost is set by the administrator as a function of objectives and constraints. The number of patterns in the feasible set have to exceed the number of nurses (since one work pattern is required for each nurse or nurse group). If the number of feasible patterns is less than the number needed by the hospital, the process backtracks to the previous step to pick the pattern with the second minimum penalty cost. The process of generating additional patterns is the same, that is, by combining that pattern with the previous set of the one-week patterns. If even after this stage not enough feasible patterns have been generated, the process continues by backtracking to the previous stage to pick the third minimum penalty cost pattern. This iterative process continues until all available patterns in the previous stage are exhausted. If still more patterns are required the process then backtracks one more level. This process continues until

enough pattern have been generated or no more feasible patterns can be generated. Figure 4-6 shows the flowchart for the best first algorithm for a four-week period.

4.2.1 Evaluation Function

The evaluation function in the best-first search procedure is used to direct the search process. Here, the evaluation function is the penalty cost which depends on the adverse effect of a constraints on nurse's or hospital's objectives.

Before the process of pattern generating begins, all constraints from hospital and nurses must be listed and associated penalty costs specified. It is not possible to provide an exhaustive set of these constraints. As mentioned before, every environment has its own policies and objectives. A representative set of constraints are outlined below; the design of the DSS is such that the system can be modified to fit a specific environment.

To minimize cost, the hospital objective is to minimize the number of nurses working in a particular shift on a particular day. However, it is also important to the hospital to have adequate number of nurses working in every shift to cover the patient's needs. The constraints usually imposed by the hospital are:

1. Minimum number of nurses work in every shift.

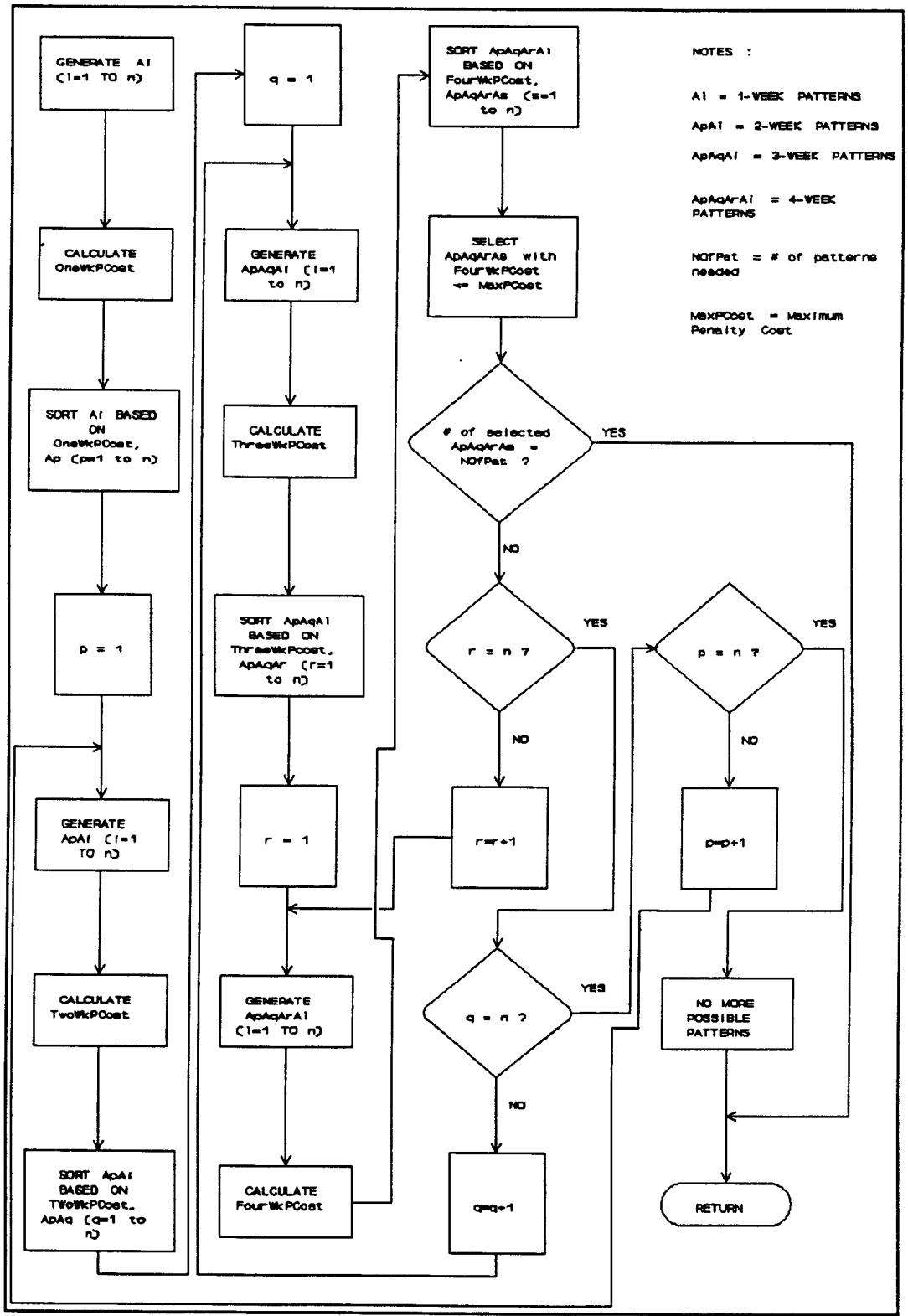

Figure 4-6 Best-First Search Algorithm for Pattern Generation (4-week period)

2. Minimum number of nurses work during weekend.
3. The number of nurses work in every shift must meet the staff requirements (assumed to be already determined).
4. Every nurse must work 5 days a week, or 40 hours a week during the day shift.
5. The maximum number of consecutive work days to be six.
6. The maximum number of consecutive days off to be four, including weekends.
7. Every nurse to work in evening shifts at least 1/3 of the time and in night shifts at least 1/6 of the time (depending on the policy of the hospital and the prediction of the ratio of number of nurses required to work every shift. Usually the number of nurses needed for evening and night shift is lower than the day shift, and the Saturday and Sunday requirements are 20 to 30 percent lower than week-day requirement [Ozkarahan, 1987]).
8. All absenteeism caused by sickness and any other reason must be covered, but the number of floating nurse must be minimized.

Nurses also have some requirements, preferences and limitations. Some constraints are imposed by the federal and state laws and labor organizations; others are imposed by physical limitations. Typical constraints for nurses are:

1. Every nurse must have a chance to have weekend off (in some cases every other weekend off or 3 weekends off every month).
2. Every nurse must have at least two consecutive days off before shift change in order to let the nurses to adjust to the new schedule.
3. The maximum number of days that a nurse can work consecutively be 6 days.
4. Single day on or off must be minimized.
5. Every nurse must be allowed to have some preferred days off (birthday, anniversary, etc.).
6. Every nurse must be allowed not to work during holidays (christmas, thanksgiving, etc.), if so desired.
7. The number of weekend working days for every nurse be the same.
8. Every nurse have the same number of evening and night shifts.

Each of the constraint is assigned a penalty cost, representing the loss to the hospital or the nurses if that constraint is violated. An example set of penalty costs is shown in Table 4-1.

CONSTRAINTS	Penalty Cost
1. Minimize # of 4 zero in sequence	10
2. Minimize # of 3 zero in sequence	5
3. Minimize # of 6 one in sequence	10
4. Minimize # of 7 one in sequence	30
5. Minimize # of 8+ one in sequence	40
6. Minimize # of on/off pattern	10
7. Minimize # of non-weekend off :	
a. Fri-Sat combination	5
b. Sun-Mon combination	5
c. Mon-Tue, Tue-Wed, Wed-Thu, Thu-Fri	10
8. Minimize single zero	10
9. Minimize single zero	10
10. Minimize working on weekend	10

Table 4-1. Example of Penalty Cost

4.3 Shift Pattern Generation

In generating shift patterns, it is assumed here that nurses work the same shift each week; shift change only occurs at the beginning of the week and remains the same throughout that week. The number of shifts in each shift pattern equals the schedule length.

The generation of shift patterns starts off by generating D, E and N patterns based on shift policy chosen by the user. For example, for 1000 schedules, and 4-week period, 4000 shift patterns will be created. If the shift ratio is 4:3:1, for

example, then 2000 D's will be generated together with 1500 E's and 500 N's. A shift pattern base is created to save these patterns. Then, these patterns will be reordered randomly, and the new order set will be saved into a different pattern base. Following this step, 4-week shift patterns are generated by random selection of four shift patterns to form a 4-week shift pattern. Then, screening process is carried out to eliminate infeasible patterns. The schedule base is generated by picking patterns from this set repeatedly until the number of patterns in the schedule base is the same as the number of required work patterns.

4.4 Combining Work and Shift Patterns

To form a complete schedule, the work patterns are combined with shift patterns. The combination process is essentially a matching process. The process, shown in Figure 4-7, consists of:

1. Pick one work pattern from pattern base.
2. Pick one shift pattern from pattern base.
3. Assign the first character (D, E or N) of the shift pattern to the first seven digits of the work pattern; this represents the work assignment for the first week. The second character of the shift pattern is then assigned to the second seven digit set in the work pattern for second week assignment. The process continues

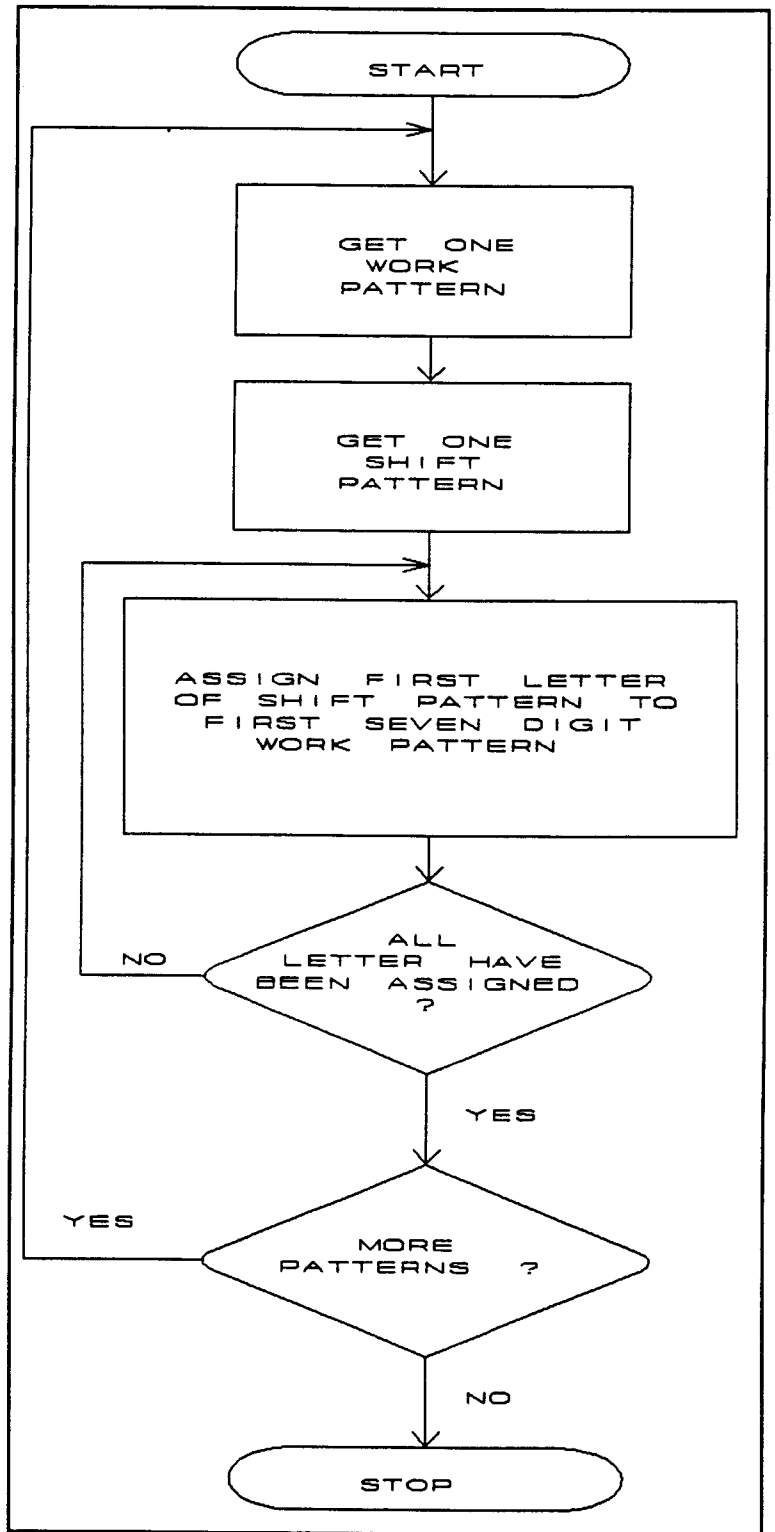

Figure 4-7 Flowchart of Matching Work and Shift Patterns

until a seven week set from the work pattern is assigned to each character (shift) of the shift pattern. This completes the work schedule for one nurse.

4. The assignment process is repeated until at least one work schedule is generated (by combining a unique work pattern and a unique shift pattern) for each nurse.

To illustrate this process, consider generating schedules for a four-week scheduling period. Table 4.2 shows seven work patterns and associated penalty costs computed based on the values given in Table 4-1. These seven work patterns are

1111100111110011111001111100	PCost = 0
1111100111110011111001111001	PCost = 5
1111100111110011111000111110	PCost = 10
1111100111110011111001110011	PCost = 10
1111100111110011111001100111	PCost = 10
1111100111110011111001110110	PCost = 15
1111100111110011111001101110	PCost = 15

Table 4-2 Example of Some Work Patterns with Penalty Cost

combined with seven shift patterns. Table 4.3 shows some randomly generated shift patterns from where the first seven patterns have been selected to be combined with work patterns (Table 4.4). The procedure for accomplishing this is shown in Figure 4-8, and work schedules resulting from this combination are shown in Figure 4.9.

Shift Patterns
DDNE
DDEN
DNEN
DEND
DDEE
DEDN
EDND
DDDE
DDDN
DDED
DDND
DDED
NDDD

Table 4-3 Example of Shift Patterns

Shift Pattern	Work Pattern
DDNE	11111001111100111111001111100
DDEN	11111001111100111111001111001
DNEN	11111001111100111111000111110
DEND	11111001111100111111001110011
DDEE	11111001111100111111001100111
DEDN	11111001111100111111001110110
EDND	11111001111100111111001101110

Table 4-4. Combination of Shift and Work Patterns

```
Pattern No:     1 : 1111100111110011111001111100
Shift             : D       D       N       E
Pattern No:     2 : 1111100111110011111001111001
Shift             : D       D       E       N
Pattern No:     3 : 1111100111110011111000111110
Shift             : D       N       E       N
Pattern No:     4 : 1111100111110011111001110011
Shift             : D       E       N       D
Pattern No:     5 : 1111100111110011111001100111
Shift             : D       D       E       E
Pattern No:     6 : 1111100111110011111001110110
Shift             : D       E       D       N
Pattern No:     7 : 1111100111110011111001101110
Shift             : E       D       N       D
```

Figure 4-8 Matching Shift and Work Patterns

```
Schedule No. :    1
------------------------------
Day  :            M T W T F S S
------------------------------
Week :   1   D:   1 1 1 1 1 0 0
Week :   2   D:   1 1 1 1 1 0 0
Week :   3   N:   0 1 1 1 1 1 0
Week :   4   E:   1 1 1 1 1 0 0
==============================

Schedule No. :    2
------------------------------
Day  :            M T W T F S S
------------------------------
Week :   1   D:   1 1 1 1 1 0 0
Week :   2   D:   1 1 1 1 1 0 0
Week :   3   E:   1 1 1 1 0 0 1
Week :   4   N:   1 0 0 1 1 1 1
==============================

Schedule No. :    3
------------------------------
Day  :            M T W T F S S
------------------------------
Week :   1   D:   1 1 1 1 1 0 0
Week :   2   N:   1 1 1 1 1 0 0
Week :   3   E:   1 1 0 0 1 1 1
Week :   4   N:   1 1 1 0 1 1 0
==============================

Schedule No. :    4
------------------------------
Day  :            M T W T F S S
------------------------------
Week :   1   D:   1 1 1 1 1 0 0
Week :   2   E:   1 1 1 1 1 0 0
Week :   3   N:   1 1 1 0 0 1 1
Week :   4   D:   0 1 1 1 1 1 0
==============================
```

```
Schedule No. :    5
------------------------------
Day  :            M T W T F S S
------------------------------
Week :   1   D:   1 1 1 1 1 0 0
Week :   2   D:   1 1 1 1 1 0 0
Week :   3   E:   1 1 1 0 0 1 1
Week :   4   E:   1 1 1 0 1 1 0
==============================

Schedule No. :    6
------------------------------
Day  :            M T W T F S S
------------------------------
Week :   1   D:   1 1 1 1 1 0 0
Week :   2   E:   1 1 1 1 1 0 0
Week :   3   D:   1 1 1 1 0 0 1
Week :   4   N:   1 0 1 1 1 0 1
==============================

Schedule No. :    7
------------------------------
Day  :            M T W T F S S
------------------------------
Week :   1   E:   1 1 1 1 1 0 0
Week :   2   D:   1 1 1 1 1 0 0
Week :   3   N:   1 1 0 0 1 1 1
Week :   4   D:   1 0 0 1 1 1 1
==============================
```

Figure 4-9 Example of Nurse Schedules

4.5 Generating Reports

Once a feasible set of work patterns have been identified, the system can be used to print the work schedule identified. This report could be used by the administrator to evaluate whether the schedule generated meets the staff requirement of the hospital or not. If the requirements are not satisfied, the administrator can try another schedule by simply picking another set of work and shift patterns from the pattern base and forming the new schedule. This process can be repeated until a desirable set of work schedules is obtained. The reports generated by the system consist of individual work schedules (Figure 4.9) and staff summary report (Figure 4.10) which shows the number of nurses that will be working in each shift on each day of the scheduling period and the deviations from the ideal shift assignment. The actual ratio in Figure 4.10 is based on the generated schedules; the ideal ratio is computed based on the specified shift policy.

STAFF LIST REPORT:

Day :		Day-Shift	Evening-Shift	Night-Shift
Monday	1	351	256	235
Tuesday	2	406	298	276
Wednesday	3	399	295	264
Thursday	4	378	279	248
Friday	5	300	218	208
Saturday	6	104	76	78
Sunday	7	142	98	91
Monday	8	350	237	215
Tuesday	9	418	274	238
Wednesday	10	379	244	206
Thursday	11	332	227	194
Friday	12	288	220	168
Saturday	13	219	164	126
Sunday	14	229	144	128
Monday	15	328	225	209
Tuesday	16	362	258	232
Wednesday	17	340	234	219
Thursday	18	329	229	192
Friday	19	295	223	191
Saturday	20	257	184	164
Sunday	21	234	152	143
Monday	22	311	194	168
Tuesday	23	330	232	186
Wednesday	24	314	245	186
Thursday	25	337	227	194
Friday	26	328	234	209
Saturday	27	295	238	191
Sunday	28	250	180	151
Total		8605	6085	5310
Average		307.321	217.321	189.643
Ratio Actual		0.430	0.304	0.265
Ratio Ideal		0.445	0.333	0.222

Figure 4-10 Staff Requirement Report

CHAPTER 5
IMPLEMENTATION AND SYSTEM EVALUATION

5.1 Implementation

The nurse scheduling system was developed in Pascal using Borland's Turbo Pascal Professional Version 6.0 compiler [Borland International, 1990]. The system was implemented on an Intel 80386-based IBM Compatible Personal Computer and an Intel 80286-based IBM PS/2 System 50-Z (microchannel). Pascal was selected as the language for the program because it is a modular and very structured language; it is easy to modify, extend and expand the program.

The program basically consists of three main parts, the user interface, the pattern generator, and the schedule generator and analyzer. The main program models the user interface. The procedures and units represent the pattern generation (work and shift patterns) including the search algorithm, and the schedule generator. The results of the program are stored in pattern and schedule bases. The program use and input screens along with extension and modification guide are described in Appendix A. The complete computer code for the program is provided in Appendix B, C, D and E.

5.1.1 User Interface

The user interface section interacts with the user to

obtain specification of the following parameter (Figure 5.1):

1. Length of the scheduling period (4-week, 8-week or 12-week).
2. Penalty cost values for constraints. Some default values are built in the program based on some references and discussion with nurse manager at Good Samaritan hospital in Corvallis, Oregon. These values are displayed along with each constraint. The user has the choice of specifying new values or running the program with the default values provided.
3. Number of patterns to be generated. For flexibility, it is suggested that this number be greater than that of nurses.
4. Maximum Penalty cost. The maximum penalty cost is the total penalty cost that can be tolerated. This value must be decided by the administrator. The guide for selection is simple: select the worst constraint or scenario that still can be tolerated and set the total penalty cost of that constraint or scenario as the maximum penalty cost.
5. Shift Policy Ratio. There are nine shift policy ratios built in the program. These policies are the most common policies selected from references and discussion with the nurse manager at Good Samaritan hospital in Corvallis, Oregon. However, the program also allows the user to specify different policies, if desired.

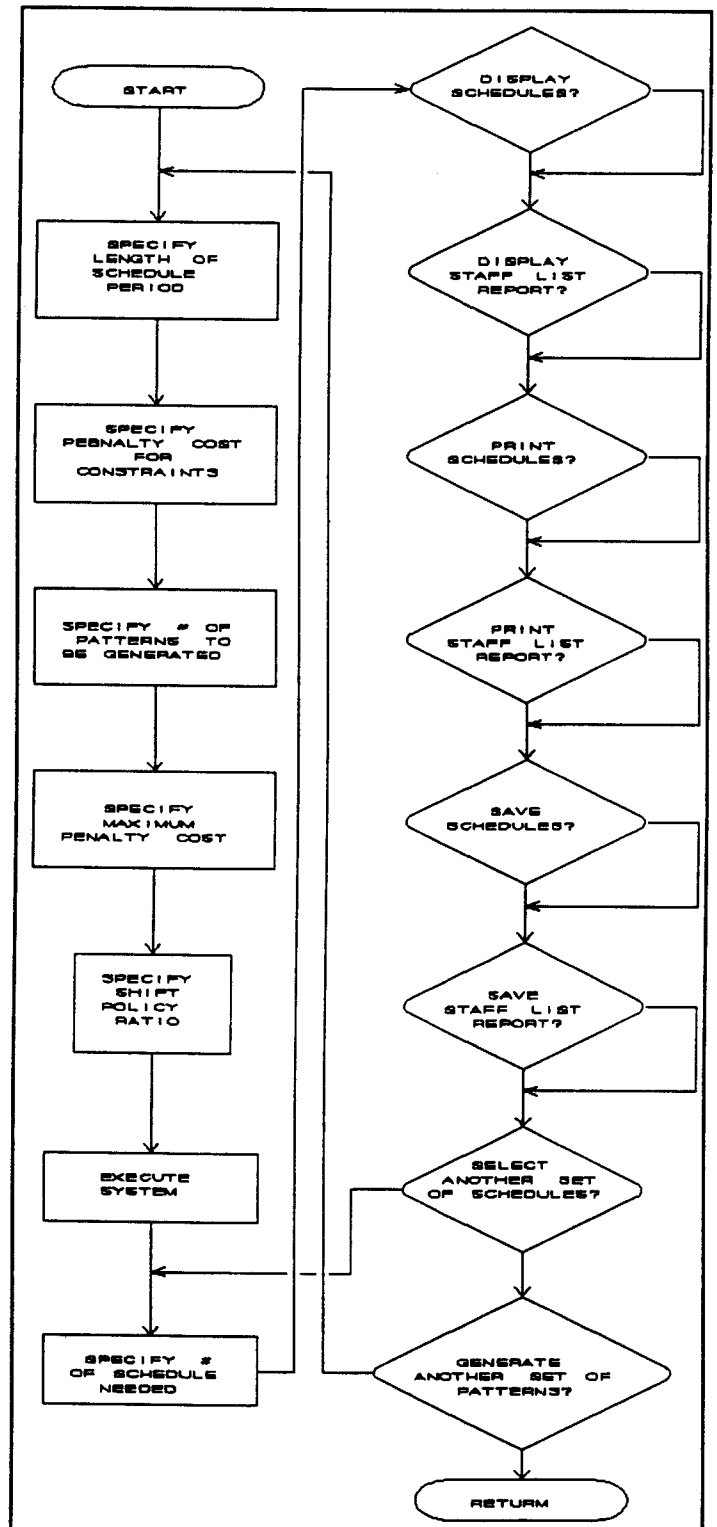

Figure 5.1 Flowchart for user interface

6. Number of schedules to be selected from the schedule base. After the work and shift patterns have been generated, the user may select any number of these schedules, the maximum being the number of patterns generated. If this set of schedules does not satisfy the user need, then he can select a different set without having to generate another set of patterns. Generating another set is also allowed, if the current set of patterns does not satisfy the user.

The interface section also allows the user to display the schedules and staff list report on the monitor before saving or printing the schedules. This allows the user to evaluate the schedules before using them.

5.1.2 Pattern Generation

The pattern generator creates work and shift patterns and screens them against constraints. The algorithm of generating and screening process has been described in the previous chapter. However, certain implementation consideration will be discussed here.

The problems encountered in the implementation of pure best-first search algorithm are associated with the limitations of the microcomputer. In the best-first search algorithm, as the length of scheduling period increases, the

number of patterns to be generated also increases. This means that the screening process will require more computer memory and time. For example, starting with 21 one-week patterns, a 12 week scheduling period requires 21^{12} patterns at the lowest level of the search tree. Bringing these patterns to computer memory is a big problem, because even if the variable is defined as a "character" variable, it will take at least 1 byte for every pattern, and this means that the entire set of patterns will require at least 21^{12} bytes (or more than 7 million gigabytes of memory). A computer with DOS operating system can only have 640 kb conventional memory and even with DOS extender program, the highest capacity of personal computer today is 32 megabyte.

If swapping data from memory to disk is considered, the problem is the tracking of data. The penalty cost calculation and the sorting process needs all data to be accessed. Hence it is not possible to implement a swapping procedure for this size of data because the processing time will be too long.

The second problem encountered in the search method was the problem of schedule variation. The term schedule variation means that schedules should have variations among each other for every week segment otherwise nurses will end up with similar schedules. This problem occurs because of the nature of the search method. The search method starts with the best first week pattern, then continues to the best first and second week combination, and so on for the scheduling period.

The process of backtracking starts off from the last week backwards. Thus, if the number of patterns needed are fulfilled by backtracking to the second week, for example, then all schedules will have the same pattern for the first week.

This problem must be avoided to provide variability in schedules. In this research, increased variability is introduced through:

1. Limiting the length of schedule period in the best-first search. This also reduces computer processing time. In addition, given a specific scheduling period, greater the number of patterns generated, higher the variability.
2. Using a random schedule selection process so that the schedule variation will be higher.

After several trials, it was found that the length of four-week schedule period is the best choice for starting point of the algorithm implementation. For example, 1000 patterns generated with a maximum penalty cost of 40 have all four week variations. Therefore, the base length of schedule period is chosen to be four weeks. For longer scheduling periods, schedules are formed by randomly combining the four-week patterns.

There are two constraints that are generally considered important in most hospitals. These are the requirement that every nurse must work five days a week and that the maximum number of consecutive days off for a nurse be four. These

constraints can be used to start off the pattern generation process. Thus the first week patterns will consist of five 1's and two 0's. This results in 21 different one-week patterns each with five 1's and two 0's meaning that every nurse will work five days a week. This also will guarantee that the maximum consecutive days-off that a nurse can have will be four days. The best-first algorithm uses these 21 one-week patterns as the starting point. The starting pattern base can be expanded by adding more combinations of one-week patterns and relaxing the above two constraints.

The algorithm for generating the four-week scheduling patterns is shown in Figure 5.2. This subprogram starts off by implementing the best-first search algorithm for generating four-week work patterns. The program then re-arranges the work patterns randomly (and save this new arrangement to a different file). For eight-week or twelve-week scheduling period, the four-week work patterns are combined accordingly.

The program also generates shift patterns according to the hospital shift policy. The process of generating shift patterns starts off by generating the number of D's, E's and N's based on the hospital policy ratio. These patterns are then rearranged randomly and screen against some constraints. The primary constraints used in the screening process are:

1. For four-week scheduling period, pattern that has all evening and night shifts is eliminated.
2. For four-week scheduling period, pattern that has three

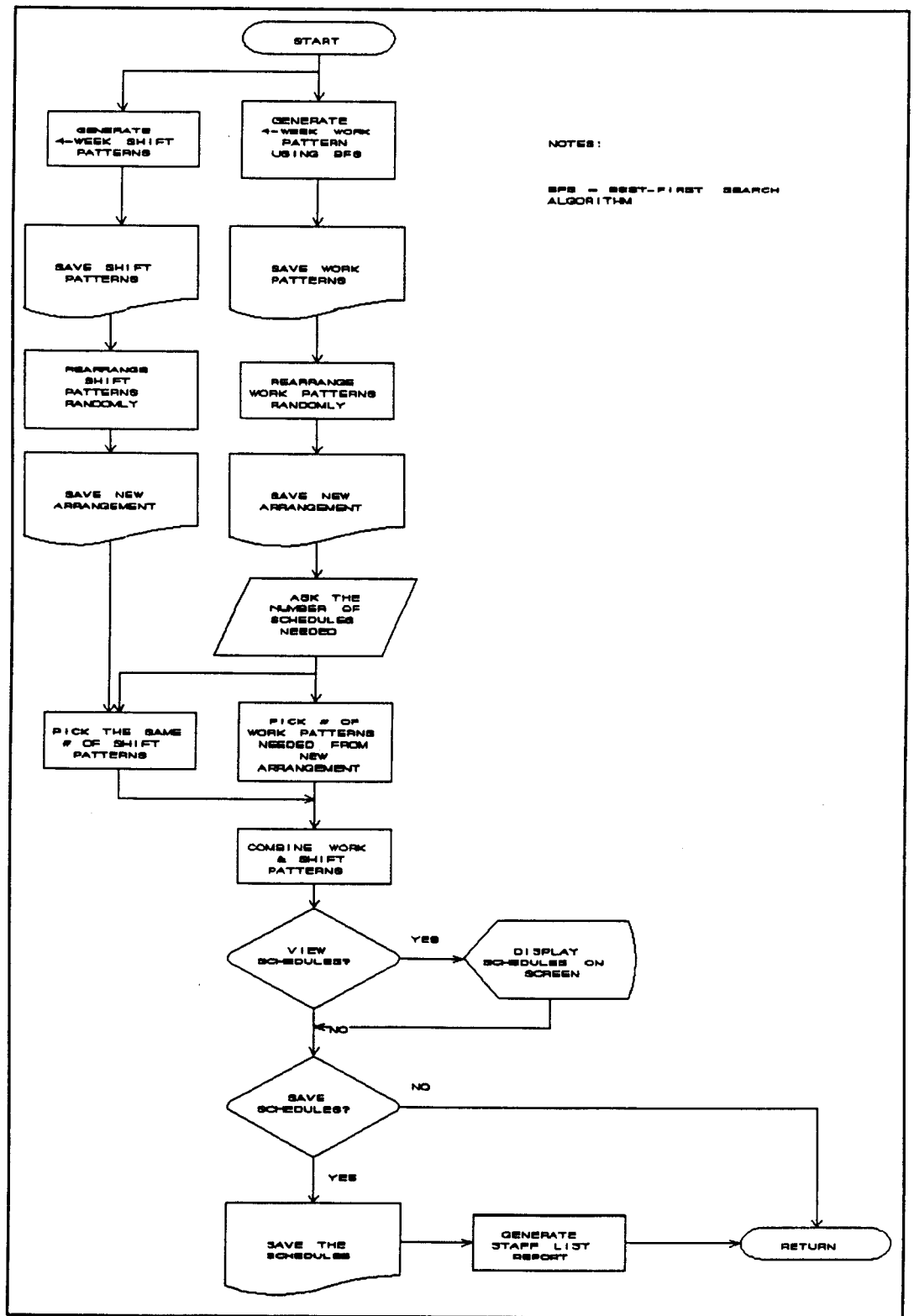

Figure 5.2 Algorithm for generating 4-week schedules

evening or night shifts is eliminated.

For scheduling period of eight or twelve weeks, the four-week shift patterns were also combined to form patterns of required length. Based on user specification of number of schedules desired, the program combines the work and shift schedules to form the required number of schedules. As mentioned previously, the 8- and 12-week schedules are based on the 4-week patterns. For example, to form 8-week scheduling period, a set of 4-week patterns are first generated. Two different random order of these patterns are then saved in different files. Patterns from these files are combined to form 8-week patterns. The same procedure is also carried out for shift patterns, and the work and shift patterns then are combined to form the required schedules.

5.1.3 Penalty Cost Calculation

An important component of the best-first search algorithm is the penalty cost calculation procedure. The penalty cost is used as a heuristic function for the search method. The penalty cost calculation is carried out by reading the patterns one by one and locating the position of the character to check if a certain condition is met or not. For example, to calculate penalty cost of working in weekend for one-week pattern, the program checks if digits 6 and 7 of the pattern are both "1". If they are, then the pattern is assigned a

specific penalty cost. If digit 6 is "1" and digit 7 is "0", a different penalty cost is assigned. Similarly if 6 is "0" and 7 is "1" then a third penalty cost is assigned (which may or may not be the same as the penalty cost for "1" and "0"). The pseudo code for this process is shown in Figure 5.3

```
procedure pcostweekend;
   for i := 1 to NumberOfPatterns do
     begin
       if digit 6 and digit 7 equal '1' then
          penalty cost := WeekendPenaltyCost;
       else
       if digit 6 equal '1' and digit 7 equal '0' then
          penalty cost := PenaltyCostOfSaturdayOnSundayOff;
       else
        if digit 6 equal '0' and digit 7 equal '1' then
          penaltycost := PenaltyCostOfSaturdayOffSundayOn;
       end if
      end;
end;
```

Figure 5.3 Pseudocode for penalty cost calculation for weekend.

A different procedure is required if the constraint is to check for a single day on or off in the pattern. This checking requires comparing a digit with its following and previous digits. If both the following and previous digits are different from the digit being checked, then it implies single-on or single-off day (depending on the digit character).

The procedure for checking two-week, three-week and four week patterns are slightly different from the one-week patterns. In these patterns, more works must be carried out. For example, weekend constraints for two-week patterns must

check for digits 6 and 7, and digits 13 and 14. For three-week patterns, checking for weekend's constraints includes digits 6, 7, 13, 14, 20 and 21; and for four-week patterns, this includes digits 6,7,13,14,20,21, 27 and 28.

Similarly, checking single day on or off in multiple week patterns is also different from one-week patterns. For one-week patterns, checking starts from digit 2 and goes up to digit 6, while for two-week patterns it starts from digit 2 up to digit 13, for three-week patterns the checking begins from digit 2 and ends at digit 20, and for the four-week patterns, this process starts from digit 2 up until digit 27. A slightly different matching process is also required to calculate penalty cost for working on 6, 7 or 8 consecutive days, and for having 3 or 4 consecutive days off.

The sum of penalty costs associated with each constraint represents the total penalty cost for the pattern. This value is used as the heuristic function for the search method.

5.1.4 Sorting the patterns

An important aspect of the best-first search technique is the sorting of the patterns based on the penalty cost associated with each pattern. In this program, the sorting algorithm used is "the straight insertion technique" [Rugg and Feldman, 1989], or simply referred to as "the insertion technique" [Sedgewick, 1988]. This method is very simple but flexible

enough to accommodate big programs. The pseudocode for this method is shown in Figure 5.4 [Sedgewick, 1988].

```
procedure insertion;
     var i,j,v : integer;
     begin
     for i := 2 to N do
          begin
          v := a[i]; j := i;
          while a[j-1] > v do
               begin a[j] := a[j-1] ; j := j - 1 end;
          a[j] := v
          end
     end;
```

Figure 5.4 Pseudocode for insertion technique

5.2 System Evaluation

Two approaches were used to evaluate the computer model:

1. Evaluation by one of the nurse scheduling managers at Good Samaritan Hospital, Corvallis, Oregon. Direct utilization of the program at Good Samaritan Hospital was not possible due to the nature of its operations. The hospital has a relatively small full-time nurse force and a large part-time pool of nurses. It thus focuses on allocation (currently using a data base system) rather than work schedules generation and assignment. However, the expertise of the scheduling manager was used to:

   (a) Evaluate the constraints included in the model and

the penalty costs. Even though the rating system used at Good Samaritan Hospital is different, the penalty cost values initially specified in the program (based on studies reported in literature) were close to their values. The set of constraints and the associated penalty values currently in the program have been revised after discussions and feedback from this scheduling manager.

(b) Evaluate the program features and output reports. A series of meetings were arranged during the project duration. The suggestions provided by the manager in the initial meetings were incorporated in the program. After the system development was completed, the manager reviewed the output. Feedback on the utility of the program was very positive.

2. The results for statistical evaluation of the program under different combinations of length of schedule and shift policy ratio are shown in Tables 5.1 through 5.3. The tables show the mean, standard deviation, minimum and maximum of the actual ratio values of the nurse staffing coverage. The ideal ratios of the staffing coverage or the shift policy ratios are also included in the tables.

The tables show that the actual ratios are close to the shift policy ratio specified by the hospital. The tables also show that the minimum and maximum values of the total deviation, that is, the sum of understaffing

Length of Schedule: 4 weeks.
Number of runs : 50; Maximum Penalty Cost : 40
Patterns generated : 1000
Schedules selected every run : 1000

Shift Policy Ratio 3:2:1

	Mean	StdDev	Min	Max	Ideal
Day	0.47746	0.007281	0.465	0.493	0.50000
Evening	0.30480	0.005821	0.287	0.318	0.33333
Night	0.21776	0.005369	0.207	0.229	0.16667
Total Deviation			0.080	0.125	

Shift Policy Ratio 2:1:1

	Mean	StdDev	Min	Max	Ideal
Day	0.47282	0.006405	0.455	0.486	0.50000
Evening	0.24896	0.005337	0.238	0.262	0.25000
Night	0.27822	0.005920	0.264	0.294	0.25000
Total Deviation			0.032	0.090	

Shift Policy Ratio 4:3:1

	Mean	StdDev	Min	Max	Ideal
Day	0.49692	0.009239	0.475	0.513	0.50000
Evening	0.32796	0.005437	0.319	0.343	0.37500
Night	0.17520	0.006280	0.162	0.194	0.12500
Total Deviation			0.073	0.138	

Shift Policy Ratio 3:2:2

	Mean	StdDev	Min	Max	Ideal
Day	0.42586	0.008285	0.404	0.443	0.42900
Evening	0.27010	0.006534	0.258	0.289	0.28600
Night	0.30414	0.006824	0.290	0.317	0.28500
Total Deviation			0.021	0.065	

Shift Policy Ratio 4:3:2

	Mean	StdDev	Min	Max	Ideal
Day	0.42148	0.007094	0.407	0.436	0.44500
Evening	0.30420	0.004919	0.294	0.314	0.33333
Night	0.27456	0.005193	0.266	0.287	0.22222
Total Deviation			0.087	0.130	

Table 5.1 Statistics for 4-week schedules

Length of Schedule: 8 weeks.
Number of runs : 50; Maximum Penalty Cost : 40
Patterns generated : 1000
Schedules selected every run : 1000

Shift Policy Ratio 3:2:1

	Mean	StdDev	Min	Max	Ideal
Day	0.48232	0.006984	0.470	0.501	0.50000
Evening	0.30220	0.006542	0.288	0.314	0.33333
Night	0.21548	0.005360	0.205	0.233	0.16667
Total Deviation			0.076	0.132	

Shift Policy Ratio 2:1:1

	Mean	StdDev	Min	Max	Ideal
Day	0.47920	0.008219	0.452	0.497	0.50000
Evening	0.25060	0.006406	0.236	0.269	0.25000
Night	0.27036	0.005959	0.260	0.288	0.25000
Total Deviation			0.022	0.096	

Shift Policy Ratio 4:3:1

	Mean	StdDev	Min	Max	Ideal
Day	0.49676	0.008123	0.480	0.515	0.50000
Evening	0.32714	0.005967	0.310	0.338	0.37500
Night	0.17614	0.006597	0.163	0.191	0.12500
Total Deviation			0.082	0.132	

Shift Policy Ratio 3:2:2

	Mean	StdDev	Min	Max	Ideal
Day	0.41680	0.006309	0.400	0.436	0.42900
Evening	0.27292	0.005336	0.263	0.287	0.28600
Night	0.31034	0.005884	0.301	0.325	0.28500
Total Deviation			0.032	0.080	

Shift Policy Ratio 4:3:2

	Mean	StdDev	Min	Max	Ideal
Day	0.43152	0.007870	0.415	0.448	0.44500
Evening	0.29882	0.006443	0.284	0.315	0.33333
Night	0.26980	0.006521	0.255	0.283	0.22222
Total Deviation			0.065	0.122	

Table 5.2 Statistics for 8-week schedules

Length of Schedule: 12 weeks.
Number of runs : 50; Maximum Penalty Cost : 40
Patterns generated : 1000
Schedules selected every run : 1000

Shift Policy Ratio 3:2:1

	Mean	StdDev	Min	Max	Ideal
Day	0.47392	0.010336	0.451	0.500	0.50000
Evening	0.30536	0.007318	0.292	0.321	0.33333
Night	0.22076	0.008014	0.202	0.241	0.16667
Total Deviation			0.070	0.148	

Shift Policy Ratio 2:1:1

	Mean	StdDev	Min	Max	Ideal
Day	0.47660	0.009031	0.455	0.494	0.50000
Evening	0.25006	0.006857	0.233	0.265	0.25000
Night	0.27316	0.007569	0.258	0.297	0.25000
Total Deviation			0.022	0.095	

Shift Policy Ratio 4:3:1

	Mean	StdDev	Min	Max	Ideal
Day	0.49306	0.008712	0.471	0.519	0.50000
Evening	0.32724	0.007254	0.313	0.342	0.37500
Night	0.17974	0.007451	0.157	0.196	0.12500
Total Deviation			0.082	0.142	

Shift Policy Ratio 3:2:2

	Mean	StdDev	Min	Max	Ideal
Day	0.41400	0.008065	0.395	0.428	0.42900
Evening	0.28084	0.006842	0.268	0.295	0.28600
Night	0.30516	0.007273	0.285	0.322	0.28500
Total Deviation			0.007	0.074	

Shift Policy Ratio 4:3:2

	Mean	StdDev	Min	Max	Ideal
Day	0.42884	0.009091	0.411	0.447	0.44500
Evening	0.30392	0.007180	0.290	0.322	0.33333
Night	0.26732	0.008620	0.239	0.292	0.22222
Total Deviation			0.034	0.140	

Table 5.3 Statistics for 12-week schedules

(negative deviation) and overstaffing (positive deviation) in terms of shift ratio for all scenarios is relatively small.

$$TotalDeviation = abs[actual\ ratio\text{-}ideal\ ratio]_{day\text{-}shift} + abs[actual\ ratio\text{-}ideal\ ratio]_{eveningshift} + abs[actual\ ratio\text{-}ideal\ ratio]_{nightshift}$$

The statistics on Tables 5.1, 5.2 and 5.3 were collected by running the program for 15 scenarios (three different length of schedules and five different shift policies). For every scenario, 1000 patterns were generated, and from these patterns every time 1000 schedules were picked (drawing with replacement). The number of runs for every scenario is 50. This procedure was used because of the following considerations.

1. After the patterns (both work and shift patterns) are generated, they are reordered randomly to avoid bias during the selection of the patterns. This bias may occur because of the nature of the search method that is used in the process of screening work patterns which guarantee that the first pattern generated will be the best, the second one will be the second best, and so on.
2. The more the patterns generated (up to a certain limit), the better the variation of the patterns, and, the better the coverage. But, for the nurse, the less the patterns generated, the better the patterns. Therefore,

in the choice of the number of patterns generated (m) there is a trade-off between penalty cost and coverage. For large m the hospital schedule is laikely to be closer to the ideal coverage but large m also allows individual schedules with higher penalty costs and takes more computing time.

Therefore, the evaluation was carried out by selecting n schedules from the m patterns at random with replacement. Each run of the program produces a list that can be regarded as independent of the lists produced by other runs so that it is possible to claim that the universe of possible runs includes all feasible lists.

Table 5.4 shows the results of generating 1000 (m) patterns, and then with 50 runs selecting 300 (n) schedules at random with replacement.

Length of Schedule	: 4-week				
Shift Policy Ratio	: 4:3:1				
Number of runs	: 50				
Patterns generated	: 1000				
Schedules selected at each run	: 300				
Maximum Penalty Cost	: 40				
	Mean	StdDev	Range Min	Range Max	Ideal Ratio
Day	0.48986	0.014479	0.455	0.525	0.50000
Evening	0.32776	0.010422	0.298	0.351	0.37500
Night	0.18232	0.009833	0.163	0.205	0.12500
Total Deviation			0.080	0.160	

Table 5.4 Statistics for m=1000 n=300, 4-week schedule

From the table we can see that the total deviation is 0.080. If the number of schedules selected are 1000, this number is 0.073. These numbers are relatively small, and all of those schedules have penalty cost less than or equal to maximum penalty cost. The probability that this best schedule set (the best set from 50 runs) contains the top 10 % of the feasible schedules is 0.995 (which is $\{1-(1-0.1)^{50} = 0.995\}$).

CHAPTER 6

CONCLUSIONS

The research developed a decision support system framework for nurse scheduling in hospitals. It then used a heuristic approach to develop schedules for nurses. The approach provides an effective solution to the complex, multiple objective nurse scheduling problem. The implementation is flexible in that it can model a number of operating scenarios and accommodate wide range of hospital and nurses constraints. The heuristic approach is computationally efficient, and overcomes the computational problems reported with use of analytic-based techniques such as Integer Programming and Goal Programming.

The program is flexible to expand and modify. Constraints can be added with minimal effort. The choices of different shift policy ratios are provided in order to enable the user to select schedules for different level of nurses. For example, a user can select shift policy ratio of 3:2:1 for Registered Nurses (RN), 3:3:2 for Licensed Practical Nurses (LPN) and 1:1:1 for Nurse Aides (NA) for the same period of schedules. Also, the possibility to select different set of schedules from the same set of patterns will enable the user to select different portion of the schedules for different level of nurses. For example, a user can generate 1000

patterns with 3:2:1 shift policy ratio, then use 150 of them for RN, 350 for LPN and 500 for NA with actual ratio close to the ideal ratio.

The accuracy (defined in terms of deviations of actual shift allocation to ideal shift policy ratio) increases as number of pattern generated increases. It is therefore recommended that the user generates a large set of patterns and pick the number of schedules close to the number of patterns generated to get better staff allocation. For the implemented model the computational time is small. For example, 1000 four-week patterns with a maximum penalty cost of 40 are generated in less then 2 minutes on a 386-based machine.

Once the schedules have been selected, they can be saved for further use. Therefore, the model can be used both for hospital with cyclical or non-cyclical scheduling approach. The work schedules can be modified using any simple text editor or word processing. The schedules can be easily linked to or imported from a database or spreadsheet program for maintenance purposes, such as changes needed for a specific or emergency request.

The use of personal computer enables small- and medium-size hospitals to implement computerized scheduling without incurring high costs associated with computer hardware, software and maintenance. The program is easy to install and use, and does not require special training for its use.

Implementation of the program on the personal computer will also enable hospitals in developing countries to start doing computerized nurse scheduling because of the relatively inexpensive cost and easy to use nature of personal computers.

6.1 Suggestions for future research

For added flexibility and capability, more constraints can be added to the model, such as the requirement that every nurse must take some proportion of weekend off. More complex hospital environments can be treated with slight modifications of the model. The inclusion of part time or other contract nurses will enhance the utility of the system.

The component of DSS not implemented in this research is nurse allocation. Nurse allocation is interactive in nature and may require changes in work patterns for some nurses. A "real time" component can be added to the system developed in this research by extending it to nurse allocation, or this system can be linked to an existing allocation package (like the one used at Good Samaritan Hospital in Corvallis, Oregon).

Finally, a number of characteristics in personnel scheduling are common across industries. As such the DSS framework and the implemented heuristic can be extended to similar areas of manpower scheduling with some modifications.

REFERENCES

Abernathy, William J., Nicholas Baloff, and John C. Hershey, "The Nurse Staffing Problem: Issues and Prospects", Sloan Management Review, Fall 1971, pp. 87-99.

Abernathy, William J., Nicholas Baloff, John C. Hershey, and Sten Wandel, "A Three-Stage Manpower Planning and Scheduling Model - A Service-Sector Example", Operations Research, Vol. 22, 1973, pp. 693-711.

Ahuja, H. and Sheppard, R., "Computerized Nurse Scheduling", Industrial Engineering, Vol. 7, No. 10, October 1975, pp. 24-29.

Alivizatos, Margaret Sinclair, "A New Concept in Scheduling for Nurses", Supervisor Nurse, February 1981, pp. 20-22.

Arbeiter, Jean S., "The Facts About Foreign Nurses", RN, September 1988, pp.56-63.

Arbeiter, Jean, "What Smart Hospitals Do to Retain Nurses", RN, November 1988, pp. 22-25.

Arnold, Barbara and Mary Etta Mills, "Core-12: Implementation of Flexible Scheduling", The Journal of Nursing Administration, July-August 1983, pp. 9-14.

Arthur, Jeffrey L., and A. Ravindran, "A Multiple Objective Nurse Scheduling Model," AIIE Transactions, Vol. 13, No.1, March 1981, pp. 55-60.

Ballantyne, Donna J., "A Computerized Scheduling System with Centralized Staffing", Journal of Nursing Administration, March 1979, pp. 38-45.

Bechtold, Stephen E., and Michael J. Showalter, "Simple Manpower Scheduling Methods for Managers", Production and Inventory Management, Third Quarter, 1985, pp. 116-133.

Bell, Peter C., Hay, Genevieve, Liang, Y., "Visual Interactive Decision Support Systems for Workforce (Nurse) Scheduling", INFOR, Vol. 24, No. 2, May 1986, pp. 134-145.

Berger, Marie Streng, et al. (eds.), Management for Nurses A Multidisciplinary Approach, second edition, The C. V. Mosby Company, 1980.

Blau, Roger A., and Alan M. Sear, " Nurse Scheduling with a Microcomputer", Journal of Ambulatory Care Management, Vol. 6, No. 3, August 1983, pp. 1-13.

Boldy, Duncan, Operational Research Applied to Health Services, St. Martin's Press, New York.

Borland International, Turbo Pascal 6.0 Library Reference, Borland International, Inc., 1990.

Borland International, Turbo Pascal 6.0 Programmer's Guide, Borland International, Inc., 1990.

Borland International, Turbo Pascal 6.0 User's Guide, Borland International, Inc., 1990.

Boston, Carol Maier and Sarah Karzel, "Will the Nursing Shortage Lead to Liability Suits?", Hospitals, November 20, 1987, pp. 64-67.

Bowlin, Thomas Howard, A Model for Analysis of Personnel Scheduling in Nonstationary Hospital Systems, Ph.D Dissertation, University of Missouri-Columbia, 1978.

Cavaiola, Lawrence J., and John P. Young, "An Integrated System for Patient Assessment and Classification and Nurse Staff Allocation for Long Term Care Facilities", Health Services Research, Vol. 15, Vol. 15, 1980, pp. 281-306.

Emra, Karin L., "Does the Nursing Shortage Change the Rules?", RN, October 1988, pp. 30-34.

Flanders, John and Timothy Lutgen, "The Development of a Microcomputer-based Nurse Management System", Proceedings Annual Symposium on Computer Applications in Medical Care 8th, 1984, IEEE, pp. 618-621.

Frances, Mary Ann, "Implementing a Program of Cyclical Scheduling of Nursing Personnel", Hospitals, Vol. 40, July 16, 1966, pp. 108-125.

Henderson, Willie B., and William L. Berry, "Heuristic Methods for Telephone Operator Shift Scheduling: An Experimental Analysis", Management Science, Vol. 22, No. 12, August 1976, pp. 1372-1380.

Howell, J.P., "Cyclical Scheduling of Nursing Personnel", Hospitals, Vol. 40, Jan. 1966, pp. 77-85.

Jelinek, Richard C., Tim K. Zinn, and James R. Brya, " Tell the Computer How Sick the Patients are and It Will Tell How Many Nurses They Need", Modern Hospital, December 1973, pp. 81-88.

Kao, Edward P.C., and Maurice Queyranne, "Budgeting Costs of Nursing in a Hospital", Management Science, Vol. 31, No. 5, May 1985, pp. 608-621.

Kao, Edward P.C., and Grace G. Tung, "Aggregate Nursing Requirement Planning in a Public Health Care Delivery System", Socio-Economic Planning Science, Vol. 15, 1981, pp. 119-127.

Kaplan, Robert S., "Analysis and Control of Nurse Staffing", Health Sevices Research, Vol. 10, Fall 1975, pp. 278-296.

Koelling, Charles Patrick, Computer-Aided Personnel Scheduling, Ph.D Dissertation, Arizona State University, 1982.

Koelling, Patrick C. and James E. Bailey, "A Multiple Criteria Decision Aid for Personnel Scheduling," IIE Transactions, Vol. 16., No.4, December 1984, pp. 299-307.

Koelling, Patrick C. and James E. Bailey, "Multi-Objective Personnel Scheduling," 1982 Annual Industrial Engineering Conference Proceedings, IIE, pp. 481-487.

Livengood, Lindsay, "Planned Shifts Save Nurses and Dollars", The Modern Hospital, Vol. 104, No. 2, February 1965, pp. 101-104 & 170.

Lukman, Deborah, An Hierarchical Approach in Schedule Formulation and Maintenance Under Uncertainty, Ph.D Dissertation, University of Pittsburgh, 1986.

Maier-Rothe, Christoph and Harry B. Wolfe, "Cyclical Scheduling and Allocation of Nursing Staff", Socio-Economic Planning Science, Vol. 7, 1973, pp. 471-487.

Marriner, Ann, Guide to Nursing Management, The C.V. Mosby Company, 1980.

McGillick, Kathleen, "Modifying Schedules Makes Jobs More Satisfying", Nursing Management, December 1983, pp. 53-55.

Megeath, Joe D., "Successful Hospital Personnel Scheduling", Interfaces, Vol. 8, No. 2, February 1978, pp. 55-59.

Miller, H.E., W.P. Pierskalla and G.J. Rath, "Nurse Scheduling Using Mathematical Programming," Operations Research, Vol. 24, No.5, September-October, 1976.

Morrish, Arthur R., and Anna R. O'Connor, "Cyclic Scheduling", Hospitals, Vol. 44, February 1970, pp. 66-71.

Murray, Donald J., "Computer Makes the Schedules for Nurses", Modern Hospital, December 1971, pp. 104-105.

Musa, A.A., and U. Saxena, "Scheduling Nurses Using Goal Programming Techniques", IIE Transactions, Vol. 16, No. 3, September 1984, pp. 216-221.

Nutt, Paul C., "Decision-Modelling Methods Used to Design Decision Support Systems for Staffing" Medical Care, Vol. 22, No. 11, 1984, pp. 1002-1013.

Ozkarahan, Irem. A Flexible Nurse Scheduling Support System, Ph.D. Dissertation, Arizona State University, 1987.

Ozkarahan, Irem, "A Flexible Nurse Scheduling Support System", 11th Symposium on Computer Applications in Medical Care, Washington D.C., November 1-4, 1987, IEEE Computer Society Press, pp. 387-392.

Ozkarahan, I. and Bailey, JE., "Goal Programming Model Subsystem of a Flexible Nurse Scheduling Support System", IIE Transactions, Vol. 20, No. 3, 1988, pp. 306-316.

Perry, Linda, "Hospitals Go to Market for Nurses", Modern Health Care, April 7, 1989, pp. 24-31.

Rich, Elaine and Kevin Knight, Artificial Intelligence, second edition, McGraw-Hill, Inc. 1991.

Rosenbloom, E.S., and N.F. Goertzen, "Cyclic Nurse Scheduling", European Journal of Operational Research, Vol. 31, No. 1, July 1987, pp. 19-23.

Rothstein, M., "Scheduling Manpower by Mathematical Programming", Industrial Engineering, Vol. 4, No. 4, April 1972, pp. 29-33.

Rothstein, Marvin, "Hospital Manpower Shift Scheduling by Mathematical Programming", Health Services Research, Spring 1973, pp-60-66.

Rugg, Tom and Phil Feldman, Turbo Pascal Programmers's Toolkit, Que Corporation, 1989.

Sedgewick, Robert, Algorithms, second edition, Addison-Wesley Publishing Company, Inc., 1988.

Shuman, Larry J., R. Dixon Speas, Jr., and John P. Young (eds), Operations Research in Health Care, A critical Analysis, The John Hopkins University Press, 1975.

Sinuany-Stern, Zilla, and Yehuda Teomi, "Multi-Objective Scheduling Plans for Security Guards," Journal of Operational Research Society, Vol. 37, No.1, 1986, pp. 67-77.

Sitompul, Darwin, and Randhawa, Sabah U., "Nurse Scheduling: A State-Of-The-Art Review", submitted to Journal of the Society for Health Systems, Vol. 2, No. 1, Spring, 1990, pp 62-72.

Smith, L. Douglas, "The Application of an Interactive Algorithm to develop Cyclical Rotational Schedules for Nursing Personnel," INFOR, Vol. 14, No.1, February 1976.

Smith, L. Douglas and A. Wiggins, "A Computer-based Nurse Scheduling System," Computer & Operations Research, Vol. 4, 1977, pp. 195-212.

Smith-Daniels, Vicki L., Sharon B. Schweikhart, and Dwight E. Smith-Daniels, "Capacity Management in Health Care Services: Review an Future Research Directions", DecisionS ciences, Vol. 19, 1988, pp. 889-919

Spitzer, Murray, "The Computer Art of Schedule Making", Datamation, April 1969, pp. 84-86.

Starr, M.K., and M. Zeleny (Eds), Multiple Criteria Decision Making, TIMS Studies in the Management Sciences, Vol. 6, North-Holland Publishing, 1977.

Steuer, Ralph E., "Multiple Objective Linear Programming with Interval Criterion Weights", Management Science, Vol. 23, No. 3, November 1976, pp. 305-316.

Steuer, R.E., and M.J. Wallance Jr., "A linear multiple objective programming model for manpower selection and allocations decisions", in Charner, A. et al., ed., Management Science Approaches to Management Planning and Organization Design, TIMS Studies in Management Science, Vol. 8., 1978, pp. 193-208.

Tibrewala, R., D. Philippe and J. Browne, " Optimal Scheduling of Two Consecutive Idle Periods", Management Science, Vol. 19, No. 1, September 1972, pp. 71-75.

Tobon Perez, Maria Victoria, An Integrated Methodology to Solve Staffing, Scheduling and Budgeting Problems in a Nursing Department, Ph.D Dissertation, University of Pittsburgh, 1984.

Trivedi, Vandankumar M., "A Mixed-Integer Goal Programming Model for Nursing Service Budgeting", OperationsR esearch, Vol. 29, No. 5, September-October 1981, pp. 1019-1034.

Trivedi, Vandankumar M., and D. Michael Warner, "A Branch and Bound Method for allocation of Float Nurses", Management Science, Vol. 22, No. 9, May 1976, pp. 972-981.

Veranth, Martha M. and Christine Cheson, "Computerized Staffing and Scheduling of PRN Nursing Personnel", Proceedings-Annual Symposium on Computer Applications in Medical Care 8th, 1984, IEEE, pp. 712-717.

Warner, D. Michael, "Nurse Staffing, Scheduling, and Reallocation in the Hospital", Hospital and Health Services Administration, Vol. 21, No. 3, 1976, pp. 77-90.

Warner, D. Michael, "Scheduling Nursing Personnel According to Nursing Preference: A Mathematical Programming Approach", Operations Research, Vol. 24, No. 5, September-October 1976, pp. 842-856.

Warner, D. Michael, "Computer-Aided System for Nurse Scheduling", in Cost Control in Hospitals (John R. Griffith, Walton M. Hancock and Fred C. Munson, Eds.), Health Administration Press, The University of Michigan, Ann Arbor, 1976.

Warner, D. Michael, Don C. Holloway, and Kyle L. Grazier, Decision Making and Control for Health Administration, second edition, Health Administration Press, 1984.

Warner, D. Michael and Juan Prawda, "A Mathematical Programming Model for Scheduling Nursing Personnel in a Hospital", Management Science, Vol.19, No. 4, December, Part I, 1972, pp. 411-422.

White, Harold C., and Michael N. Wolfe, "Nursing Administration and Personnel Administration", The Journal of Nursing Administration, July-August 1983, pp. 15-19.

Wilson, E.J.G., and R.J. Willis, "Scheduling of Telephone Betting Operators - A Case Study", Journal of the Operational Research Society, Vol. 34, No. 10, Oct. 1983, pp. 991-998.

FURTHER READING

Arabeyre, J.P., J. Fearnley, F.C. Steiger and W. Teather, "The Airline Crew Scheduling Problem: A Survey", Transportation Science, Vol.3, 1969, pp. 140-163.

Austin, Larry M., and Parvis Ghandforoush, "A Surrogate Cutting Plane Algorithm for All-Integer Programming", Computers and Operations Research, Vol. 12, No.3, 1985, pp. 241-250.

Baker, Edward Keefer, III, Efficient Heuristic Solutions for the Airline Crew Scheduling Problem, Ph.D Dissertation, University of Maryland, 1979.

Baker, Edward K., Lawrence D. Bodin, William F.Finnegan and Ronny J.Ponder, "Efficient Heuristic Solutions to an Airline Crew Scheduling Problem," AIIE Transactions, June 1979, pp. 190-196.

Bailey, James, and John Field, "Personnel Scheduling with Flexshift Models", Journal of Operations Management, Vol. 5, No. 3, May 1985, pp. 327-338.

Bailey, J., "Integrated Days Off and Shift Personnel Scheduling", Computers & Industrial Engineering, Vol. 9, No. 4, 1985, pp. 395-404.

Baker, Kenneth R., "Scheduling A Full-time Workforce to Meet Cyclic Staffing Requirements", Management Science, Vol. 20, No. 12, August 1974, pp. 1561-1568.

Baker, Kenneth R., "Workforce Allocation in Cyclical Scheduling Problems: A Survey", Operational Research Quarterly, Vol. 27, No.1, 1976, pp. 155-167.

Baker, K.R., R.N. Burns and M.W. Carter, "Staff Scheduling with Day Off and Workstretch Constraints," AIIE Transactions, December, 1980.

Baker, Kenneth R., and Michael J. Magazine, "Workforce Scheduling with Cyclic Demands and Day-off Constraints," Management Science, Vol. 24, No. 2, October 1977, pp.161-167.

Ball, Michael, Lawrence Bodin and Robert Dial, "A Matching Based Heuristic for Scheduling Mass Transit Crews and Vehicles", Transportation Science, Vol. 17, No. 1, February 1983, pp. 4-31.

Ball, Michael and Anito Roberts, "A Graph Partitioning Approach to Airline Crew Scheduling", Transportation Science, Vol. 19, No. 2, May 1985, pp. 107-126.

Bartholdi III, John J., James B. Orlin, and H. Donald Ratliff, "Cyclic Scheduling via Integer Programs with Circular Ones", Operations Research, Vol. 28, No. 5, September-October 1980, pp. 1074-1085.

Bonczek, Robert H., Clyde W. Holsapple, and Andrew B. Whinston, "The Evolving Roles of Models an Decision Support Systems", Decision Sciences, Vol. 11, No. 2, 1980, pp. 337-356.

Brownell, William S., "Scheduling of Work Forces Required in Continuous Operations under Alternative Labor Policies", Management Science, Vol. 22, No. 5, January 1976, pp. 597-605.

Budnick, Frank S., Dennis McLeavey and Richard Mojena, Principles of Operations Research for Management, Second Edition, Richard D. Irwin, 1988.

Buffa, Elwood S., "An Integrated Work Shift Scheduling System", Decision Sciences, Vol. 7, 1976, pp. 620-630.

Bunch, Howard McRaven, "Comparison of the Construction Planning and Manpower Schedules for Building the PD214 General Mobilization Ship in a U.S. Shipyard and in a Japanese Shipyard", Journal of Ship Production, Vol. 3, No. 1, February 1987, pp. 25-36.

Burns,R.N., "Manpower Scheduling with Variable Demands and Alternate Weekends Off, "INFOR, Vol.16, No.2, June, 1978.

Burns, R.N. and G.J. Koop, "A Modular Approach to Optimal Multiple-shift Manpower Scheduling," Operations Research, Vol. 35, No.1, January-February 1987, pp. 100-110.

Burns, R.N and M.W. Carter, "Work Force Size and Single Shift Schedules with Variable Demands",

Management Science, Vol 31, No. 5, May 1985, pp. 599-607.

Cavalier, Tom M. and M. Jeya Chandra, "A Heuristic Algorithm for Assigning Crews Among Bases in an Airlift Operation", Journal of Operational Research Society, Vol. 37, No. 4., 1986, pp. 381-386.

Clark, Gordon M., "Multiple Objective Decision Making," In 1986 International Industrial Engineering Conference Proceedings, IIE, 1986, pp. 304-309.

Cochrane, James L. and Milan Zeleny (Eds.), Multiple Criteria Decision Making, University of South Carolina Press, 1973.

Collura, John and Paul McOwen, "Management Information Systems for Small, Fixed-Route, Fixed-Schedule Operators", Transportation Research Record 994, 1984, pp. 71-75.

Dantzig, G.B., "A Comment on Eddie's "Traffic Delays at Tool Booths", Operations Research, Vol. 2, No. 3, 1954, pp. 339-341.

deVries, Guus, "Nursing Workload Measurement as Management Information", European Journal of Operational Research, Vol. 29, No. 2, May 1987, pp. 199-208.

Edgecumbe, Robert H., "The CASH Approach to Hospital Management Engineering", Hospitals, Vol. 39, March 16, 1965, pp. 70-74.

Edie, L.C., "Traffic Delays at Toll Booths," Journal of Operational Research Society of America, Vol.2, No. 2, 1954, pp. 107-139.

Evans, J.P., and Steuer, R.E., "A Revised Simplex Method for Linear Multiple Objective Programming", Mathematical Programming, Vol. 5, No. 1, 1973.

Evans, James R., "A Microcomputer-Based Decision Support System for Scheduling Umpires in the American Baseball League", Interfaces, Vol. 18, No.6, November-December, 1988, pp. 42-51.

Field, John Martin, Integrated Personnel Scheduling with Flexshift Models, Ph.D. Dissertation, Arizona State University, 1983.

Gabbani, Diaa and Michael Magazine, "An Interactive Heuristic Approach for Multi-Objective Integer-Programming Problems", Journal of Operational Research Society, Vol. 37, No.3, 1986, pp. 285-291.

Ganti, Andrew R. and Emil J. Nagy, "Hospital Decision Support Systems to Optimize Staffing, Service Intensity and Quality", in Proceedings of the 10th Annual Conference on Computers & Industrial Engineering, Computers & Industrial Engineering, Vol. 15, Nos. 1-4, 1988, pp. 272-276.

Garner, Tim, "Security Staffing in Virginia's Adult Prisons", 1986 International Industrial Engineering Conference Proceedings, pp. 461-463.

Giovannetti, Phyllis, "Understanding Patient Classification Systems", Journal of Nursing Administration, February 1979, pp. 4-9.

Gilbert, Kenneth C. and Ruth Bisgrove Hofstra, "An Algorithm for a Class of Three-Dimensional Assignment Problems Arising in Scheduling Applications", IIE Transactions, March 1987, pp. 29-33.

Haessler, Robert W., and F. Brian Talbot, "A 0-1 Model for Solving the Corrugator Trim Problem", Management Science, Vol. 29, No. 2, February 1983, pp. 200-209.

Henderson, Willie B., and William L. Berry, "Heuristic Methods for Telephone Operator Shift Scheduling: An Experimental Analysis", Management Science, Vol. 22, No. 12, August 1976, pp. 1372-1380.

Henderson, Willie B., and William L. Berry, "Determining Optimal Shift Schedules for Telephone Traffic Exchange Operators", Decision Sciences, Vol. 8, 1977, pp. 239-255.

Hershey, John C., William J. Abernathy, and Nicholas Baloff, "Comparison of Nurse Allocation Policies - A Monte Carlo Model", Decision Sciences, Vol. 5, 1974, pp. 58-72.

Hershey, John, William Pierskalla and Sten Wandel, "Nurse Staffing Management," in Boldy, Duncan, Operational Research Applied to Health Services, St. Martin's Press, 1981, pp. 189- 220.

Hershey, John C., Elliot N. Weiss, and Morris A. Cohen, "A Stochastic Service Network Model with Application to Hospital Facilities", Operations Research, Vol. 29, No.1, January- February 1981, pp. 1-22.

Holloran, Thomas J., and Byrn, Judson E., "United Airlines Station Manpower Planning System", Interfaces, Vol. 16, No. 1, January-February 1986, pp. 39-50.

Holt, Charles C., Franco Modligiani and Herbert A. Simon, "A Linear Decision Rule For Production and Employment Scheduling," Management Science, Vol. 2, No. 1, October 1955, pp. 1-30.

Hwang, Ching-Lai and A.S.M. Masud, Multiple Objective Decision Making - Methods and Applications : A State-of-the-Art Survey, Springer-Verlag, 1979.

Hwang, Ching-Lai and Kwangsun Yoon, Multiple Attribute Decision Making - Methods and Applications: A State-of-the-Art Survey, Springer-Verlag, 1981.

Hwang, C.L., S.R. Paidy, K.Yoon and A.S.M. Masud,"Mathematical Programming with Multiple Objectives: A Tutorial,"Computers & Operations Research, Vol. 7, 1980, pp. 5-31.

Ignizio, James P., Linear Programming in Single & Multiple Objective Systems, Prentice Hall Inc., 1982.

Ignizio, James P., "A Review of Goal Programming: A Tool for Multiobjective Analysis", Journal of Operational Research Society, Vol. 29, No. 11, 1978, pp. 1109-1119.

Iskander, Wafik H. and Jar Chou, "Manpower Scheduling for Unbalanced Production Lines," In 1985 Annual International Industrial Engineering Conference Proceedings, IIE, 1985, pp. 546-552.

Jackson, Richard H.F., and Albert W. T. Jones (ed.), "Real-time Optimization in Automated Manufacturing Facilities", National Bureau of Standards, Special Publication 724, 1986, pp. 109-126.

Kamiyama, Angelica, Rotating Manpower Schedules: Algorithmic Developments and Applications, Ph.D Dissertation, Rensselaer Polytechnique Institute, 1984.

Keith, Elbridge Gerry, "Operator Scheduling", AIIE Transactions, March 1979, pp. 185-189.

Kim, Chung Young, Manpower Scheduling Involving Both Delayable and Non-delayable Jobs, Ph.D Dissertation, Arizona State University, 1983.

Kirsch, Kenneth Copeland, Management Policies for Scheduling Patients in an Emergency Room: A Computer Simulation Approach, Ph.D. Dissertation, Northwestern University, 1980.

Koop, Gerald Jacob, Optimal Multiple Shift Manpower Scheduling: Models and Algorithms, Ph.D Dissertation, University of Waterloo (Canada), 1984.

Koop, Gerald J., "Multiple Shift Workforce Lower Bounds", Management Science, Vol. 34, No. 10, October 1988, pp. 1221-1230.

Kornbluth, J.S.H., "A Survey of Goal Programming", OMEGA, Vol.1, No. 2, 1973, pp. 193 -205.

Kuhn, H.W., and A.W. Tucker,"Nonlinear programming," in J. Neyman (Ed.), Proceeding of the 2-nd Berkeley Symposium of Mathematical Statistics and Probability, University of California Press, 1951, pp. 481-491.

Lauer, Joachim, Manpower Scheduling to Accommodate Absenteeism on Assembly Lines, Ph.D Dissertation, Illinois Institute of Technology, 1980.

Lawrence, K.D., Gary R. Reeves and Sheila M. Lawrence,"A Multiple Objective Shift Allocation Model," IIE Transactions, Vol.16, No. 4, December, 1984, pp. 323-328.

Lee, Sang M., Goal Programming for Decision Analysis, First Edition, Auerbach Publishers Inc., 1972.

Levary, Reuven R., "A Zero-One Stochastic Programming Model for Personnel Scheduling Solved by a Sequential Simulation Procedure", Simulation,

Vol. 43, No. 5, November 1984, pp. 247-250.

Liebman, Judith S., John P. Young, and Mandell Bellmore, "Allocation of Nursing Personnel in an Extended Care Facilities", Health Services Research, Vol. 7, Fall 1972, pp. 209-220.

Lowerre, James M., "Work Stretch Properties for the Scheduling of Continuous Operations Under Alternative Labor Policies", Management Science, Vol. 23, No. 9, May 1977, pp. 963-971.

Lowerre, James M., "On Personnel Budgeting for Continuous Operations (with Emphasis on Hospitals)", Decision Sciences, Vol. 10, 1979, pp. 126-135.

Lyons, Joseph P. and John P. Young, "A Staff Allocation Model for Mental Health Facilities", Health Services Research, Vol. 11, Spring 1976, pp. 53-68.

Mabert, Vincent A. and Alan R. Raedels, "The Detail Scheduling of a Part-time Work Force: A Case Study of Teller Staffing", Decision Sciences, Vol. 8, 1977, pp. 109-120.

Monroe, G., "Scheduling Manpower for Service Operations," Industrial Engineering, August 1970, pp.10-17.

Morris, James G., and Michael J. Showalter, "Simple Approaches to Shift, Days-Off and Tour Scheduling Problems", Management Science, Vol. 29, No. 8, August 1983, pp. 942-950.

Ozan, Turgut M., Applied Mathematical Programming for Production and Engineering Management, Prentice-Hall, 1986.

Ravindran, A., Don T. Phillips, and James J. Solberg, Operations Research, Principles and Practice, second edition, John Wiley & Sons, 1987.

Reeves, Gary R. and Lori S. Franz, "A Simplified Interactive Multiple Objective Linear Programming Procedure', Computer & Operations Research, Vol. 12, No.6, 1985, pp. 589-601.

Roberts, S.M., and L.F. Escudero, "Scheduling of Plant Maintenance Scheduling", Journal of Optimization Theory and Applications, Vol. 39, No.

3, March 1983, pp. 323-343.

Rubin, Jerrold, "A Technique for the Solution of Massive Set Covering Problems, with Application to Airline Crew Scheduling", Transportation Science, Vol. 7, 1973, pp. 34-48.

Roberts, S.M., and L.F. Escudero, "Minimum Problem-Size Formulation for the Scheduling of Plant Maintenance Personnel", Journal of Optimization Theory and Applications, Vol. 39, No. 3, March 1983, pp. 345-362.

Saatçioglu, Ömer, "A Multi-attribute Assignment Goal-programming Model with Incentives", Journal of Operational Research Society, Vol. 38, No. 4, 1987, pp. 361-365.

Segal, M., "The Operator-Scheduling Problem: A Network-Flow Approach", Operations Research, Vol. 22., 1974, pp. 808-823.

Sellar, Thomas V., "The 4/40: Does it Raise Personnel Costs?, Hospitals, Vol. 47, September 1, 1973, pp. 94-101.

Shepardson, Fred, and Roy E. Marsten, "A Lagrangean Relaxation Algorithm for The Two Duty Period Scheduling Problem", Management Science, Vol. 26, No. 3, March 1980, pp. 274-281.

Sherali, Hanif D. and Minerva Rios, "An Air Force Crew Allocation and Scheduling Problem", Journal of Operational Research Society, Vol. 35, No.2, 1984, pp. 91-103.

Showalter, M.J., L.J. Krajewski and L.P. Ritzman, "Manpower Allocation in U.S. Postal Facilities: A Heuristic Approach," Computer & Operations Research, Vol. 4, 1977, pp. 257-269.

APPENDICES

APPENDIX A
USER GUIDE AND EXTENSION AND MODIFICATION FOR NURSE SCHEDULING PROGRAM

The nurse scheduling program is easy to use. The program can be invoked by typing :

nurse

follows by pressing <Enter> key. The initial screen (Figure A1.1) will appear containing the name of the program and the programmer. You may press any key to get out of this screen.

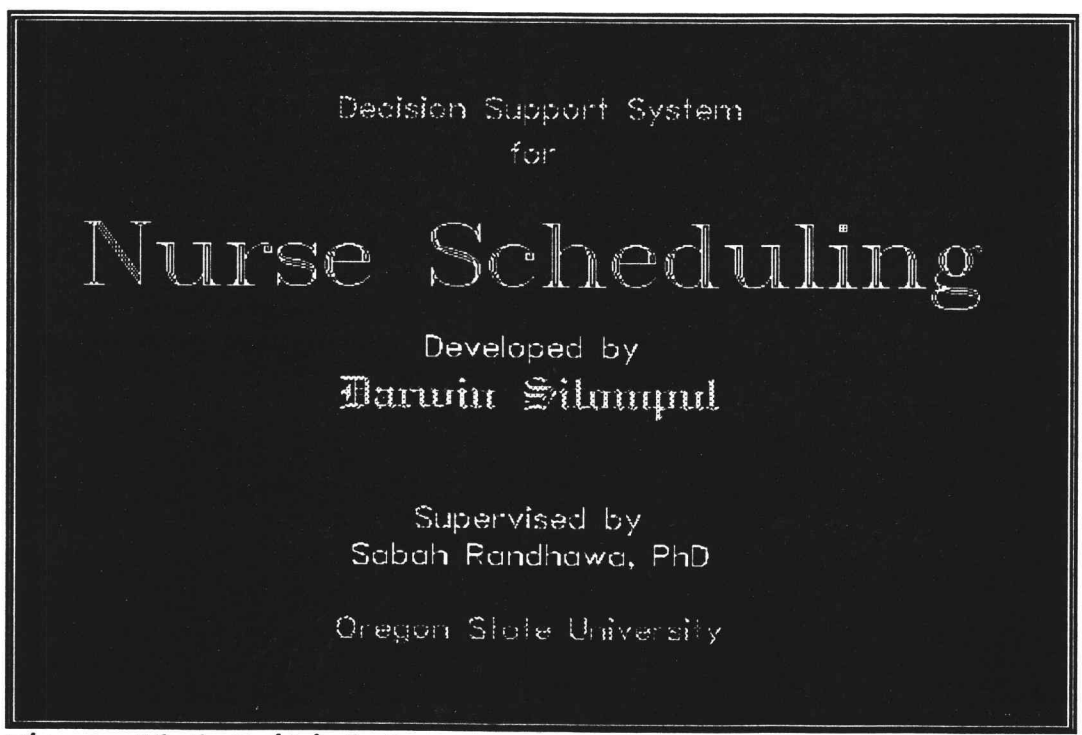

Figure A1.1 Initial Screen

The first screen following the initial screen is a welcome screen which briefly describes the use of the program. Figure A1.2 shows this welcome screen.

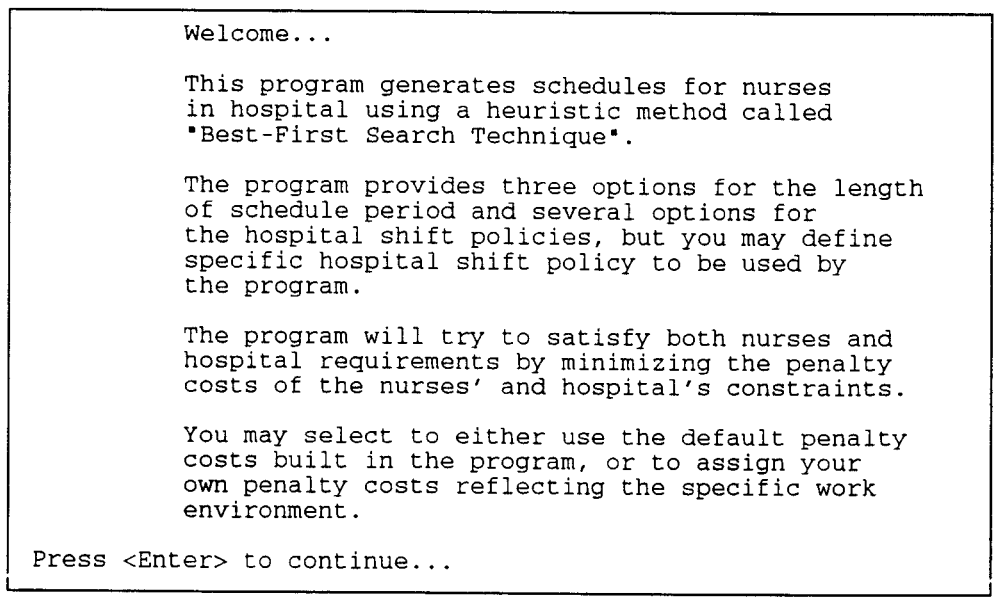

Figure A1.2 Welcome Screen

After you finish reading this welcome message, you may press <Enter> key to continue, and the introduction screen as in Figure A1.3 will appear. This screen briefly describes what you can do with the program. After reading this message, you may press <Enter> key to continue, or, if you decide to terminate or quit the program, you may press <Esc> key. If you decide to continue by pressing the <Enter> key, you will be shown the next screen (Figure A1.4). This step allows you to select the length of schedule period. You may enter 1 for four-week schedule, 2 for eight-week schedule, or 3 for twelve-week schedule. At this stage, you may also quit

```
...
You will first be asked to specify the number of
patterns you want to generate. It is recommended
that for flexibility you specify this number to be
greater than the number of schedules required.

Once the program generates a set of patterns,
it will provide you with several set of schedules
from which you may select the number you need.

You may view the schedules and the staff list
report before you decide to pick and save schedules
or to discard the schedules and generate another
set.

If you think that the patterns you have generated
do not satisfy your needs, you may generate
another pattern set.

Good Luck...

Press any key to continue or <Esc> to terminate ...
```

Figure A1.3 Introduction Screen

```
LENGTH OF SCHEDULE :
____________________

1. Four-week.
2. Eight-week.
3. Twelve-week.
4. Exit

Select 1, 2 or 3 (4 for exit) :
```

Figure A1.4 Choice of Length of Schedule Period

the program by entering 4.

If you choose 1, or 2, or 3, the next screen will be the screen that shows you the constraints with their associated default penalty costs built-in the program (Figure A1.5). The penalty costs for this set of constraints can be modified if so desired.

To choose default penalty costs, you may enter **Y** or **y**. If you prefer to define your own penalty costs, select **N** or **n**.

If you choose to use default penalty cost, then the program will ask you the maximum penalty cost that

```
Default Values of Penalty Costs:


Penalty Cost for having 3 consecutive days off     := 5
Penalty Cost for having 4 consecutive days off     := 10
Penalty Cost for working 6 consecutive days        := 10
Penalty Cost for working 7 consecutive days        := 30
Penalty Cost for working 8 consecutive days        := 40
Penalty Cost for having single day off             := 10
Penalty Cost for having single day on              := 10
Penalty Cost for working on Saturday, Sunday off   := 5
Penalty Cost for working on Sunday, Saturday off   := 5
Penalty Cost for working on Week End               := 10


MAXIMUM PENALTY COST can be set equal to the highest
value of penalty cost. The higher the maximum penalty
cost the less strict the schedules.

Use default value for Penalty Costs? (Y/N):
```

Figure A1.5 Default Penalty Cost Built in the Program

you can still tolerate. The program also give you guidelines on selecting the maximum penalty cost. Figure A1.6 shows this process.

If you choose not to use the default penalty costs and enter **N** or **n**, then you will be allowed to define your own penalty costs. The screen (Figure A1.7) also displays penalty costs values. You may then enter the penalty costs one by one as shown in Figure A1.7.

```
Default Values of Penalty Costs:


Penalty Cost for having 3 consecutive days off     :=  5
Penalty Cost for having 4 consecutive days off     := 10
Penalty Cost for working 6 consecutive days        := 10
Penalty Cost for working 7 consecutive days        := 30
Penalty Cost for working 8 consecutive days        := 40
Penalty Cost for having single day off             := 10
Penalty Cost for having single day on              := 10
Penalty Cost for working on Saturday, Sunday off   :=  5
Penalty Cost for working on Sunday, Saturday off   :=  5
Penalty Cost for working on Week End               := 10


MAXIMUM PENALTY COST can be set equal to the highest
value of penalty cost. The higher the maximum penalty
cost the less strict the schedules.

Use default value for Penalty Costs? (Y/N): y

Enter maximum penalty cost:  40
```

Figure A1.6 Entering Maximum Penalty Cost (Case of Default Penalty Costs)

```
Penalty Cost for having 3 consecutive days off     :=  5;
Penalty Cost for having 4 consecutive days off     := 10;
Penalty Cost for working 6 consecutive days        := 10;
Penalty Cost for working 7 consecutive days        := 30;
Penalty Cost for working 8 consecutive days        := 40;
Penalty Cost for having single day off             := 10;
Penalty Cost for having single day on              := 10;
Penalty Cost for working on Saturday, Sunday off   :=  5;
Penalty Cost for working on Sunday, Saturday off   :=  5;
Penalty Cost for working on Week End               := 10; )


Enter Penalty Cost for having 3 consecutive days off     : 10
Enter Penalty Cost for having 4 consecutive days off     : 20
Enter Penalty Cost for working 6 consecutive days        : 10
Enter Penalty Cost for working 7 consecutive days        : 60
Enter Penalty Cost for working 8 consecutive days        : 100
Enter Penalty Cost for having single day off             : 5
Enter Penalty Cost for having single day on              : 5
Enter Penalty Cost for working on Saturday, Sunday off   : 5
Enter Penalty Cost for working on Sunday, Saturday off   : 5
Enter Penalty Cost for working on Week End               : 50

      Enter maximum penalty cost:  100
```

Figure A1.7 Example of Customizing Penalty Costs

Following this step, the program will ask for the number of patterns you want to generate. You may enter the number as required, e.g., 1000. Please note that with this number of patterns, you may generate unlimited number of different schedules. The screen will show:

```
Enter number of patterns to be generated:
```

The program will then create feasible patterns as many as you require. This process takes very little time. For example, for 1000 patterns with default penalty costs and maximum penalty cost of 40, it will take less than 2 minutes on an Intel 386-based computer (average 1 minute and 25 seconds).

After generating the patterns, program will ask you to select shift policy ratio (Figure A1.8). There are 9 ratios built-in the program; in addition, you may define your own shift policy ratios. If you decide to use one of the ratios provided by the program, just enter the number corresponding to it.

If you choose to define your own shift policy ratio, then you may type **10** and press the <Enter> key (Figure A1.9). Then you will be asked to enter the day, evening and night shift policy ratio, respectively. Example is shown in Figure A1.10 (assuming your shift policy ratio is 5:2:1).

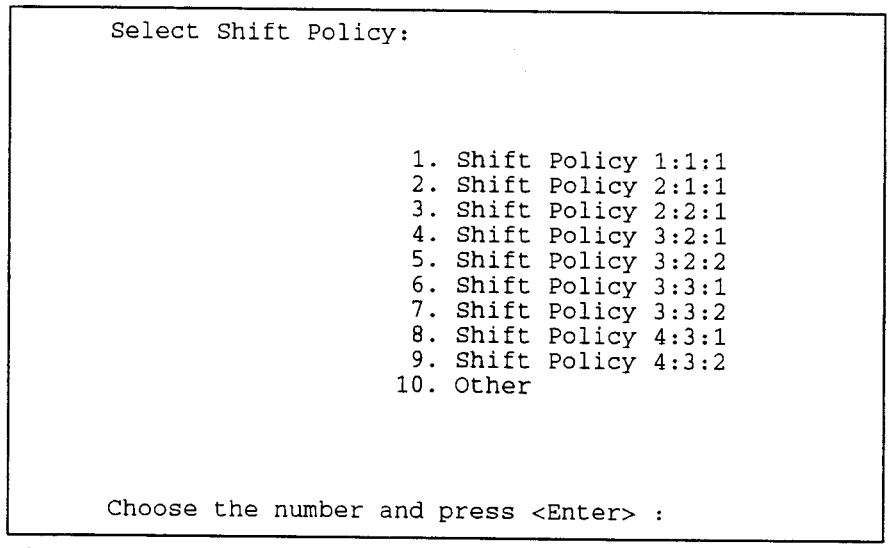

Figure A1.8 Shift Policy Ratio

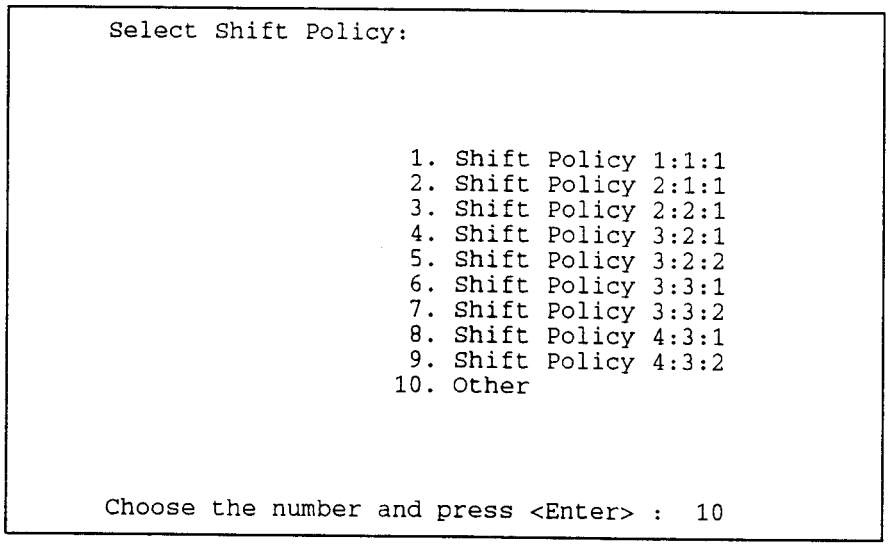

Figure A1.9 Choosing to Define Your Own Shift Policy Ratio

```
Select Shift Policy:

Day Ratio     : 5
Evening Ratio : 2
Night Ratio   : 1
```

Figure A1.10 Entering Your Own Shift Policy Ratio

In either case, the program next asks for the number of schedules you want to generate. The maximum number of schedules that you can pick will also be shown. This number will be equal to the number of pattern you asked the program to generate. The screen will show (assuming you enter 1000 as the number of patterns you want to generate):

```
Enter number of schedules to be selected  (Maximum = 1000 ):
```

After you enter the required number, the program will ask you if you want to display the schedules on the screen:

```
Display schedules on the screen? (Y/N)
```

You may choose not to display by entering **N** or **n**. If you want to see the schedule, then you may type **Y** or **y**. The schedule appear on the screen as shown in Figure A1.11.

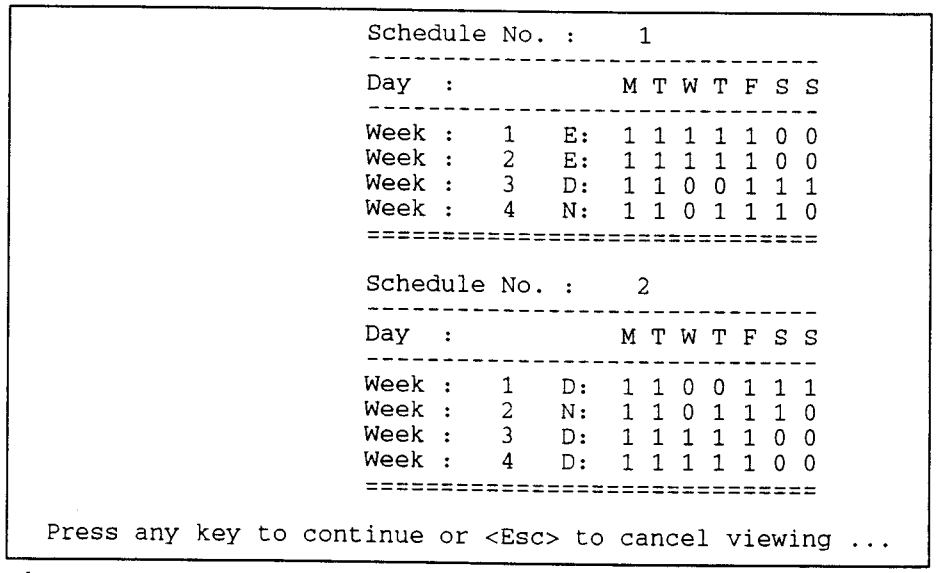

Figure A1.11 Example of Schedules Displayed on the Screen

For your convenience, the program will display two schedules at a times for four-week period, and one schedules at a times for eight and twelve week period. You may press any key to continue viewing the schedules until all the schedules are shown. However, if you want to cancel the viewing process, you simply press <Esc> key.

The next step will be generating the Staff List or Staff Requirement Report. This takes a negligible amount of time. The program will ask if you want to see the report. This report is necessary to evaluate the schedule. The report shows some statistics that you can use to compare the shift policy ratio and the actual ratio given by the schedules generated. The program shows:

```
Display Staff List Report? (Y/N)
```

and you may press **Y** or **y**, or **N** or **n**. If you select to enter **Y** or **y**, then the report will be shown on the screen. You may need to press <Enter> key several times to see the entire report. An example is shown in Figure A1.12 and A1.13.

After viewing the report you will be given a choice to save the schedules. The program will ask:

```
Save the schedules? (Y/N)
```

If you think that the schedules are good enough, then you may save them. You will then be asked to select either to save the Staff List Report or not, and either to print the schedules, and/or the Staff List Report. If you do, just enter **Y** or **y**,

STAFF LIST REPORT:

Day :		Day-Shift	Evening-Shift	Night-Shift
Monday	1	23	30	36
Tuesday	2	27	34	38
Wednesday	3	25	33	38
Thursday	4	23	32	38
Friday	5	18	26	27
Saturday	6	7	5	5
Sunday	7	12	10	13
Monday	8	30	28	31
Tuesday	9	34	30	30
Wednesday	10	33	23	29
Thursday	11	27	21	23
Friday	12	23	15	23
Saturday	13	13	15	15
Sunday	14	20	23	14
Monday	15	27	23	25
Tuesday	16	28	30	28
Wednesday	17	24	28	26

Press <Enter> to continue ...

Figure A1.12 First Part of 4-week Report

```
     Sunday        14          20          23          14
     Monday        15          27          23          25
     Tuesday       16          28          30          28
     Wednesday     17          24          28          26
Press <Enter> to continue ...

     Thursday      18          20          29          25
     Friday        19          21          25          18
     Saturday      20          21          24          16
     Sunday        21          19          21          22
     Monday        22          20          21          25
     Tuesday       23          21          23          27
     Wednesday     24          23          22          28
     Thursday      25          27          24          28
     Friday        26          24          26          30
     Saturday      27          21          20          28
     Sunday        28          19          19          24
   ---------------------------------------------------------------------
     Total                    630         660         710
     Average               22.500      23.571      25.357
     Ratio Actual           0.315       0.330       0.355
     Ratio Ideal            0.333       0.333       0.333
   =====================================================================
Press <Enter> to continue ...
```

Figure A1.13 Last Part of 4-week Report

and the program will execute the appropriate commands.

You may save the schedules and can still generate another set of schedules by answering the appropriate set of questions. If for example you do not want to save or print the schedules and the report, you may answer all enquiries with **N** or **n**, as shown in Figure A1.14.

```
     Save the schedules? (Y/N)    n

     Save Staff List Report? (Y/N)    n

     Print the schedules? (Y/N)    n

     Print StaffListReport? (Y/N)    n
```

Figure A1.14 Examples of Save and Print Choice

Then, you may select another set of schedules by answering **Y**

or **y** to this question:

```
Select another set of schedules? (Y/N)
```

Then the screen as in Figure A1.8 will be shown, and you may continue the process as described above.

If you decide to enter **N** or **n** then Figure A1.15 will be shown on the screen. If you want to generate another set of patterns, you may answer with **Y** or **y**, and the program will repeat starting from Figure A1.4. If you decide not to generate another patterns, then you may answer with **N** or **n**, and the program will terminate.

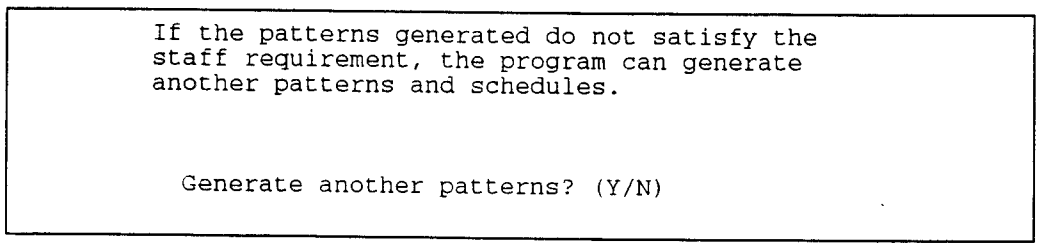

Figure A1.15 Last Chance to Generate Another Set of Patterns

Extension and Modification

To extend and modify the program, user needs to be able to program in Pascal language, preferably using Borland Turbo Pascal Version 6.0.

The program consists of several files:

1. **NURSE.PAS**: the main program source code.

2. **PATGEN.PAS**: the unit source code to generate work patterns.
3. **PNTCOST.PAS**: the unit source code to calculate penalty costs.
4. **SORTPAT.PAS**: the unit source code to sort work patterns based on penalty costs.
5. **PATGEN.TPU**: the unit resulted by compiling PATGEN.PAS.
6. **PNTCOST.TPU**: the unit resulted by compiling PCOST.PAS
7. **SORTPAT.TPU**: the unit resulted by compiling SORTPAT.PAS
8. **GRAPH.TPU**: the unit needed to generate initial screen.
9. **CRT.TPU**: the unit needed to handle DOS commands in the program.
10. **BASE.DAT**: the file to store work patterns.
11. **SCHDBAS.DAT**: the work pattern base for 4-week patterns to be used to generate schedules.
12. **SCHDBAS1.DAT**: the first randomly reordered work patterns.
13. **SCHDBAS2.DAT**: the second randomly reordered work patterns.
14. **SCHD8WK.DAT**: the work patterns base for 8-week schedule period.
15. **SCHD12WK.DAT**: the work patterns base for 12-week schedule period.
16. **SHIFPAT.DAT**: the shift patterns base.
17. **NEWSHIFT.DAT**: the randomly reordered shift patterns (4-week).
18. **SHIFBASE.DAT**: the first randomly reordered shift patterns.

19. **SH2BASE.DAT**: the second randomly reordered shift patterns.
20. **SH3BASE.DAT**: the third randomly reordered shift patterns.
21. **SH8BASE.DAT**: the shift patterns base for 8-week schedule period.
22. **SH12BASE.DAT**: the shift patterns base for 12-week schedule period.
23. **SCHEDULE.DAT**: the schedule base for 4-week period.
24. **BASE8.DAT**: the schedule base for 8-week period.
25. **BASE12.DAT**: the schedule base for 12-week period.
26. **STAF4.DAT**: the Staff List Report for 4-week period.
27. **STAF8.DAT**: the Staff List Report for 8-week period.
28. **STAF12.DAT**: the Staff List Report for 12-week period.
29. Several files with **.BGI** extension which are needed for the program to adjust to different graphics card of the computer. The program will self-detect the card of the computer you are using.

The first modification that can be carried out is whether the user want to have the new pattern and schedule bases appended to the old bases or not. The default program will overwrite the old patterns and schedule bases every time the user run the program. If you want the new data will be appended to the old one, simply change the command of preparation of the required files from **rewrite** to **append** and recompile the program.

To add new work pattern, just modify the constants of work pattern which is **OneWeekPattern** in unit PATGEN.PAS, then recompile the unit and the program. Please note that if you modify the work pattern, you should modify the size of the array accordingly.

To add new constraints, modify the unit PATGEN.PAS and PNTCOST.PAS. Please note that if you modify the constraint, you may need to add algorithm to calculate the penalty costs accordingly.

If you want to modify the default value of penalty costs and shift policy simply change the value in the unit PNTCOST.PAS (for penalty costs) and main program for the shift policy ratio.

The result of the program will be in ASCII text mode, therefore it can be imported from any text editor, spreadsheet, or database software, as for example, if changes needed in the daily use of the program and/or to assigning schedules to the nurses' database.

APPENDIX B
SOURCE CODE OF UNIT FOR WORK PATTERN GENERATION

```
Unit PatGen;

interface

Type

    OneWeek        = string[7];
    TwoWeek        = string[14];
    ThreeWeek      = string[21];
    FourWeek       = string[28];
    TwoWeekPat     = Array [1..21] of TwoWeek;
    ThreeWeekPat   = Array [1..21] of ThreeWeek;
    FourWeekPat    = Array [1..21] of FourWeek;

const

     OneWeekPattern : Array [1..21] of OneWeek =
                      ('1111100','1111001','1110011','1100111','1001111',
                       '0011111','1110101','1101101','1011101','0111101',
                       '1101011','1011011','0111011','1010111','0110111',
                       '0101111','1111010','1110110','1101110','1011110',
                       '0111110');

     {NOTES:                                                              }
     {This OneWeekPattern can be expanded to include all possible combina-}
     {nation of one-week patterns only by changing the size of the array  }
     {and adding the new patterns.                                        }

var

   TwoWeekPattern           : TwoWeekPat;
   ThreeWeekPattern         : ThreeWeekPat;
   FourWeekPattern          : FourWeekPat;
   i, j, k, l, m, q, r, s   : integer;

Procedure TwoWeekPatternGeneration(var TwoWeekPattern : TwoWeekPat;
                                       q : integer);
Procedure ThreeWeekPatternGeneration(var ThreeWeekPattern : ThreeWeekPat;
                                         r : integer);
Procedure FourWeekPatternGeneration(var FourWeekPattern : FourWeekPat;
                                        s : integer);

implementation

Procedure TwoWeekPatternGeneration(var TwoWeekPattern : TwoWeekPat;
                                       q : integer);

begin

     k := 1;

     {for i:=1 to 21 do begin}
         for j := 1 to 21 do begin
             TwoWeekPattern [k] := OneWeekPattern[q] + OneWeekPattern[j];
             {writeln('TwoWeekPattern ',k,' is ',TwoWeekPattern [k]);}
             k := k + 1;
         end;
     end; {******************TwoWeekPatternGeneration*********************}
```

```
{*************************************************************************}

Procedure ThreeWeekPatternGeneration(var ThreeWeekPattern : ThreeWeekPat;
                                         r : integer);

begin

     l := 1;

     {TwoWeekPatternGeneration(TwoWeekPattern);}

     {for i := 1 to 21 do begin}
         for j := 1 to 21 do begin
             ThreeWeekPattern[l] := TwoWeekPattern[r] + OneWeekPattern[j];
             {writeln('ThreeWeekPattern ',l,' is ', ThreeWeekPattern[l]);}
             l := l + 1;
         end;
     end; {*****************ThreeWeekPatternGeneration*******************}

{*************************************************************************}

Procedure FourWeekPatternGeneration(var FourWeekPattern : FourWeekPat;
                                        s : integer);

begin

     m := 1;

     {ThreeWeekPatternGeneration(ThreeWeekPattern);}
     {for i := 1 to 21 do begin}
         for j := 1 to 21 do begin
             FourWeekPattern[m] := ThreeWeekPattern[s] + OneWeekPattern[j];
             {writeln('FourWeekPattern ',m,' is ',FourWeekPattern[m]);}
             m := m + 1;
         end;
     end; {*****************FourWeekPatternGeneration*******************}
end. {Of Unit PatGen}
{*************************************************************************}
```

APPENDIX C
SOURCE CODE OF THE UNIT FOR CALCULATING PENALTY COST

```
Unit PntCost;

interface

type
    TotalPCost    = Array [1..21] of integer;
var
   PenaltyCost,
   TotalPenaltyCost0,
   TotalPenaltyCost1,
   TotalPenaltyCost2,
   TotalPenaltyCost3,
   TotalPenaltyCost4,
   TotalPenaltyCost5,
   TotalPenaltyCost6,
   TotalPenaltyCost7,
   TotalPenaltyCost8,
   TotalPenaltyCost9,
   TotalPenaltyCost10,
   TotalPenaltyCost11,
   TotalPenaltyCost12,
   TotalPenaltyCost13,
   TotalPenaltyCost14,
   TotalPenaltyCost15      : Integer;
   PenaltyCost3Zero,
   PenaltyCost4Zero,
   PenaltyCost6One,
   PenaltyCost7One,
   PenaltyCost8One,
   PenaltyCostSingle0,
   PenaltyCostSingle1,
   PenaltyCostSatOnSunOff,
   PenaltyCostSatOffSunOn,
   PenaltyCostWEndOn       : Integer;
   i,j,k,l,m,n             : Integer;


         Procedure Initialize;
         Procedure DefaultPCost;
         Procedure CustomizePCost;
         Procedure OneWeekPenaltyCost(var TotalPenaltyCost : TotalPCost);
         Procedure TwoWeekPenaltyCost(var TotalPenaltyCost : TotalPCost);
         Procedure ThreeWeekPenaltyCost(var TotalPenaltyCost: TotalPcost);
         Procedure FourWeekPenaltyCost(var TotalPenaltyCost: TotalPcost);

Implementation

Uses Crt, PatGen;

Procedure Initialize; { Initialization of Penalty Cost }

{ This initialization is needed to calculate Penalty Cost for every       }
{ pattern. Everytime the penalty cost must be initialize to zero,         }
{ otherwise the Penalty Cost will be accumulated.                         }
```

```
begin

     TotalPenaltyCost0  := 0;
     TotalPenaltyCost1  := 0;
     TotalPenaltyCost2  := 0;
     TotalPenaltyCost3  := 0;
     TotalPenaltyCost4  := 0;
     TotalPenaltyCost5  := 0;
     TotalPenaltyCost6  := 0;
     TotalPenaltyCost7  := 0;
     TotalPenaltyCost8  := 0;
     TotalPenaltyCost9  := 0;
     TotalPenaltyCost10 := 0;
     TotalPenaltyCost11 := 0;
     TotalPenaltyCost12 := 0;
     TotalPenaltyCost13 := 0;
     TotalPenaltyCost14 := 0;
     TotalPenaltyCost15 := 0;

end;

{*************************************************************************}

Procedure DefaultPCost; { This procedure give a default value for        }
                        { Penalty Costs                                  }

begin

     PenaltyCost3Zero        :=  5;
     PenaltyCost4Zero        := 10;
     PenaltyCost6One         := 10;
     PenaltyCost7One         := 30;
     PenaltyCost8One         := 40;
     PenaltyCostSingle0      := 10;
     PenaltyCostSingle1      := 10;
     PenaltyCostSatOnSunOff  :=  5;
     PenaltyCostSatOffSunOn  :=  5;
     PenaltyCostWEndOn       := 10;

end; { ****** Default Penalty Cost ****** }

{*************************************************************************}

Procedure CustomizePCost; { This procedure give the user a chance to     }
                          { assign the value of Penalty Cost by him/her- }
                          { self.                                        }

begin

ClrScr;

     Writeln;
     Writeln ('Enter Penalty Cost for every cases :');
     Writeln;
     Writeln('( Example :');
     Writeln;
     Writeln('Penalty Cost for having 3 consecutive days off    :=  5;');
     Writeln('Penalty Cost for having 4 consecutive days off    := 10;');
     Writeln('Penalty Cost for working 6 consecutive days       := 10;');
     Writeln('Penalty Cost for working 7 consecutive days       := 30;');
     Writeln('Penalty Cost for working 8 consecutive days       := 40;');
     Writeln('Penalty Cost for having single day off             := 10;');
     Writeln('Penalty Cost for having single day on              := 10;');
     Writeln('Penalty Cost for working on Saturday, Sunday off  :=  5;');
     Writeln('Penalty Cost for working on Sunday, Saturday off  :=  5;');
```

```
      Writeln('Penalty Cost for working on Week End              := 10; )');
      Writeln;
      Writeln;
      Write('Enter Penalty Cost for having 3 consecutive days off    : ');
      Readln (PenaltyCost3Zero);
      Write('Enter Penalty Cost for having 4 consecutive days off    : ');
      Readln (PenaltyCost4Zero);
      Write('Enter Penalty Cost for working 6 consecutive days       : ');
      Readln (PenaltyCost6One);
      Write('Enter Penalty Cost for working 7 consecutive days       : ');
      Readln (PenaltyCost7One);
      Write('Enter Penalty Cost for working 8 consecutive days       : ');
      Readln (PenaltyCost8One);
      Write('Enter Penalty Cost for having single day off            : ');
      Readln (PenaltyCostSingle0);
      Write('Enter Penalty Cost for having single day on             : ');
      Readln (PenaltyCostSingle1);
      Write('Enter Penalty Cost for working on Saturday, Sunday off  : ');
      Readln (PenaltyCostSatOnSunOff);
      Write('Enter Penalty Cost for working on Sunday, Saturday off  : ');
      Readln (PenaltyCostSatOffSunOn);
      Write('Enter Penalty Cost for working on Week End              : ');
      Readln (PenaltyCostWEndOn);

end; { ******* Customize Penalty Cost ****** }

Procedure OneWeekPenaltyCost(var TotalPenaltyCost : TotalPCost);

begin

      for i := 1 to 21 do begin

          Initialize;

{Penalty Cost for working on WeekEnd}

      if (OneWeekPattern[i,6] = '1') and (OneWeekPattern[i,7] = '1')
         then TotalPenaltyCost1 := PenaltyCostWEndOn;
      if (OneWeekPattern[i,6] = '1') and (OneWeekPattern[i,7] = '0')
         then TotalPenaltyCost2 := PenaltyCostSatOnSunOff;

      for m := 2 to 6 do begin
          if (OneWeekPattern [i,m] = '0') and
             (OneWeekPattern [i,m] <> OneWeekPattern [i,m+1]) and
             (OneWeekPattern [i,m-1] <> OneWeekPattern [i,m]) then
             PenaltyCost := PenaltyCostSingle0
          else
             PenaltyCost := 0;

      TotalPenaltyCost3 := TotalPenaltyCost3 + PenaltyCost;
      end;

      for n := 2 to 6 do begin
          if (OneWeekPattern [i,n] = '1') and
             (OneWeekPattern [i,n] <> OneWeekPattern [i,n+1]) and
             (OneWeekPattern [i,n-1] <> OneWeekPattern [i,n]) then
             PenaltyCost := PenaltyCostSingle1
          else
             PenaltyCost := 0;

      TotalPenaltyCost4 := TotalPenaltyCost4 + PenaltyCost;
      end;

      TotalPenaltyCost[i] := TotalPenaltyCost1 + TotalPenaltyCost2
                           + TotalPenaltyCost3 + TotalPenaltyCost4;
```

```
     {writeln ('OneWeekPattern ',i,' Cost = ',TotalPenaltyCost[i]:3);}
end;
end; {************OneWeekPenaltyCost************************************}

{*******************************************************************}

Procedure TwoWeekPenaltyCost(var TotalPenaltyCost : TotalPCost);

begin

     {TwoWeekPatternGeneration(TwoWeekPattern);}

     for i := 1 to 21 do begin

     Initialize;

{Penalty Cost for working on WeekEnd}

     if (TwoWeekPattern[i,6] = '1') and (TwoWeekPattern[i,7] = '1')
        then TotalPenaltyCost4 := PenaltyCostWEndOn;
     if (TwoWeekPattern[i,13]='1') and (TwoWeekPattern[i,14] = '1')
        then TotalPenaltyCost5 := PenaltyCostWEndOn;

     if (TwoWeekPattern[i,6] = '1') and (TwoWeekPattern[i,7] = '0')
        then TotalPenaltyCost7 := PenaltyCostSatOnSunOff;
     if (TwoWeekPattern[i,13]='1') and (TwoWeekPattern[i,14] = '0')
        then TotalPenaltyCost8 := PenaltyCostSatOnSunOff;

     if (TwoWeekPattern[i,6] = '0') and (TwoWeekPattern[i,7] = '1')
        then TotalPenaltyCost10 := PenaltyCostSatOffSunOn;
     if (TwoWeekPattern[i,13]='0') and (TwoWeekPattern[i,14] = '1')
        then TotalPenaltyCost11 := PenaltyCostSatOffSunOn;


{Penalty Cost for single day on or off}

     for m := 2 to 13 do begin
         if (TwoWeekPattern [i,m] = '0') and
            (TwoWeekPattern [i,m] <> TwoWeekPattern [i,m+1]) and
            (TwoWeekPattern [i,m-1] <> TwoWeekPattern [i,m]) then
            PenaltyCost := PenaltyCostSingle0
         else
            PenaltyCost := 0;

     TotalPenaltyCost2 := TotalPenaltyCost2 + PenaltyCost;
     end;

     for n := 2 to 13 do begin
         if (TwoWeekPattern [i,n] = '1') and
            (TwoWeekPattern [i,n] <> TwoWeekPattern [i,n+1]) and
            (TwoWeekPattern [i,n-1] <> TwoWeekPattern [i,n]) then
            PenaltyCost := PenaltyCostSingle1
         else
            PenaltyCost := 0;

     TotalPenaltyCost3 := TotalPenaltyCost3 + PenaltyCost;
     end;


{Penalty Cost for eight or more consecutive days on}

         j := 1;
         while j <= 14 do begin
             if (TwoWeekPattern [i,j]='1') and
             (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+1)]) and
```

```
            (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+2)]) and
            (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+3)]) and
            (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+4)]) and
            (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+5)]) and
            (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+6)]) and
            (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+7)]) then
            begin PenaltyCost := PenaltyCost8One; j := j + 8; end
            else
            begin PenaltyCost := 0; j :=  j + 1; end;
            TotalPenaltyCost1 := TotalPenaltyCost1 + PenaltyCost;
            end;


{Penalty Cost for seven consecutive days on}

            if TotalPenaltyCost1 = 0 then begin
               j := 1;
               while j <= 14 do begin
                     if (TwoWeekPattern [i,j]='1') and
                     (TwoWeekPattern [i,j]=TwoWeekPattern[i,(j+1)]) and
                     (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+2)]) and
                     (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+3)]) and
                     (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+4)]) and
                     (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+5)]) and
                     (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+6)]) then
                     begin
                          PenaltyCost := PenaltyCost7One;
                          j := j + 7;
                     end
                     else
                     begin PenaltyCost :=0; j :=  j + 1; end;
                     TotalPenaltyCost1 := TotalPenaltyCost1 + PenaltyCost;
                end;
         end;

{Penalty Cost for six consecutive days on}


         if TotalPenaltyCost1 = 0 then begin
         j := 1;
         while j <= 14 do begin
              if (TwoWeekPattern [i,j]='1') and
              (TwoWeekPattern [i,j]=TwoWeekPattern[i,(j+1)]) and
              (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+2)]) and
              (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+3)]) and
              (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+4)]) and
              (TwoWeekPattern[i,j]=TwoWeekPattern[i,(j+5)]) then
              begin PenaltyCost := PenaltyCost6One; j := j + 6; end
              else
              begin PenaltyCost := 0; j :=  j + 1; end;
              TotalPenaltyCost1 := TotalPenaltyCost1 + PenaltyCost;
         end;
         end;

{Penalty Cost for four consecutive days off}


              j := 1;
              while j <= 14 do begin
              if (TwoWeekPattern [i,j] = '0') and
                 (TwoWeekPattern [i,j] = TwoWeekPattern [i, j+1]) and
                 (TwoWeekPattern [i,j] = TwoWeekPattern [i, j+2]) and
                 (TwoWeekPattern [i,j] = TwoWeekPattern [i, j+3]) then
                 begin PenaltyCost := PenaltyCost4Zero; j := j + 4; end
                 else
```

```
                begin PenaltyCost := 0; J := j + 1; end;
              TotalPenaltyCost0 := TotalPenaltyCost0 + PenaltyCost;
              end;

{Penalty Cost for three consecutive days off}


              if TotalPenaltyCost0 = 0 then begin
              j := 1;
              while j <= 14 do begin
              if (TwoWeekPattern [i,j] = '0') and
                 (TwoWeekPattern [i,j] = TwoWeekPattern [i, j+1]) and
                 (TwoWeekPattern [i,j] = TwoWeekPattern [i, j+2]) then
                 begin PenaltyCost := PenaltyCost3Zero; j := j + 3; end
                 else
                 begin PenaltyCost := 0; J := j + 1; end;
              TotalPenaltyCost0 := TotalPenaltyCost0 + PenaltyCost;
              end;
              end;

            TotalPenaltyCost[i] := (TotalPenaltyCost0 + TotalPenaltyCost1
                                 + TotalPenaltyCost2 + TotalPenaltyCost3
                                 + TotalPenaltyCost4 + TotalPenaltyCost5
                                 + TotalPenaltyCost6 + TotalPenaltyCost7
                                 + TotalPenaltyCost8 + TotalPenaltyCost9
                                 + TotalPenaltyCost10 + TotalPenaltyCost11
                                 + TotalPenaltyCost12);

      {writeln ('TwoWeekPattern ',i,' Cost = ',TotalPenaltyCost[i]:3);}
      end;
end; {********************TwoWeekPenaltyCost*****************************}

{************************************************************************}

Procedure ThreeWeekPenaltyCost(var TotalPenaltyCost: TotalPcost);

begin

{ThreeWeekPatternGeneration(ThreeWeekPattern);}


for i := 1 to 21 do begin {ThreeWeekPattern Penalty Cost Calculation}

         Initialize;

{Penalty Cost for working on WeekEnd}

     if (ThreeWeekPattern[i,6] = '1') and (ThreeWeekPattern[i,7] = '1')
        then TotalPenaltyCost4 := PenaltyCostWEndOn;
     if (ThreeWeekPattern[i,13]='1') and (ThreeWeekPattern[i,14] = '1')
        then TotalPenaltyCost5 := PenaltyCostWEndOn;
     if (ThreeWeekPattern[i,20]='1') and (ThreeWeekPattern[i,21]='1')
        then TotalPenaltyCost6 := PenaltyCostWEndOn;

     if (ThreeWeekPattern[i,6] = '1') and (ThreeWeekPattern[i,7] = '0')
        then TotalPenaltyCost7 := PenaltyCostSatOnSunOff;
     if (ThreeWeekPattern[i,13]='1') and (ThreeWeekPattern[i,14] = '0')
        then TotalPenaltyCost8 := PenaltyCostSatOnSunOff;
     if (ThreeWeekPattern[i,20]='1') and (ThreeWeekPattern[i,21]='0')
        then TotalPenaltyCost9 := PenaltyCostSatOnSunOff;

     if (ThreeWeekPattern[i,6] = '0') and (ThreeWeekPattern[i,7] = '1')
        then TotalPenaltyCost10 := PenaltyCostSatOffSunOn;
     if (ThreeWeekPattern[i,13]='0') and (ThreeWeekPattern[i,14] = '1')
        then TotalPenaltyCost11 := PenaltyCostSatOffSunOn;
```

```
     if (ThreeWeekPattern[i,20]='0') and (ThreeWeekPattern[i,21]='1')
        then TotalPenaltyCost12 := PenaltyCostSatOffSunOn;

{Penalty Cost for Single day on or off}

     for m := 2 to 20 do begin
         if (ThreeWeekPattern [i,m] = '0') and
            (ThreeWeekPattern [i,m] <> ThreeWeekPattern [i,m+1]) and
            (ThreeWeekPattern [i,m-1] <> ThreeWeekPattern [i,m]) then
            PenaltyCost := PenaltyCostSingle0
         else
            PenaltyCost := 0;

     TotalPenaltyCost2 := TotalPenaltyCost2 + PenaltyCost;
     end;

     for n := 2 to 20 do begin
         if (ThreeWeekPattern [i,n] = '1') and
            (ThreeWeekPattern [i,n] <> ThreeWeekPattern [i,n+1]) and
            (ThreeWeekPattern [i,n-1] <> ThreeWeekPattern [i,n]) then
            PenaltyCost := PenaltyCostSingle1
         else
            PenaltyCost := 0;

     TotalPenaltyCost3 := TotalPenaltyCost3 + PenaltyCost;
     end;

{Penalty Cost for working on eight or more consecutive days}

     j := 1;
     while j <= 21 do begin
           if (ThreeWeekPattern [i,j]='1') and
             (ThreeWeekPattern [i,j]=ThreeWeekPattern[i,(j+1)]) and
             (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+2)]) and
             (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+3)]) and
             (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+4)]) and
             (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+5)]) and
             (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+6)]) and
             (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+7)]) then
             begin PenaltyCost := PenaltyCost8One; j := j + 8; end
           else
             begin PenaltyCost := 0; j :=  j + 1; end;
             TotalPenaltyCost1 := TotalPenaltyCost1 + PenaltyCost;
           end;

{Penalty Cost for working on seven consecutive days}

     if TotalPenaltyCost1 = 0 then begin
        j := 1;
        while j <= 21 do begin
              if (ThreeWeekPattern [i,j]='1') and
              (ThreeWeekPattern [i,j]=ThreeWeekPattern[i,(j+1)]) and
              (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+2)]) and
              (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+3)]) and
              (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+4)]) and
              (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+5)]) and
              (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+6)]) then
              begin PenaltyCost := PenaltyCost7One; j := j + 7; end
              else
              begin PenaltyCost :=0; j :=  j + 1; end;
              TotalPenaltyCost1 := TotalPenaltyCost1 + PenaltyCost;
              end;
        end;

{Penalty Cost for working on six consecutive days}
```

```
      if TotalPenaltyCost1 = 0 then begin
         j := 1;
         while j <= 21 do begin
              if (ThreeWeekPattern [i,j]='1') and
              (ThreeWeekPattern [i,j]=ThreeWeekPattern[i,(j+1)]) and
              (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+2)]) and
              (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+3)]) and
              (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+4)]) and
              (ThreeWeekPattern[i,j]=ThreeWeekPattern[i,(j+5)]) then
              begin PenaltyCost := PenaltyCost6One; j := j + 6; end
              else
              begin PenaltyCost :=0; j :=  j + 1; end;
              TotalPenaltyCost1 := TotalPenaltyCost1 + PenaltyCost;
              end;
          end;

{Penalty Cost for four consecutive days off}

      j := 1;
      while j <= 21 do begin
            if (ThreeWeekPattern [i,j] = '0') and
            (ThreeWeekPattern [i,j] = ThreeWeekPattern [i, j+1]) and
            (ThreeWeekPattern [i,j] = ThreeWeekPattern [i, j+2]) and
            (ThreeWeekPattern [i,j] = ThreeWeekPattern [i, j+3]) then
               begin PenaltyCost := PenaltyCost4Zero; j := j + 4; end
            else
               begin PenaltyCost := 0; J := j + 1; end;
            TotalPenaltyCost0 := TotalPenaltyCost0 + PenaltyCost;
      end;

{Penalty Cost for three consecutive days off}

      if TotalPenaltyCost0 = 0 then begin
         j := 1;
         while j <= 21 do begin
               if (ThreeWeekPattern [i,j] = '0') and
                  (ThreeWeekPattern [i,j] = ThreeWeekPattern [i, j+1]) and
                  (ThreeWeekPattern [i,j] = ThreeWeekPattern [i, j+2]) then
                  begin PenaltyCost := PenaltyCost3Zero; j := j + 3; end
               else
                  begin PenaltyCost := 0; J := j + 1; end;
              TotalPenaltyCost0 := TotalPenaltyCost0 + PenaltyCost;
         end;
      end;
      TotalPenaltyCost[i] := (TotalPenaltyCost0 + TotalPenaltyCost1
                        + TotalPenaltyCost2 + TotalPenaltyCost3
                        + TotalPenaltyCost4 + TotalPenaltyCost5
                        + TotalPenaltyCost6 + TotalPenaltyCost7
                        + TotalPenaltyCost8 + TotalPenaltyCost9
                        + TotalPenaltyCost10 + TotalPenaltyCost11
                        + TotalPenaltyCost12);
      {writeln ('ThreeWeekPattern ',i,' Cost = ',TotalPenaltyCost[i]:4);}
end;
end; {*****************ThreeWeekPenaltyCost*****************************}

{*************************************************************************}

Procedure FourWeekPenaltyCost(var TotalPenaltyCost: TotalPcost);

begin

{FourWeekPatternGeneration(FourWeekPattern);}

for i := 1 to 21 do begin {FourWeekPattern Penalty Cost Calculation}
```

```
        Initialize;

{Penalty Cost for working on WeekEnd}

     if (FourWeekPattern[i,6] = '1') and (FourWeekPattern[i,7] = '1')
        then TotalPenaltyCost4 := PenaltyCostWEndOn;
     if (FourWeekPattern[i,13]='1') and (FourWeekPattern[i,14] = '1')
        then TotalPenaltyCost5 := PenaltyCostWEndOn;
     if (FourWeekPattern[i,20]='1') and (FourWeekPattern[i,21]='1')
        then TotalPenaltyCost6 := PenaltyCostWEndOn;
     if (FourWeekPattern[i,27]='1') and (FourWeekPattern[i,28]='1')
        then TotalPenaltyCost13 := PenaltyCostWEndOn;

     if (FourWeekPattern[i,6] = '1') and (FourWeekPattern[i,7] = '0')
        then TotalPenaltyCost7 := PenaltyCostSatOnSunOff;
     if (FourWeekPattern[i,13]='1') and (FourWeekPattern[i,14] = '0')
        then TotalPenaltyCost8 := PenaltyCostSatOnSunOff;
     if (FourWeekPattern[i,20]='1') and (FourWeekPattern[i,21]='0')
        then TotalPenaltyCost9 := PenaltyCostSatOnSunOff;
     if (FourWeekPattern[i,27]='1') and (FourWeekPattern[i,28]='0')
        then TotalPenaltyCost14 := PenaltyCostSatOnSunOff;

     if (FourWeekPattern[i,6] = '0') and (FourWeekPattern[i,7] = '1')
        then TotalPenaltyCost10 := PenaltyCostSatOffSunOn;
     if (FourWeekPattern[i,13]='0') and (FourWeekPattern[i,14] = '1')
        then TotalPenaltyCost11 := PenaltyCostSatOffSunOn;
     if (FourWeekPattern[i,20]='0') and (FourWeekPattern[i,21]='1')
        then TotalPenaltyCost12 := PenaltyCostSatOffSunOn;
     if (FourWeekPattern[i,27]='0') and (FourWeekPattern[i,28]='1')
        then TotalPenaltyCost15 := PenaltyCostSatOffSunOn;

{Penalty Cost for Single day on or off}


     for m := 2 to 27 do begin
         if (FourWeekPattern [i,m] = '0') and
            (FourWeekPattern [i,m] <> FourWeekPattern [i,m+1]) and
            (FourWeekPattern [i,m-1] <> FourWeekPattern [i,m]) then
            PenaltyCost := PenaltyCostSingle0
         else
            PenaltyCost := 0;

     TotalPenaltyCost2 := TotalPenaltyCost2 + PenaltyCost;
     end;

     for n := 2 to 27 do begin
         if (FourWeekPattern [i,n] = '1') and
            (FourWeekPattern [i,n] <> FourWeekPattern [i,n+1]) and
            (FourWeekPattern [i,n-1] <> FourWeekPattern [i,n]) then
            PenaltyCost := PenaltyCostSingle1
         else
            PenaltyCost := 0;

     TotalPenaltyCost3 := TotalPenaltyCost3 + PenaltyCost;
     end;

{Penalty Cost for working on eight or more consecutive days}

     j := 1;
     while j <= 28 do begin
           if (FourWeekPattern [i,j]='1') and
             (FourWeekPattern [i,j]=FourWeekPattern[i,(j+1)]) and
             (FourWeekPattern[i,j]=FourWeekPattern[i,(j+2)]) and
             (FourWeekPattern[i,j]=FourWeekPattern[i,(j+3)]) and
             (FourWeekPattern[i,j]=FourWeekPattern[i,(j+4)]) and
```

```
            (FourWeekPattern[i,j]=FourWeekPattern[i,(j+5)]) and
            (FourWeekPattern[i,j]=FourWeekPattern[i,(j+6)]) and
            (FourWeekPattern[i,j]=FourWeekPattern[i,(j+7)]) then
            begin PenaltyCost := PenaltyCost8One; j := j + 8; end
          else
            begin PenaltyCost := 0; j :=  j + 1; end;
            TotalPenaltyCost1 := TotalPenaltyCost1 + PenaltyCost;
          end;

{Penalty Cost for working on seven consecutive days}

     if TotalPenaltyCost1 = 0 then begin
        j := 1;
        while j <= 28 do begin
              if (FourWeekPattern [i,j]='1') and
              (FourWeekPattern [i,j]=FourWeekPattern[i,(j+1)]) and
              (FourWeekPattern[i,j]=FourWeekPattern[i,(j+2)]) and
              (FourWeekPattern[i,j]=FourWeekPattern[i,(j+3)]) and
              (FourWeekPattern[i,j]=FourWeekPattern[i,(j+4)]) and
              (FourWeekPattern[i,j]=FourWeekPattern[i,(j+5)]) and
              (FourWeekPattern[i,j]=FourWeekPattern[i,(j+6)]) then
              begin PenaltyCost := PenaltyCost7One; j := j + 7; end
              else
              begin PenaltyCost := 0; j :=  j + 1; end;
              TotalPenaltyCost1 := TotalPenaltyCost1 + PenaltyCost;
              end;
        end;

{Penalty Cost for working on six consecutive days}

     if TotalPenaltyCost1 = 0 then begin
        j := 1;
        while j <= 28 do begin
             if (FourWeekPattern [i,j]='1') and
             (FourWeekPattern [i,j]=FourWeekPattern[i,(j+1)]) and
             (FourWeekPattern[i,j]=FourWeekPattern[i,(j+2)]) and
             (FourWeekPattern[i,j]=FourWeekPattern[i,(j+3)]) and
             (FourWeekPattern[i,j]=FourWeekPattern[i,(j+4)]) and
             (FourWeekPattern[i,j]=FourWeekPattern[i,(j+5)]) then
             begin PenaltyCost := PenaltyCost6One; j := j + 6; end
             else
             begin PenaltyCost :=0; j :=  j + 1; end;
             TotalPenaltyCost1 := TotalPenaltyCost1 + PenaltyCost;
             end;
         end;

{Penalty Cost for four consecutive days off}

     j := 1;
     while j <= 28 do begin
           if (FourWeekPattern [i,j] = '0') and
           (FourWeekPattern [i,j] = FourWeekPattern [i, j+1]) and
           (FourWeekPattern [i,j] = FourWeekPattern [i, j+2]) and
           (FourWeekPattern [i,j] = FourWeekPattern [i, j+3]) then
              begin PenaltyCost := PenaltyCost4Zero; j := j + 4; end
           else
              begin PenaltyCost := 0; J := j + 1; end;
           TotalPenaltyCost0 := TotalPenaltyCost0 + PenaltyCost;
     end;

{Penalty Cost for three consecutive days off}

     if TotalPenaltyCost0 = 0 then begin
        j := 1;
        while j <= 28 do begin
```

```
            if (FourWeekPattern [i,j] = '0') and
               (FourWeekPattern [i,j] = FourWeekPattern [i, j+1]) and
               (FourWeekPattern [i,j] = FourWeekPattern [i, j+2]) then
               begin PenaltyCost := PenaltyCost3Zero; j := j + 3; end
            else
               begin PenaltyCost := 0; J := j + 1; end;
               TotalPenaltyCost0 := TotalPenaltyCost0 + PenaltyCost;
      end;
   end;
   TotalPenaltyCost[i] := (TotalPenaltyCost0 + TotalPenaltyCost1
                         + TotalPenaltyCost2 + TotalPenaltyCost3
                         + TotalPenaltyCost4 + TotalPenaltyCost5
                         + TotalPenaltyCost6 + TotalPenaltyCost7
                         + TotalPenaltyCost8 + TotalPenaltyCost9
                         + TotalPenaltyCost10 + TotalPenaltyCost11
                         + TotalPenaltyCost12 + TotalPenaltyCost13
                         + TotalPenaltyCost14 + TotalPenaltyCost15);

   {writeln ('FourWeekPattern ',i,' Cost = ',TotalPenaltyCost[i]:4);}
end;
end; {********************FourWeekPenaltyCost*****************************}

{*************************************************************************}

end. {Of Unit PntCost}
{*************************************************************************}
```

APPENDIX D
SOURCE CODE OF UNIT FOR SORTING WORK PATTERN

```
Unit SortPat;

interface

uses PntCost, PatGen;


Procedure SortPattern1 (var  NumArray : TotalPCost;

                        Start, Count  : Integer);

Procedure SortPattern2 (var  NumArray : TotalPCost;
                        var  Pattern  : TwoWeekPat;
                        Start, Count  : Integer);
Procedure SortPattern3 (var  NumArray : TotalPCost;
                        var  Pattern  : ThreeWeekPat;
                        Start, Count  : Integer);
Procedure SortPattern4 (var  NumArray : TotalPCost;
                        var  Pattern  : FourWeekPat;
                        Start, Count  : Integer);
implementation


Procedure SortPattern1 (var  NumArray : TotalPCost;

                        Start, Count  : Integer);
var

   J,K         : integer;
   ThisValue   : integer;
   ThisPattern : OneWeek;

begin
     if Count <= 1 then exit;

     for j := (Start + 1) to Count do
         begin
              ThisValue := NumArray[J];
              ThisPattern := OneWeekPattern[J];
              K := J - 1;
              while (ThisValue < NumArray [K]) and
                    (K>0) do
                    begin
                         NumArray[K + 1] := NumArray [K];
                         OneWeekPattern[K + 1] := OneWeekPattern [K];
                         K := K - 1
                    end;
              NumArray[K + 1] := ThisValue;
              OneWeekPattern[K + 1] := ThisPattern;
         end;
end; {******************SortOneWeekPattern*****************************}

{*************************************************************************}

Procedure SortPattern2 (var  NumArray : TotalPCost;
                        var  Pattern  : TwoWeekPat;
```

```
                        Start, Count  : Integer);
var

   J,K          : integer;
   ThisValue    : integer;
   ThisPattern  : TwoWeek;

begin
     if Count <= 1 then exit;

     for j := (Start + 1) to Count do
         begin
              ThisValue := NumArray[J];
              ThisPattern := Pattern[J];
              K := J - 1;
              while (ThisValue < NumArray [K]) and
                    (K>0) do
                    begin
                         NumArray[K + 1] := NumArray [K];
                         Pattern[K + 1] := Pattern [K];
                         K := K - 1
                    end;
              NumArray[K + 1] := ThisValue;
              Pattern[K + 1] := ThisPattern;
         end;
end; {********************SortTwoWeekPattern*****************************}

{*************************************************************************}

Procedure SortPattern3 (var  NumArray : TotalPCost;
                        var  Pattern  : ThreeWeekPat;
                        Start, Count  : Integer);
var

   J,K          : integer;
   ThisValue    : integer;
   ThisPattern  : ThreeWeek;

begin
     if Count <= 1 then exit;

     for j := (Start + 1) to Count do
         begin
              ThisValue := NumArray[J];
              ThisPattern := Pattern[J];
              K := J - 1;
              while (ThisValue < NumArray [K]) and
                    (K>0) do
                    begin
                         NumArray[K + 1] := NumArray [K];
                         Pattern[K + 1] := Pattern [K];
                         K := K - 1
                    end;
              NumArray[K + 1] := ThisValue;
              Pattern[K + 1] := ThisPattern;
         end;
end; {********************SortThreeWeekPattern**************************}

{*************************************************************************}

Procedure SortPattern4 (var  NumArray : TotalPCost;
                        var  Pattern  : FourWeekPat;
                        Start, Count  : Integer);
var
```

```
   J,K         : integer;
   ThisValue   : integer;
   ThisPattern : FourWeek;

begin
     if Count <= 1 then exit;

     for j := (Start + 1) to Count do
         begin
              ThisValue := NumArray[J];
              ThisPattern := Pattern[J];
              K := J - 1;
              while (ThisValue < NumArray [K]) and
                    (K>0) do
                    begin
                         NumArray[K + 1] := NumArray [K];
                         Pattern[K + 1] := Pattern [K];
                         K := K - 1
                    end;
              NumArray[K + 1] := ThisValue;
              Pattern[K + 1] := ThisPattern;
         end;
end; {************************SortFourWeekPattern**********************}
end. {Of Unit SortPat}

{*****************************************************************************}
```

APPENDIX E
SOURCE CODE OF MAIN PROGRAM

```
Program NurseScheduling;
{$M 65520,0,655360}

Uses Crt, Graph, Patgen, PntCost, SortPat;

{ This program generate schedules for nurses in the hospital.              }
{ The program first generate working patterns, then screen the patterns    }
{ based on the constraints invoke by hospital and nurses, using            }
{ *Heuristic Best-First Search Technique.                                  }
{ The patterns will then be combined with the shift patterns to form       }
{ complete schedules for nurses.                                           }


Var

   GraphDriver, GraphMode : Integer;
   xMax, yMax : Integer;

   SeeSchedule,
   SaveSchedule,
   ViewReport,
   SaveReport,
   DefaultValue,
   UserAnswer,
   PickAgain,
   PrintSchedule4,
   PrintSchedule8,
   PrintSchedule12,
   PrintReport4,
   PrintReport8,
   PrintReport12,
   Display8Staff,
   Display12Staff,
   EightWeek          : Char;
   PatternBase,
   Schedule1,
   SchedOut,
   SchedOutTwo,
   SchedOutThree,
   ShiftBase,
   EightWeekBase,
   EightBase,
   TwelveBase,
   TwelveWeekBase,
   ShiftBase1,
   ShiftBase2,
   ShiftBase3,
   ShiftSecondBase,
   ShiftThirdBase,
   Shift8WBase,
   Shift12WBase,
   StafList4,
   StafList8,
   StafList12,
   Print4Schedule,
   Print12Schedule,
   PrintStaff4,
   PrintStaff8,
```

```
    PrintStaff12,
    PrintEight          : Text;
    ScheduleNum         : Integer;

    ShiftPat            : Array[1..10000] of Char;
    Shift               : Array [1..10000] of Char;
    ShiftPattern        : Array [1..5000] of String[4];
    FourWkPattern,
    ShPattern,
    NewPattern,
    NewShift            : String;
    i,j,k,l,m,n,
    p,q,r,s             : integer;
    x1, x2, x3          : integer; {Shift Ratio:                          }
                                   {x1 : Day shift proportion             }
                                   {x2 : Evening Shift proportion         }
                                   {x3 : Night Shift proportion           }
    DayRatio, EveRatio,
    NiteRatio           : real;    {Ideal distribution of nurse based on  }
                                   {shift policy                          }

    TotalPenaltyCost    : TotalPcost;

    NOfPat,
    NumOfPat,
    NumOfShift,
    MaxPcost,
    LengthOfSchedule            : integer;

{*************************************************************************}

Procedure CreatePattern;

begin

Assign (PatternBase,'BASE.DAT'); {Prepare a file for Pattern Base}
Rewrite (PatternBase);

NOfPat := 1;  { Initialize counter to check how many patterns have been }
              { generated }

OneWeekPenaltyCost(TotalPenaltyCost);
SortPattern1(TotalPenaltyCost, 1, 21);

q := 1;
repeat        {*********************TwoWeekPattern***********************}

TwoWeekPatternGeneration(TwoWeekPattern,q);
TwoWeekPenaltyCost(TotalPenaltyCost);
SortPattern2(TotalPenaltyCost, TwoWeekPattern, 1, 21);

r := 1;
repeat       {*********************ThreeWeekPattern**********************}

ThreeWeekPatternGeneration(ThreeWeekPattern,r);
ThreeWeekPenaltyCost(TotalPenaltyCost);
SortPattern3(TotalPenaltyCost, ThreeWeekPattern, 1, 21);

s := 1;
repeat       {*********************FourWeekPattern**********************}

FourWeekPatternGeneration(FourWeekPattern,s);
FourWeekPenaltyCost(TotalPenaltyCost);
SortPattern4(TotalPenaltyCost, FourWeekPattern, 1, 21);
```

```
for i := 1 to 21 do
    if TotalPenaltyCost[i] <= MaxPCost then begin
    writeln(PatternBase, FourWeekPattern[i]);

    NOfPat := NOfPat + 1;
    end;

s := s + 1;
until (NOfPat >= NumOfPat) or (s = 21);

            {**********************FourWeekPattern**********************}

r := r + 1;
until (NOfPat >= NumOfPat) or (r = 21);

            {**********************ThreeWeekPattern**********************}

q := q + 1;
until (NOfPat >= NumOfPat) or (q = 21);

           {**********************TwoWeekPattern**********************}

Close (PatternBase);

end; {************* Procedure CreatePattern ************** }

{*********************************************************************}

Procedure PickFourWeekPattern;

type
    FWP = String[28];
var
    Start,
    Sched,
    Count      : integer;
    FourWkPat : Array [1..1000] of FWP;
begin

    Randomize;
    Assign(SchedOut,'SCHDBAS.DAT');
    Rewrite(SchedOut);

    Reset (PatternBase);

    For Start := 1 to NumOfPat do
        Readln (PatternBase, FourWkPat [Start]);

    For Sched := 1 to ScheduleNum do

        begin

             Count := Random (NumOfPat);
             If Count = 0 then
                Count := Count + 1;
             Writeln (SchedOut, FourWkPat [Count]);

        end;
    Close(SchedOut);
    Close(PatternBase);

end; {Procedure PickFourWeekPattern}

{*********************************************************************}
```

```
Procedure PickSecondPattern;

type
    FWP = String[28];
var
    Start,
    Sched,
    Count      : integer;
    FourWkPat : Array [1..1000] of FWP;
begin

    Randomize;
    Assign(SchedOutTwo,'SCHDBAS1.DAT');
    Rewrite(SchedOutTwo);

    Reset (PatternBase);

    For Start := 1 to NumOfPat do
        Readln (PatternBase, FourWkPat [Start]);

    For Sched := 1 to ScheduleNum do

        begin

             Count := Random (NumOfPat);
             If Count = 0 then
                Count := Count + 1;
             Writeln (SchedOutTwo, FourWkPat [Count]);

        end;
    Close(SchedOutTwo);
    Close(PatternBase);

end; {Procedure PickSecondPattern}

{*************************************************************************}

Procedure PickThirdPattern;

type
    FWP = String[28];
var
    Start,
    Sched,
    Count      : integer;
    FourWkPat : Array [1..1000] of FWP;
begin

    Randomize;
    Assign(SchedOutThree,'SCHDBAS2.DAT');
    Rewrite(SchedOutThree);

    Reset (PatternBase);

    For Start := 1 to NumOfPat do
        Readln (PatternBase, FourWkPat [Start]);

    For Sched := 1 to ScheduleNum do

        begin

             Count := Random (NumOfPat);
             If Count = 0 then
                Count := Count + 1;
             Writeln (SchedOutThree, FourWkPat [Count]);
```

```
        end;
    Close(SchedOutThree);
    Close(PatternBase);

end; {Procedure PickThirdPattern}

{*************************************************************************}

Procedure EightWeekPattern;
{$M 65520,0,655360}

{Type

    FWP = String[28];}


Var

    FourWkPat   : String[28];
    FWP         : String[28];
    EightWkPat  : String[56];
    MaxSched,
    jj,kk,ll : integer;

begin

PickSecondPattern;

Assign (EightWeekBase, 'SCHD8WK.DAT');
Rewrite(EightWeekBase);

Reset (SchedOut);
Reset (SchedOutTwo);

While Not Eof (SchedOut) or Not Eof (SchedOutTwo) do

    begin
         readln(SchedOut, FourWkPat);
         readln(SchedOutTwo,FWP);
         EightWkPat := FourWkPat + FWP;
         Writeln (EightWeekBase,EightWkPat);
    end;

    Close(SchedOut);
    Close(SchedOutTwo);
    Close(EightWeekBase);

end; {Procedure EightWeek Pattern}

{*************************************************************************}

Procedure TwelveWeekPattern;
{$M 65520,0,655360}

{Type

    FWP = String[28];}


Var

    FourWkPat   : String[28];
    FWP         : String[28];
```

```
    FWP2        : String[28];
    TwelveWkPat : String[84];
    MaxSched,
    jj,kk,ll : integer;

begin

PickSecondPattern;
PickThirdPattern;

Assign (TwelveWeekBase, 'SCHD12WK.DAT');
Rewrite(TwelveWeekBase);

Reset (SchedOut);
Reset (SchedOutTwo);
Reset (SchedOutThree);

While Not Eof (SchedOut) or Not Eof (SchedOutTwo) or
      Not Eof (SchedOutThree) do

    begin
         readln(SchedOut, FourWkPat);
         readln(SchedOutTwo,FWP);
         readln(SchedOutThree,FWP2);
         TwelveWkPat := FourWkPat + FWP + FWP2;
         Writeln (TwelveWeekBase,TwelveWkPat);
    end;

    Close(SchedOut);
    Close(SchedOutTwo);
    Close(SChedOutThree);
    Close(TwelveWeekBase);

end; {Procedure TwelveWeekPattern}

{*************************************************************************}

Procedure CreateShiftBase;

var

   ShiftRatio : integer;
   x1, x2, x3   : integer;       {Shift Ratio}
                                 {x1 : Day shift proportion}
                                 {x2 : Evening Shift proportion}
                                 {x3 : Night Shift proportion}
{-------------------------------------------------------------------------}

Procedure ShiftPolicy; {This procedure give the user the opportunity      }
                       {to select the shift policy for the hospital       }
                       {There are 5 policies listed here, but can         }
                       {be extended if necessary.                         }
var
   Proportion : real;
   TotalRatio : real;
   r1,r2,r3   : real;

begin

     ClrScr;
     writeln;
     writeln( '       Select Shift Policy:                        ');
     writeln;
     writeln;
     writeln;
```

```
writeln;
writeln( '                              1. Shift Policy 1:1:1 ');
writeln( '                              2. Shift Policy 2:1:1 ');
writeln( '                              3. Shift Policy 2:2:1 ');
writeln( '                              4. Shift Policy 3:2:1 ');
writeln( '                              5. Shift Policy 3:2:2 ');
writeln( '                              6. Shift Policy 3:3:1 ');
writeln( '                              7. Shift Policy 3:3:2 ');
writeln( '                              8. Shift Policy 4:3:1 ');
writeln( '                              9. Shift Policy 4:3:2 ');
writeln( '                             10. Other              ');
writeln;
writeln;
writeln;
writeln;
write( '       Choose the number and press <Enter> :  ');
readln(ShiftRatio);

NumOfShift := 4 * NumOfPat;

Case ShiftRatio of

     1: begin
          x1 := Round ((1/3) * NumOfShift);
          x2 := Round ((1/3) * NumOfShift);
          x3 := (NumOfShift - (x1 + x2));
          DayRatio  := 0.333;
          EveRatio  := 0.333;
          NiteRatio := 0.333;
        end;
     2: begin
          x1 := Round ((2/4) * NumofShift);
          x2 := Round ((1/4) * NumofShift);
          x3 := (NumOfShift - (x1 + x2));
          DayRatio  := 0.500;
          EveRatio  := 0.250;
          NiteRatio := 0.250;
        end;
     3: begin
          x1 := Round ((2/5) * NumOfShift);
          x2 := Round ((2/5) * NumOfShift);
          x3 := (NumOfShift - (x1 + x2));
          DayRatio  := 0.400;
          EveRatio  := 0.400;
          NiteRatio := 0.200;
        end;
     4: begin
          x1 := Round ((3/6) * NumOfShift);
          x2 := Round ((2/6) * NumOfShift);
          x3 := (NumOfShift - (x1 + x2));
          DayRatio  := 0.500;
          EveRatio  := 0.333;
          NiteRatio := 0.167;
        end;
     5: begin
          x1 := Round ((3/7) * NumOfShift);
          x2 := Round ((2/7) * NumOfShift);
          x3 := (NumOfShift - (x1 + x2));
          DayRatio  := 0.429;
          EveRatio  := 0.286;
          NiteRatio := 0.285;
        end;
     6: begin
          x1 := Round ((3/7) * NumOfShift);
          x2 := Round ((3/7) * NumOfShift);
```

```
                x3 := (NumOfShift - (x1 + x2));
                DayRatio  := 0.429;
                EveRatio  := 0.429;
                NiteRatio := 0.142;
            end;
         7: begin
                x1 := Round ((3/8) * NumOfShift);
                x2 := Round ((3/8) * NumOfShift);
                x3 := (NumOfShift - (x1 + x2));
                DayRatio  := 0.375;
                EveRatio  := 0.375;
                NiteRatio := 0.250;
            end;
         8: begin
                x1 := Round ((4/8) * NumOfShift);
                x2 := Round ((3/8) * NumOfShift);
                x3 := (NumOfShift - (x1 + x2));
                DayRatio  := 0.500;
                EveRatio  := 0.375;
                NiteRatio := 0.125;
            end;
         9: begin
                x1 := Round ((4/9) * NumOfShift);
                x2 := Round ((3/9) * NumOfShift);
                x3 := (NumOfShift - (x1 + x2));
                DayRatio  := 0.445;
                EveRatio  := 0.333;
                NiteRatio := 0.222;
            end;
        10: begin
                ClrScr;
                writeln;
                writeln('       Select Shift Policy: ');
                writeln;
                writeln;
                write('       Day Ratio     :  ');
                readln(r1);
                writeln;
                write('       Evening Ratio :  ');
                readln(r2);
                writeln;
                write('       Night Ratio   :  ');
                readln(r3);
                TotalRatio  := r1 + r2 + r3;
                Proportion  := 1/TotalRatio;

                x1 := Round ((r1/TotalRatio) * NumOfShift);
                x2 := Round ((r2/TotalRatio) * NumOfShift);
                x3 := (NumOfShift - (x1 + x2));
                DayRatio    := (r1 * Proportion);
                EveRatio    := (r2 * Proportion);
                NiteRatio   := (r3 * Proportion);
            end;
         else
         writeln ('        Error in entry, try again !');

    end; {of Case}

end;{of Procedure ShiftPolicy}

{-------------------------------------------------------------------------}

begin
```

```
Assign (ShiftBase1,'SHIFPAT.DAT');
Rewrite(ShiftBase1);


{NumOfShift := 4 * NumOfPat;}

ShiftPolicy;

     for i := 1 to x1 do
         begin
         ShiftPat[i] := 'D';
         writeln(ShiftBase1, ShiftPat[i]);
         end;
     for i:= (x1 + 1) to (x1 + x2) do
         begin
         ShiftPat[i] := 'E';
         writeln(ShiftBase1, ShiftPat[i]);
         end;
     for i := (x1 + x2 + 1) to (x1 + x2 + x3) do
         begin
         ShiftPat[i] := 'N';
         writeln(ShiftBase1, ShiftPat[i]);
         end;

Close(ShiftBase1);

end; {Procedure CreateShiftBase}

{*************************************************************************}

Procedure ReOrderShift;

var
   ShiftPat2 : Array [1..10000] of char;
begin

Assign(ShiftBase2, 'NEWSHIFT.DAT');
Rewrite(ShiftBase2);

Reset(ShiftBase1);

Randomize;

for i := 1 to NumOfShift do
    begin
         k := random(NumOfShift);
         if k=0 then k := k + 1;
         readln(ShiftBase1,ShiftPat[k]);
         ShiftPat2[i] := ShiftPat[k];
         writeln(ShiftBase2, ShiftPat2[i]);
    end;

Close(ShiftBase1);
Close(ShiftBase2);

end; {Procedure ReOrderShift}

{*************************************************************************}

Procedure CreateShift;

begin

Assign(ShiftBase3,'SHIFBASE.DAT');
Rewrite(ShiftBase3);
```

```
Reset(ShiftBase2);

Randomize;
i := 0;

repeat

      for j := 1 to 4 do
          begin
               k := random(NumOfShift);
               if k = 0 then k := k + 1;
               Shift[j] := ShiftPat[k];
          end;

      If Not
         (((Shift[1] = 'E') and
         ((Shift[1] = Shift[2]) and (Shift[1] = Shift[3]) and
         (Shift[1] = Shift[4]))) or
         ((Shift[1] = 'E') and
         ((Shift[1] = Shift[2]) and (Shift[1] = Shift[3]))) or
         ((Shift[1] = 'E') and
         ((Shift[1] = Shift[3]) and (Shift[1] = Shift[4]))) or
         ((Shift[1] = 'E') and
         ((Shift[1] = Shift[2]) and (Shift[1] = Shift[4]))) or
         ((Shift[2] = 'E') and
         ((Shift[2] = Shift[3]) and (Shift[2] = Shift[4]))) or
         ((Shift[1] = 'N') and
         ((Shift[1] = Shift[2]) and (Shift[1] = Shift[3])) and
         ((Shift[1] = Shift[4]))) or
         ((Shift[1] = 'N') and
         ((Shift[1] = Shift[2]) and (Shift[1] = Shift[3]))) or
         ((Shift[1] = 'N') and
         ((Shift[1] = Shift[3]) and (Shift[1] = Shift[4]))) or
         ((Shift[1] = 'N') and
         ((Shift[1] = Shift[2]) and (Shift[1] = Shift[4]))) or
         ((Shift[2] = 'N') and
         ((Shift[2] = Shift[3]) and (Shift[2] = Shift[4])))) then
      begin
           ShiftPattern[i] := Shift[1] + Shift[2] + Shift[3] + Shift[4];
           writeln(ShiftBase3, ShiftPattern[i]);
           i := i + 1;
      end;

Until i = NumOfPat;

Close(ShiftBase2);
Close(ShiftBase3);

end; { Procedure CreateShift }

{*************************************************************************}

Procedure CreateSecondShift;

Var
    ii,jj,kk          : integer;
    SecondShift       : Array [1..5000] of String[4];

begin

{CreateShiftBase;
ReOrderShift;
CreateShift;}

Assign (ShiftSecondBase,'SH2BASE.DAT');
```

```
Rewrite(ShiftSecondBase);

Reset (ShiftBase3);

For ii := 1 to NumOfPat do
    Readln(ShiftBase3, ShiftPattern [ii]);

Randomize;

For kk := 1 to NumOfPat do
    begin
         jj := Random (NumOfPat);
         If jj = 0 then
            jj := jj + 1;
         SecondShift[kk] := ShiftPattern[jj];
         Writeln(ShiftSecondBase, SecondShift[kk]);
    end;

Close(ShiftSecondBase);
Close(ShiftBase3);

end; { Procedure CreateSecondShift }

{*************************************************************************}

Procedure CreateThirdShift;

Var
    ii,jj,kk    : integer;
    SecondShift,
    ThirdShift      : Array [1..1000] of String[4];

begin

Assign (ShiftThirdBase,'SH3BASE.DAT');
Rewrite(ShiftThirdBase);

Reset(ShiftSecondBase);

For ii := 1 to NumOfPat do
    Readln(ShiftSecondBase, SecondShift [ii]);

Randomize;

For kk := 1 to NumOfPat do
    begin
         jj := Random (NumOfPat);
         If jj = 0 then
            jj := jj + 1;
         ThirdShift[kk] := SecondShift[jj];
         Writeln(ShiftThirdBase, ThirdShift [kk]);
    end;

Close(ShiftSecondBase);
Close(ShiftThirdBase);

end; { Procedure CreateThirdShift }

{*************************************************************************}

Procedure Create8WeekShift;

Var
   Shift4 : String[4];
   Shift4Too: String[4];
```

```
   Shift8 : String[8];

begin

Assign(Shift8WBase,'SH8BASE.DAT');
Rewrite(Shift8WBase);

Reset(ShiftBase3);
Reset(ShiftSecondBase);

While Not Eof (ShiftBase3) and Not Eof(ShiftSecondBase) do

      begin
           readln(ShiftBase3,Shift4);
           readln(ShiftSecondBase,Shift4Too);
           Shift8 := Shift4 + Shift4Too;
           Writeln(Shift8WBase,Shift8);
      end;

Close(Shift8WBase);
Close(ShiftBase3);
Close(ShiftSecondBase);

end; { Procedure Create8WeekShift }

{*************************************************************************}

Procedure Create12WeekShift;

Var
   Shift4 : String[4];
   Shift4Too: String[4];
   Shift4Three : String[4];
   Shift12  : String[12];

begin


Assign(Shift12WBase,'SH12BASE.DAT');
Rewrite(Shift12WBase);

Reset(ShiftBase3);
Reset(ShiftSecondBase);
Reset(ShiftThirdBase);

While Not Eof (ShiftBase3) and Not Eof(ShiftSecondBase) and
      Not Eof (ShiftThirdBase) do

      begin
           readln(ShiftBase3,Shift4);
           readln(ShiftSecondBase,Shift4Too);
           readln(ShiftThirdBase,Shift4Three);
           Shift12 := Shift4 + Shift4Too + Shift4Three;
           Writeln(Shift12WBase,Shift12);
      end;

Close(Shift12WBase);
Close(ShiftBase3);
Close(ShiftSecondBase);
Close(ShiftThirdBase);

end; { Procedure Create12WeekShift }

{*************************************************************************}
```

```
Procedure ScreenView;

Type

    FWPT = String[28];
    SPat = String[4];

Var

    FWeekPat : Array [1..1000] of FWPT;
    SPT      : Array [1..1000] of SPat;
    jj, kk,
    ll, s    : integer;

begin

Reset (SchedOut);
Reset (ShiftBase3);
    i := 0;
    for jj := 1 to ScheduleNum do
        begin
            writeln;
            writeln('                          Schedule No. : ',jj:4);
            writeln('                          ----------------------',
                    '--------');
            m := 1;
            readln(SchedOut, FWeekPat[jj]);
            readln(ShiftBase3, SPT[jj]);
            writeln('                          Day  :         ',
                    '  M T W T F S S');
            writeln('                          --------------',
                    '----------------');
               for kk := 1 to 4 do begin
                   write('                              Week :   ',kk,
                         '   ',SPT[jj,kk],':  ');
                      for ll := 1 to 7 do begin
                         write(FWeekPat[jj,m],' ');
                         m := m + 1;
                      end;
               writeln;
               end;
            writeln('                          ==================',
                    '=============');
        i := i + 1;
        if i = 2 then
        begin
        i := 0;
        writeln;
     writeln('Press any key to continue or <Esc> to cancel viewing ... ');

        if ReadKey = #27 then Exit
        else
        ClrScr;

        end;
        end; {for jj}

    Close(SchedOut);
    Close(ShiftBase3);

end; {Procedure ScreenView}

{*************************************************************************}

Procedure PrintSchedule;
```

```
Type

    FWPT = String[28];
    SPat = String[4];

Var

    FWeekPat : Array [1..1000] of FWPT;
    SPT      : Array [1..1000] of SPat;
    jj, kk,
    ll, s    : integer;

begin

Assign(Print4Schedule, 'LPT1');
Rewrite(Print4Schedule);

Reset (SchedOut);
Reset (ShiftBase3);

    for jj := 1 to ScheduleNum do
        begin
            writeln(Print4Schedule);
            writeln(Print4Schedule,
                    '                                   Schedule No. : ',jj:4);
            writeln(Print4Schedule,
                    '                                   ----------------------',
                    '--------');
            m := 1;
            readln(SchedOut, FWeekPat[jj]);
            readln(ShiftBase3, SPT[jj]);
            writeln(Print4Schedule,
                    '                                   Day  :          ',
                    '  M T W T F S S');
            writeln(Print4Schedule,
                    '                                   --------------',
                    '----------------');
               for kk := 1 to 4 do begin
                   write(Print4Schedule,
                         '                                        Week :    ',kk,
                         '    ',SPT[jj,kk],':  ');
                     for ll := 1 to 7 do begin
                        write(Print4Schedule,FWeekPat[jj,m],' ');
                        m := m + 1;
                      end;
               writeln(Print4Schedule);
               end;
            writeln(Print4Schedule,
                    '                                   ==================',
                    '=============');

        end; {for jj}

    Close(SchedOut);
    Close(ShiftBase3);
    Close(Print4Schedule);
end; {Procedure Print4Schedule}

{*************************************************************************}

Procedure SaveTheSchedule;

Type

    FWPT = String[28];
```

```
    SPat = String[4];

Var

    FWeekPat : Array [1..1000] of FWPT;
    SPT      : Array [1..1000] of SPat;
    jj, kk,
    ll       : integer;

begin

Reset (SchedOut);
Reset (ShiftBase3);

Assign ( Schedule1, 'SCHEDULE.DAT');
Rewrite ( Schedule1);

    for jj := 1 to ScheduleNum do
        begin
            writeln(Schedule1);
            writeln(Schedule1,'Schedule No. : ',jj:4);
            writeln(Schedule1,'-----------------------------');
            m := 1;
            readln(SchedOut, FWeekPat[jj]);
            readln(ShiftBase3, SPT[jj]);

            writeln(Schedule1,'Day  :         ',' M T W T F S S');
            writeln(Schedule1,'-----------------------------');
               for kk := 1 to 4 do begin
                   write(Schedule1,'Week :   ',kk,'   ',SPT[jj,kk],':  ');
                      for ll :=1 to 7 do begin
                         write(Schedule1,FWeekPat[jj,m],' ');
                         m := m + 1;
                      end;
               writeln(Schedule1);
               end;

        writeln(Schedule1,'=============================');
        end; {for j}

Close (SchedOut);
Close (ShiftBase3);
Close(Schedule1);
end; {Procedure SaveTheSchedule}

{*************************************************************************}

Procedure ViewEightSchedule;

Type

    SPat = String[4];

Var

    EWB : Text;
    EightWkPat    : Char;
    Shift8        : Char;
    jj,kk,ll      : integer;


begin

{Assign ( EightBase, 'BASE8.DAT');}
{Rewrite ( EightBase);}
```

```
Reset(EightWeekBase);
Reset(Shift8WBase);

i := 0;

While Not Eof(EightWeekBase) do
  begin
  for jj := 1 to ScheduleNum do begin

       writeln;
       writeln('                          Schedule No. : ',jj:4);
       writeln('                          ------------------------------');
       writeln('                          Day  :         ',' M T W T F S S ');
       writeln('                          ------------------------------');

       While Not Eoln(EightWeekBase) do
       begin
            for kk := 1 to 8 do begin
            read(Shift8WBase,Shift8);
            write('                             Week :   ',kk,'  ',Shift8);
            write('    ');
                for ll := 1 to 7 do begin
                    read(EightWeekBase, EightWkPat);
                    write(EightWkPat,' ');
                end;
            writeln;
            end;
       end;
       Readln(EightWeekBase);
       Readln(Shift8WBase);
       writeln('                          ==============================');
       i := i + 1;
       if i = 1 then
       begin
       i := 0;
       writeln;
     writeln('Press any key to continue or <Esc> to cancel viewing ... ');

       if ReadKey = #27 then Exit
       else
       ClrScr;

       end;
       end;
end;

Close (EightWeekBase);
Close(Shift8WBase);

end; { Procedure ViewEightSchedule }

{*************************************************************************}

Procedure PrintEightSchedule;

Type

    SPat = String[4];

Var

    EWB : Text;
    EightWkPat    : Char;
    Shift8        : Char;
```

```
    jj,kk,ll       : integer;

begin

Assign ( PrintEight, 'LPT1');
Rewrite ( PrintEight);


Reset(EightWeekBase);
Reset(Shift8WBase);

i := 0;

While Not Eof(EightWeekBase) do
  begin
  for jj := 1 to ScheduleNum do begin

       writeln(PrintEight);
       writeln(PrintEight,'                                Schedule No. : ',jj:4);
       writeln(PrintEight,'                                ----------------------',
               '-------');
       writeln(PrintEight,'                                Day  :          ',
               '  M T W T F S S ');
       writeln(PrintEight,'                                ----------------------',
               '-------');

       While Not Eoln(EightWeekBase) do
       begin
            for kk := 1 to 8 do begin
            read(Shift8WBase,Shift8);
            write(PrintEight,'                                   Week :    ',
                  kk,'  ',Shift8);
            write(PrintEight,'     ');
                for ll := 1 to 7 do begin
                    read(EightWeekBase, EightWkPat);
                    write(PrintEight,EightWkPat,' ');
                end;
            writeln(PrintEight);
            end;
       end;
       Readln(EightWeekBase);
       Readln(Shift8WBase);
       writeln(PrintEight,'                               ======================',
               '========');
       end;
end;

Close (EightWeekBase);
Close(Shift8WBase);

end; { Procedure PrintEightSchedule }

{*************************************************************************}


Procedure SaveEightSchedule;

Type

    SPat = String[4];

Var

    EWB : Text;
```

```
    EightWkPat      : Char;
    Shift8          : Char;
    jj,kk,ll        : integer;


begin

Assign ( EightBase, 'BASE8.DAT');
Rewrite ( EightBase);


Reset(EightWeekBase);
Reset(Shift8WBase);

While Not Eof(EightWeekBase) do
  begin
  for jj := 1 to ScheduleNum do begin

       writeln(EightBase);
       writeln(EightBase,'Schedule No. : ',jj:4);
       writeln(EightBase,'-----------------------------');
       writeln(EightBase,'Day  :         ',' M T W T F S S ');
       writeln(EightBase,'-----------------------------');

       While Not Eoln(EightWeekBase) do
       begin
            for kk := 1 to 8 do begin
            read(Shift8WBase,Shift8);
            write(EightBase,'Week :   ',kk,'  ',Shift8);
            write(EightBase,'    ');
                for ll := 1 to 7 do begin
                    read(EightWeekBase, EightWkPat);
                    write(EightBase,EightWkPat,' ');
                end;
            writeln(EightBase);
            end;
       end;
       Readln(EightWeekBase);
       Readln(Shift8WBase);
       writeln(EightBase,'=============================');
       end;
end;

Close (EightWeekBase);
Close(Shift8WBase);
Close(EightBase);

end; { Procedure SaveEightSchedule }

{*************************************************************************}

Procedure ViewTwelveSchedule;

Type

    SPat = String[4];

Var

    TWB             : Text;
    TwelveWkPat     : Char;

    Shift12         : Char;
    jj,kk,ll        : integer;
```

```
begin

Assign (TWB,'SCHD12WK.DAT');

Reset(TWB);

Reset(Shift12WBase);
i := 0;

While Not Eof(TWB) do
  begin
  for jj := 1 to ScheduleNum do begin

       writeln;
       writeln('                    Schedule No. : ',jj:6);
       writeln('                    ------------------------------');
       writeln('                    Day  :         ',' M T W T F S S ');
       writeln('                    ------------------------------');

       While Not Eoln(TWB) do
       begin
            for kk := 1 to 12 do begin
            read(Shift12WBase,Shift12);
            write('                      Week :  ',kk:2,'  ',Shift12);
            write('    ');
                for ll := 1 to 7 do begin
                    read(TWB, TwelveWkPat);
                    write(TwelveWkPat,' ');
                end;
            writeln;
            end;
       end;
       Readln(TWB);
       Readln(Shift12WBase);
       writeln('                    ==============================');
       i := i + 1;
       if i = 1 then
       begin
       i := 0;
       writeln;
     writeln('Press any key to continue or <Esc> to cancel viewing ... ');

       if ReadKey = #27 then Exit
       else
       ClrScr;

       end;
       end;
end;

Close (TWB);
Close(Shift12WBase);

end; { Procedure ViewTwelveSchedule }

{*************************************************************************}

Procedure PrintTwelveSchedule;

Type

    SPat = String[4];

Var
```

```
    TWB             : Text;
    TwelveWkPat     : Char;

    Shift12         : Char;
    jj,kk,ll        : integer;


begin

Assign(Print12Schedule, 'LPT1');
Rewrite(Print12Schedule);

Assign (TWB,'SCHD12WK.DAT');

Reset(TWB);

Reset(Shift12WBase);

While Not Eof(TWB) do
  begin
  for jj := 1 to ScheduleNum do begin

       writeln(Print12Schedule);
       writeln(Print12Schedule,'     Schedule No. : ',jj:4);
       writeln(Print12Schedule,'     ------------------------------');
       writeln(Print12Schedule,'     Day  :         ',' M T W T F S S ');
       writeln(Print12Schedule,'     ------------------------------');

       While Not Eoln(TWB) do
       begin
            for kk := 1 to 12 do begin
            read(Shift12WBase,Shift12);
            write(Print12Schedule,'     Week :  ',kk:2,'  ',Shift12);
            write(Print12Schedule,'    ');
                for ll := 1 to 7 do begin
                    read(TWB, TwelveWkPat);
                    write(Print12Schedule,TwelveWkPat,' ');
                end;
            writeln(Print12Schedule);
            end;
       end;
       Readln(TWB);
       Readln(Shift12WBase);
       writeln(Print12Schedule,'     ==============================');
       end;
end;

Close (TWB);
Close(Shift12WBase);
Close(Print12Schedule);

end; { Procedure PrintTwelveSchedule }

{*************************************************************************}


Procedure SaveTwelveSchedule;

Type

    SPat = String[4];

Var

    TWB             : Text;
```

```
    TwelveWkPat     : Char;

    Shift12         : Char;
    jj,kk,ll        : integer;


begin

Assign ( TwelveBase, 'BASE12.DAT');
Rewrite ( TwelveBase);

Assign (TWB,'SCHD12WK.DAT');

Reset(TWB);

Reset(Shift12WBase);

While Not Eof(TWB) do
  begin
  for jj := 1 to ScheduleNum do begin

       writeln(TwelveBase);
       writeln(TwelveBase,'Schedule No. : ',jj:4);
       writeln(TwelveBase,'------------------------------');
       writeln(TwelveBase,'Day  :          ',' M T W T F S S ');
       writeln(TwelveBase,'------------------------------');

       While Not Eoln(TWB) do
       begin
            for kk := 1 to 12 do begin
            read(Shift12WBase,Shift12);
            write(TwelveBase,'Week :  ',kk:2,'  ',Shift12);
            write(TwelveBase,'    ');
                for ll := 1 to 7 do begin
                    read(TWB, TwelveWkPat);
                    write(TwelveBase, TwelveWkPat,' ');
                end;
            writeln(TwelveBase);
            end;
       end;
       Readln(TWB);
       Readln(Shift12WBase);
       writeln(TwelveBase,'==============================');
       end;
end;

Close (TWB);
Close(Shift12WBase);
Close(TwelveBase);

end; { Procedure SaveTwelveSchedule }

{*************************************************************************}

Procedure ViewStaffList4Week;

Var
   CountD,
   CountE,
   CountN                  : Array[1..28] of Longint;
   Shift4,
   Pattern4                : Char;
   i,j,k,l                 : integer;
   TotalD, TotalE, TotalN  : Longint;
   AveD, AveE, AveN        : real;
```

```
   RatioD, RatioE, RatioN : real;

Procedure InitCount;
var
   m : integer;
begin
     for m := 1 to 28 do
     begin
         CountD[m] := 0;
         CountE[m] := 0;
         CountN[m] := 0;
     end;
end;

begin

InitCount;

Reset(SchedOut);
Reset(ShiftBase3);

While Not Eof(SchedOut) and Not Eof(ShiftBase3) do
begin
     j := 1;
     While Not Eoln(ShiftBase3) do
     begin
          read(ShiftBase3,Shift4);
          for i:=j to j + 6 do
          begin
               read(Schedout, Pattern4);
               if (Shift4 = 'D') and (Pattern4 = '1') then
                  CountD[i] := CountD[i] + 1
               else
               if (Shift4 = 'E') and (Pattern4 = '1') then
                  CountE[i] := CountE[i] + 1
               else
               if (Shift4 = 'N') and (Pattern4 = '1') then
                  CountN[i] := CountN[i] + 1;
          end;
          j := j + 7;
     end;
readln (ShiftBase3);
readln (SchedOut);
end;

Close(SchedOut);
Close(ShiftBase3);

writeln('       STAFF LIST REPORT:                                          ');
writeln('                                                                   ');
writeln;
writeln('     ------------------------------------------------------------',
        '----------');
writeln('       Day :              ','   Day-Shift',
        '    ',' Evening-Shift','    ',' Night-Shift');
writeln('     ------------------------------------------------------------',
        '----------');
TotalD := 0;
TotalE := 0;
TotalN := 0;
k := 0;
for l := 1 to 28 do
begin
    if (l=1) or (l=8) or (l=15) or (l=22) then
    write('       Monday   ');
```

```
    if (l=2) or (l=9) or (l=16) or (l=23) then
    write('       Tuesday  ');
    if (l=3) or (l=10) or (l=17) or (l=24) then
    write('       Wednesday');
    if (l=4) or (l=11) or (l=18) or (l=25) then
    write('       Thursday ');
    if (l=5) or (l=12) or (l=19) or (l=26) then
    write('       Friday   ');
    if (l=6) or (l=13) or (l=20) or (l=27) then
    write('       Saturday ');
    if (l=7) or (l=14) or (l=21) or (l=28) then
    write('       Sunday   ');

    writeln('     ',l:2,'    ',CountD[l]:8,'       ',CountE[l]:8,
            '         ',CountN[l]:8);

    TotalD := TotalD + CountD[l];
    TotalE := TotalE + CountE[l];
    TotalN := TotalN + CountN[l];
    k := k + 1;
    if k = 17 then
    begin
    writeln('Press <Enter> to continue ... ');
    readln;
    end

end;

    AveD := (TotalD / 28);
    AveE := (TotalE / 28);
    AveN := (TotalN / 28);
    RatioD := (TotalD) / (TotalD + TotalE + TotalN);
    RatioE := (TotalE) / (TotalD + TotalE + TotalN);
    RatioN := (TotalN) / (TotalD + TotalE + TotalN);

writeln('     ------------------------------------------------------------',
        '----------');
writeln('       Total   ','            ',TotalD:8,'       ',TotalE:8,
        '         ',TotalN:8);
writeln('       Average ','            ',AveD:8:3,'       ',AveE:8:3,
        '         ',AveN:8:3);
writeln('       Ratio Actual ','       ',RatioD:8:3,'       ',
                RatioE:8:3,'         ',RatioN:8:3);
writeln('       Ratio Ideal ','        ',DayRatio:8:3,'       ',
                EveRatio:8:3,'         ',NiteRatio:8:3);

writeln('     ============================================================',
        '==========');
writeln('Press <Enter> to continue ... ');
readln;
ClrScr;

end; {*********Procedure ViewStaffList4Week*******************************}

{*************************************************************************}


Procedure PrintStaffList4Week;

Var
   CountD,
   CountE,
   CountN                : Array[1..28] of Longint;
   Shift4,
   Pattern4              : Char;
```

```
   i,j,k,l                 : integer;
   TotalD, TotalE, TotalN : Longint;
   AveD, AveE, AveN        : real;
   RatioD, RatioE, RatioN : real;

Procedure InitCount;
var
   m : integer;
begin
     for m := 1 to 28 do
     begin
         CountD[m] := 0;
         CountE[m] := 0;
         CountN[m] := 0;
     end;
end;

begin

InitCount;

Assign(PrintStaff4,'LPT1');
Rewrite(PrintStaff4);

Reset(SchedOut);
Reset(ShiftBase3);

While Not Eof(SchedOut) and Not Eof(ShiftBase3) do
begin
     j := 1;
     While Not Eoln(ShiftBase3) do
     begin
          read(ShiftBase3,Shift4);
          for i:=j to j + 6 do
          begin
               read(Schedout, Pattern4);
               if (Shift4 = 'D') and (Pattern4 = '1') then
                  CountD[i] := CountD[i] + 1
               else
               if (Shift4 = 'E') and (Pattern4 = '1') then
                  CountE[i] := CountE[i] + 1
               else
               if (Shift4 = 'N') and (Pattern4 = '1') then
                  CountN[i] := CountN[i] + 1;
          end;
          j := j + 7;
     end;
readln (ShiftBase3);
readln (SchedOut);
end;

Close(SchedOut);
Close(ShiftBase3);

writeln(PrintStaff4,'       STAFF LIST REPORT:                          ');
writeln(PrintStaff4,'                                                   ');
writeln(PrintStaff4);
writeln(PrintStaff4,'     ------------------------------------------------',
                    '----------------------');
writeln(PrintStaff4,'       Day :               ',',   Day-Shift',
                    '    ',',' Evening-Shift',',    ',' Night-Shift');
writeln(PrintStaff4,'     ------------------------------------------------',
                    '----------------------');
TotalD := 0;
TotalE := 0;
```

```
TotalN := 0;

for l := 1 to 28 do
begin
    if (l=1) or (l=8) or (l=15) or (l=22) then
    write(PrintStaff4,'       Monday   ');
    if (l=2) or (l=9) or (l=16) or (l=23) then
    write(PrintStaff4,'       Tuesday  ');
    if (l=3) or (l=10) or (l=17) or (l=24) then
    write(PrintStaff4,'       Wednesday');
    if (l=4) or (l=11) or (l=18) or (l=25) then
    write(PrintStaff4,'       Thursday ');
    if (l=5) or (l=12) or (l=19) or (l=26) then
    write(PrintStaff4,'       Friday   ');
    if (l=6) or (l=13) or (l=20) or (l=27) then
    write(PrintStaff4,'       Saturday ');
    if (l=7) or (l=14) or (l=21) or (l=28) then
    write(PrintStaff4,'       Sunday   ');

    writeln(PrintStaff4,'     ',l:2,'     ',CountD[l]:8,
                        '         ',CountE[l]:8,'           ',CountN[l]:8);

    TotalD := TotalD + CountD[l];
    TotalE := TotalE + CountE[l];
    TotalN := TotalN + CountN[l];

end;

    AveD := (TotalD / 28);
    AveE := (TotalE / 28);
    AveN := (TotalN / 28);
    RatioD := (TotalD) / (TotalD + TotalE + TotalN);
    RatioE := (TotalE) / (TotalD + TotalE + TotalN);
    RatioN := (TotalN) / (TotalD + TotalE + TotalN);

writeln(PrintStaff4,'     ---------------------------------------------',
                    '----------------------');
writeln(PrintStaff4,'       Total   ','             ',TotalD:8,
                    '         ',TotalE:8,'           ',TotalN:8);
writeln(PrintStaff4,'       Average','              ',AveD:8:3,
                    '         ',AveE:8:3,'           ',AveN:8:3);
writeln(PrintStaff4,'       Ratio Actual ','        ',RatioD:8:3,
                    '         ',RatioE:8:3,'           ',RatioN:8:3);
writeln(PrintStaff4,'       Ratio Ideal ','         ',DayRatio:8:3,
                    '         ',EveRatio:8:3,'           ',NiteRatio:8:3);

writeln(PrintStaff4,'     =============================================',
                    '======================');


end; {*********Procedure PrintStaffList4Week*****************************}

{************************************************************************}


Procedure StaffList4Week;

Var
   CountD,
   CountE,
   CountN                  : Array[1..28] of Longint;
   Shift4,
   Pattern4                : Char;
   i,j,k,l                 : integer;
   TotalD, TotalE, TotalN  : Longint;
```

```
   AveD, AveE, AveN       : real;
   RatioD, RatioE, RatioN : real;

Procedure InitCount;
var
   m : integer;
begin
     for m := 1 to 28 do
     begin
         CountD[m] := 0;
         CountE[m] := 0;
         CountN[m] := 0;
     end;
end;

begin

InitCount;

Reset(SchedOut);
Reset(ShiftBase3);

While Not Eof(SchedOut) and Not Eof(ShiftBase3) do
begin
     j := 1;
     While Not Eoln(ShiftBase3) do
     begin
          read(ShiftBase3,Shift4);
          for i:=j to j + 6 do
          begin
               read(Schedout, Pattern4);
               if (Shift4 = 'D') and (Pattern4 = '1') then
                  CountD[i] := CountD[i] + 1
               else
               if (Shift4 = 'E') and (Pattern4 = '1') then
                  CountE[i] := CountE[i] + 1
               else
               if (Shift4 = 'N') and (Pattern4 = '1') then
                  CountN[i] := CountN[i] + 1;
          end;
          j := j + 7;
     end;
readln (ShiftBase3);
readln (SchedOut);
end;

Close(SchedOut);
Close(ShiftBase3);

Assign(StafList4,'STAF4.DAT');
Rewrite(StafList4);

writeln(StafList4,'     STAFF LIST REPORT:                ');
writeln(StafList4,'                                       ');
writeln(StafList4);
writeln(StafList4,
        '     ------------------------------------------------------',
        '----------------');
writeln(StafList4,'     Day :                ',',   Day-Shift',
        '    ',' Evening-Shift','    ',' Night-Shift');
writeln(StafList4,
        '     ------------------------------------------------------',
        '----------------');
TotalD := 0;
TotalE := 0;
```

```
TotalN := 0;

for l := 1 to 28 do
begin
    if (l=1) or (l=8) or (l=15) or (l=22) then
    write(StafList4,'Monday   ');
    if (l=2) or (l=9) or (l=16) or (l=23) then
    write(Staflist4,'Tuesday  ');
    if (l=3) or (l=10) or (l=17) or (l=24) then
    write(StafList4,'Wednesday');
    if (l=4) or (l=11) or (l=18) or (l=25) then
    write(StafList4,'Thursday ');
    if (l=5) or (l=12) or (l=19) or (l=26) then
    write(Staflist4,'Friday   ');
    if (l=6) or (l=13) or (l=20) or (l=27) then
    write(StafList4,'Saturday ');
    if (l=7) or (l=14) or (l=21) or (l=28) then
    write(StafList4,'Sunday   ');

    writeln (StafList4,
             '     ',l:2,'     ',CountD[l]:8,'         ',CountE[l]:8,
             '           ',CountN[l]:8);

    TotalD := TotalD + CountD[l];
    TotalE := TotalE + CountE[l];
    TotalN := TotalN + CountN[l];

end;

    AveD := (TotalD / 28);
    AveE := (TotalE / 28);
    AveN := (TotalN / 28);
    RatioD := (TotalD) / (TotalD + TotalE + TotalN);
    RatioE := (TotalE) / (TotalD + TotalE + TotalN);
    RatioN := (TotalN) / (TotalD + TotalE + TotalN);

writeln(StafList4,
        '     ----------------------------------------------------',
        '---------------');
writeln(StafList4,
        '     Total   ','             ',TotalD:8,'         ',TotalE:8,
        '           ',TotalN:8);
writeln(StafList4,
         '     Average','              ',AveD:8:3,'         ',AveE:8:3,
        '           ',AveN:8:3);
writeln(StafList4,
         '     Ratio Actual ','        ',RatioD:8:3,'          ',RatioE:8:3,
        '           ',RatioN:8:3);
writeln(StafList4,
         '     Ratio Ideal ','         ',DayRatio:8:3,'         ',
         EveRatio:8:3,'          ',NiteRatio:8:3);

writeln(StafList4,
        '     ====================================================',
        '===============');
Close(StafList4);

end; {*********Procedure StaffList4Week**********************************}

{*****************************************************************************}

Procedure ViewStaffList8Week;

Var
   CountD,
```

```
   CountE,
   CountN                  : Array[1..56] of Longint;
   Shift8,
   Pattern8                : Char;
   i,j,k,l                 : integer;
   TotalD, TotalE, TotalN  : Longint;
   AveD, AveE, AveN        : real;
   RatioD, RatioE, RatioN  : real;

Procedure InitCount;
var
   m : integer;
begin
     for m := 1 to 56 do
     begin
         CountD[m] := 0;
         CountE[m] := 0;
         CountN[m] := 0;
     end;
end;

begin

InitCount;

Reset(EightWeekBase);
Reset(Shift8WBase);

While Not Eof(EightWeekBase) and Not Eof(Shift8WBase) do
begin
     j := 1;
     While Not Eoln(Shift8WBase) do
     begin
          read(Shift8WBase,Shift8);
          for i:=j to j + 6 do
          begin
               read(EightWeekBase, Pattern8);
               if (Shift8 = 'D') and (Pattern8 = '1') then
                  CountD[i] := CountD[i] + 1
               else
               if (Shift8 = 'E') and (Pattern8 = '1') then
                  CountE[i] := CountE[i] + 1
               else
               if (Shift8 = 'N') and (Pattern8 = '1') then
                  CountN[i] := CountN[i] + 1;
          end;
          j := j + 7;
     end;
readln (Shift8WBase);
readln (EightWeekBase);
end;

Close(EightWeekBase);
Close(Shift8WBase);

writeln('     STAFF LIST REPORT:                    ');
writeln('                                           ');
writeln;
writeln('     ------------------------------------------------------',
        '----------');
writeln('     Day :               ','   Day-Shift',
        '    ',' Evening-Shift','    ',' Night-Shift');
writeln('     ------------------------------------------------------',
        '----------');
```

```
TotalD := 0;
TotalE := 0;
TotalN := 0;

k := 0;
for l := 1 to 56 do
begin
    if (l=1) or (l=8) or (l=15) or (l=22) or (l=29) or (l=36) or (l=43)
        or (l=50) then
        write('     Monday   ');
    if (l=2) or (l=9) or (l=16) or (l=23) or (l=30) or (l=37) or (l=44)
        or (l=51) then
        write('     Tuesday  ');
    if (l=3) or (l=10) or (l=17) or (l=24) or (l=31) or (l=38) or (l=45)
        or (l=52) then
        write('     Wednesday');
    if (l=4) or (l=11) or (l=18) or (l=25) or (l=32) or (l=39) or (l=46)
       or (l=53) then
        write('     Thursday ');
    if (l=5) or (l=12) or (l=19) or (l=26) or (l=33) or (l=40) or (l=47)
       or (l=54) then
        write('     Friday   ');
    if (l=6) or (l=13) or (l=20) or (l=27) or (l=34) or (l=41) or (l=48)
       or (l=55) then
        write('     Saturday ');
    if (l=7) or (l=14) or (l=21) or (l=28) or (l=35) or (l=42) or (l=49)
       or (l=56) then
        write('     Sunday   ');

    writeln ('     ',l:2,'     ',CountD[l]:8,'         ',CountE[l]:8,
             '           ',CountN[l]:8);

    TotalD := TotalD + CountD[l];
    TotalE := TotalE + CountE[l];
    TotalN := TotalN + CountN[l];

    k := k + 1;
    if k = 17 then
    begin
    k := 0;
    writeln('Press <Enter> to continue ... ');
    readln;
    end

end;

    AveD := (TotalD / 56);
    AveE := (TotalE / 56);
    AveN := (TotalN / 56);
    RatioD := (TotalD) / (TotalD + TotalE + TotalN);
    RatioE := (TotalE) / (TotalD + TotalE + TotalN);
    RatioN := (TotalN) / (TotalD + TotalE + TotalN);

writeln('     ------------------------------------------------------------',
        '----------');
writeln('      Total   ','             ',TotalD:8,'         ',TotalE:8,
        '           ',TotalN:8);
writeln('      Average','             ',AveD:8:3,'         ',AveE:8:3,
        '           ',AveN:8:3);
writeln('      Ratio Actual ','        ',RatioD:8:3,'          ',
       RatioE:8:3,'           ',RatioN:8:3);
writeln('      Ratio Ideal ','         ',DayRatio:8:3,'          ',
       EveRatio:8:3,'           ',NiteRatio:8:3);

writeln('     ============================================================',
```

```
        '==========');
writeln('Press <Enter> to continue ...                                    ');
readln;
ClrScr;

end; {*********Procedure ViewStaffList8Week*****************************}

{*************************************************************************}


Procedure PrintStaffList8Week;

Var
   CountD,
   CountE,
   CountN                  : Array[1..56] of Longint;
   Shift8,
   Pattern8                : Char;
   i,j,k,l                 : integer;
   TotalD, TotalE, TotalN  : Longint;
   AveD, AveE, AveN        : real;
   RatioD, RatioE, RatioN  : real;

Procedure InitCount;
var
   m : integer;
begin
     for m := 1 to 56 do
     begin
         CountD[m] := 0;
         CountE[m] := 0;
         CountN[m] := 0;
     end;
end;

begin

InitCount;

Assign(PrintStaff8,'LPT1');
Rewrite(PrintStaff8);

Reset(EightWeekBase);
Reset(Shift8WBase);

While Not Eof(EightWeekBase) and Not Eof(Shift8WBase) do
begin
     j := 1;
     While Not Eoln(Shift8WBase) do
     begin
          read(Shift8WBase,Shift8);
          for i:=j to j + 6 do
          begin
               read(EightWeekBase, Pattern8);
               if (Shift8 = 'D') and (Pattern8 = '1') then
                  CountD[i] := CountD[i] + 1
               else
               if (Shift8 = 'E') and (Pattern8 = '1') then
                  CountE[i] := CountE[i] + 1
               else
               if (Shift8 = 'N') and (Pattern8 = '1') then
                  CountN[i] := CountN[i] + 1;
          end;
          j := j + 7;
     end;
```

```
readln (Shift8WBase);
readln (EightWeekBase);
end;

Close(EightWeekBase);
Close(Shift8WBase);

writeln(PrintStaff8,'     STAFF LIST REPORT:                    ');
writeln(PrintStaff8,'                                           ');
writeln(PrintStaff8);
writeln(PrintStaff8,'     ------------------------------------------',
                    '-------------------------');
writeln(PrintStaff8,'     Day :                ',',   Day-Shift',
                    '    ',' Evening-Shift','    ',' Night-Shift');
writeln(PrintStaff8,'     ------------------------------------------',
                    '-------------------------');

TotalD := 0;
TotalE := 0;
TotalN := 0;

for l := 1 to 56 do
begin
    if (l=1) or (l=8) or (l=15) or (l=22) or (l=29) or (l=36) or (l=43)
       or (l=50) then
       write(PrintStaff8,'     Monday   ');
    if (l=2) or (l=9) or (l=16) or (l=23) or (l=30) or (l=37) or (l=44)
       or (l=51) then
       write(PrintStaff8,'     Tuesday  ');
    if (l=3) or (l=10) or (l=17) or (l=24) or (l=31) or (l=38) or (l=45)
       or (l=52) then
       write(PrintStaff8,'     Wednesday');
    if (l=4) or (l=11) or (l=18) or (l=25) or (l=32) or (l=39) or (l=46)
      or (l=53) then
       write(PrintStaff8,'     Thursday ');
    if (l=5) or (l=12) or (l=19) or (l=26) or (l=33) or (l=40) or (l=47)
      or (l=54) then
       write(PrintStaff8,'     Friday   ');
    if (l=6) or (l=13) or (l=20) or (l=27) or (l=34) or (l=41) or (l=48)
      or (l=55) then
       write(PrintStaff8,'     Saturday ');
    if (l=7) or (l=14) or (l=21) or (l=28) or (l=35) or (l=42) or (l=49)
      or (l=56) then
       write(PrintStaff8,'     Sunday   ');

    writeln (PrintStaff8,'     ',l:2,'     ',CountD[l]:8,'          ',
                         CountE[l]:8,'           ',CountN[l]:8);

    TotalD := TotalD + CountD[l];
    TotalE := TotalE + CountE[l];
    TotalN := TotalN + CountN[l];

end;

    AveD := (TotalD / 56);
    AveE := (TotalE / 56);
    AveN := (TotalN / 56);
    RatioD := (TotalD) / (TotalD + TotalE + TotalN);
    RatioE := (TotalE) / (TotalD + TotalE + TotalN);
    RatioN := (TotalN) / (TotalD + TotalE + TotalN);

writeln(PrintStaff8,'     ------------------------------------------',
                    '-------------------------');
writeln(PrintStaff8,'     Total    ',',             ',TotalD:8,
                    '          ',TotalE:8,'           ',TotalN:8);
```

```
writeln(PrintStaff8,'     Average',','               ',AveD:8:3,
                    '          ',AveE:8:3,'           ',AveN:8:3);
writeln(PrintStaff8,'     Ratio Actual ',','         ',RatioD:8:3,
                    '          ',RatioE:8:3,'           ',RatioN:8:3);
writeln(PrintStaff8,'     Ratio Ideal ',','          ',DayRatio:8:3,
                    '          ',EveRatio:8:3,'           ',NiteRatio:8:3);

writeln(PrintStaff8,'     ==========================================',
                    '========================');
writeln(PrintStaff8);


end; {*********Procedure PrintStaffList8Week*****************************}

{*************************************************************************}


Procedure StaffList8Week;

Var
   CountD,
   CountE,
   CountN                  : Array[1..56] of Longint;
   Shift8,
   Pattern8                : Char;
   i,j,k,l                 : integer;
   TotalD, TotalE, TotalN  : Longint;
   AveD, AveE, AveN        : real;
   RatioD, RatioE, RatioN  : real;

Procedure InitCount;
var
   m : integer;
begin
     for m := 1 to 56 do
     begin
          CountD[m] := 0;
          CountE[m] := 0;
          CountN[m] := 0;
     end;
end;

begin

InitCount;

Reset(EightWeekBase);
Reset(Shift8WBase);

While Not Eof(EightWeekBase) and Not Eof(Shift8WBase) do
begin
     j := 1;
     While Not Eoln(Shift8WBase) do
     begin
          read(Shift8WBase,Shift8);
          for i:=j to j + 6 do
          begin
               read(EightWeekBase, Pattern8);
               if (Shift8 = 'D') and (Pattern8 = '1') then
                  CountD[i] := CountD[i] + 1
               else
               if (Shift8 = 'E') and (Pattern8 = '1') then
                  CountE[i] := CountE[i] + 1
               else
               if (Shift8 = 'N') and (Pattern8 = '1') then
```

```
                    CountN[i] := CountN[i] + 1;
            end;
            j := j + 7;
      end;
readln (Shift8WBase);
readln (EightWeekBase);
end;

Close(EightWeekBase);
Close(Shift8WBase);

Assign(StafList8,'STAF8.DAT');
Rewrite(StafList8);

writeln(StafList8,'     STAFF LIST REPORT:                ');
writeln(StafList8,'                                       ');
writeln;
writeln(StafList8,
        '     ------------------------------------------------------',
        '------------');
writeln(StafList8,'     Day :                ',',   Day-Shift',
        '    ',' Evening-Shift','    ',' Night-Shift');
writeln(StafList8,
        '     ------------------------------------------------------',
        '------------');

TotalD := 0;
TotalE := 0;
TotalN := 0;

for l := 1 to 56 do
begin
    if (l=1) or (l=8) or (l=15) or (l=22) or (l=29) or (l=36) or (l=43)
        or (l=50) then
        write(StafList8,'     Monday   ');
    if (l=2) or (l=9) or (l=16) or (l=23) or (l=30) or (l=37) or (l=44)
        or (l=51) then
        write(Staflist8,'     Tuesday  ');
    if (l=3) or (l=10) or (l=17) or (l=24) or (l=31) or (l=38) or (l=45)
        or (l=52) then
        write(StafList8,'     Wednesday');
    if (l=4) or (l=11) or (l=18) or (l=25) or (l=32) or (l=39) or (l=46)
        or (l=53) then
        write(StafList8,'     Thursday ');
    if (l=5) or (l=12) or (l=19) or (l=26) or (l=33) or (l=40) or (l=47)
        or (l=54) then
        write(Staflist8,'     Friday   ');
    if (l=6) or (l=13) or (l=20) or (l=27) or (l=34) or (l=41) or (l=48)
        or (l=55) then
        write(StafList8,'     Saturday ');
    if (l=7) or (l=14) or (l=21) or (l=28) or (l=35) or (l=42) or (l=49)
        or (l=56) then
        write(StafList8,'     Sunday   ');

    writeln (StafList8,
             '     ',l:2,'     ',CountD[l]:8,'         ',CountE[l]:8,
             '           ',CountN[l]:8);

    TotalD := TotalD + CountD[l];
    TotalE := TotalE + CountE[l];
    TotalN := TotalN + CountN[l];

end;

    AveD := (TotalD / 56);
```

```
    AveE := (TotalE / 56);
    AveN := (TotalN / 56);
    RatioD := (TotalD) / (TotalD + TotalE + TotalN);
    RatioE := (TotalE) / (TotalD + TotalE + TotalN);
    RatioN := (TotalN) / (TotalD + TotalE + TotalN);

writeln(StafList8,
        '     ------------------------------------------------------------',
        '-----------');
writeln(StafList8,
        '     Total   ','             ',TotalD:8,'         ',TotalE:8,
        '           ',TotalN:8);
writeln(StafList8,
        '     Average','              ',AveD:8:3,'         ',AveE:8:3,
        '           ',AveN:8:3);
writeln(StafList8,
        '     Ratio Actual ','        ',RatioD:8:3,'          ',RatioE:8:3,
        '           ',RatioN:8:3);
writeln(StafList8,
         '    Ratio Ideal ','         ',DayRatio:8:3,'          ',
         EveRatio:8:3,'           ',NiteRatio:8:3);

writeln(StafList8,
        '     ============================================================',
        '===========');

Close(StafList8);


end; {*********Procedure StaffList8Week**********************************}

{*****************************************************************************}

Procedure ViewStaffList12Week;

Var
   CountD,
   CountE,
   CountN                : Array[1..84] of Longint;
   Shift12,
   Pattern12             : Char;
   i,j,k,l               : integer;
   TotalD, TotalE, TotalN : Longint;
   AveD, AveE, AveN      : real;
   RatioD, RatioE, RatioN : real;

Procedure InitCount;
var
   m : integer;
begin
     for m := 1 to 84 do
     begin
         CountD[m] := 0;
         CountE[m] := 0;
         CountN[m] := 0;
     end;
end;

begin

InitCount;

Reset(TwelveWeekBase);
Reset(Shift12WBase);
```

```
While Not Eof(TwelveWeekBase) and Not Eof(Shift12WBase) do
begin
     j := 1;
     While Not Eoln(Shift12WBase) do
     begin
          read(Shift12WBase,Shift12);
          for i:=j to j + 6 do
          begin
               read(TwelveWeekBase, Pattern12);
               if (Shift12 = 'D') and (Pattern12 = '1') then
                  CountD[i] := CountD[i] + 1
               else
               if (Shift12 = 'E') and (Pattern12 = '1') then
                  CountE[i] := CountE[i] + 1
               else
               if (Shift12 = 'N') and (Pattern12 = '1') then
                  CountN[i] := CountN[i] + 1;
          end;
          j := j + 7;
     end;
readln (Shift12WBase);
readln (TwelveWeekBase);
end;

Close(TwelveWeekBase);
Close(Shift12WBase);


writeln('     STAFF LIST REPORT:                    ');
writeln('                                           ');
writeln;
writeln('     ------------------------------------------------------',
        '----------------');
writeln('     Day :               ','   Day-Shift',
        '    ',' Evening-Shift','    ',' Night-Shift');
writeln('     ------------------------------------------------------',
        '----------------');

TotalD := 0;
TotalE := 0;
TotalN := 0;

k := 0;
for l := 1 to 84 do
begin
    if (l=1) or (l=8) or (l=15) or (l=22) or (l=29) or (l=36) or (l=43)
        or (l=50) or (l=57) or (l=64) or (l=71) or (l=78) then
        write('     Monday   ');
    if (l=2) or (l=9) or (l=16) or (l=23) or (l=30) or (l=37) or (l=44)
        or (l=51) or (l=58) or (l=65) or (l=72) or (l=79) then
        write('     Tuesday  ');
    if (l=3) or (l=10) or (l=17) or (l=24) or (l=31) or (l=38) or (l=45)
        or (l=52) or (l=59) or (l=66) or (l=73) or (l=80) then
        write('     Wednesday');
    if (l=4) or (l=11) or (l=18) or (l=25) or (l=32) or (l=39) or (l=46)
        or (l=53) or (l=60) or (l=67) or (l=74) or (l=81) then
        write('     Thursday ');
    if (l=5) or (l=12) or (l=19) or (l=26) or (l=33) or (l=40) or (l=47)
        or (l=54) or (l=61) or (l=68) or (l=75) or (l=82) then
        write('     Friday   ');
    if (l=6) or (l=13) or (l=20) or (l=27) or (l=34) or (l=41) or (l=48)
       or (l=55) or (l=62) or (l=69) or (l=76) or (l=83) then
        write('     Saturday ');
    if (l=7) or (l=14) or (l=21) or (l=28) or (l=35) or (l=42) or (l=49)
        or (l=56) or (l=63) or (l=70) or (l=77) or (l=84) then
```

```
        write('     Sunday   ');

    writeln ('     ',l:2,'     ',CountD[l]:8,'         ',CountE[l]:8,
             '           ',CountN[l]:8);

    TotalD := TotalD + CountD[l];
    TotalE := TotalE + CountE[l];
    TotalN := TotalN + CountN[l];

    k := k + 1;
    if k = 17 then
    begin
    k := 0;
    writeln('Press <Enter> to continue ... ');
    readln;
    end;

end;

    AveD := (TotalD / 84);
    AveE := (TotalE / 84);
    AveN := (TotalN / 84);
    RatioD := (TotalD) / (TotalD + TotalE + TotalN);
    RatioE := (TotalE) / (TotalD + TotalE + TotalN);
    RatioN := (TotalN) / (TotalD + TotalE + TotalN);

writeln('     ------------------------------------------------------------',
        '---------------');
writeln('     Total   ','             ',TotalD:8,'         ',TotalE:8,
        '           ',TotalN:8);
writeln('     Average','             ',AveD:8:3,'         ',AveE:8:3,
        '           ',AveN:8:3);
writeln('     Ratio Actual ','        ',RatioD:8:3,'          ',
        RatioE:8:3,'           ',RatioN:8:3);
writeln('     Ratio Ideal ','         ',DayRatio:8:3,'          ',
        EveRatio:8:3,'           ',NiteRatio:8:3);

writeln('     ============================================================',
        '===============');
writeln;
writeln('Press <Enter> to continue ...                                    ');
readln;
ClrScr;

end; {*********Procedure ViewStaffList12Week*****************************}

{*************************************************************************}

Procedure PrintStaffList12Week;

Var
   CountD,
   CountE,
   CountN                  : Array[1..84] of Longint;
   Shift12,
   Pattern12               : Char;
   i,j,k,l                 : integer;
   TotalD, TotalE, TotalN  : Longint;
   AveD, AveE, AveN        : real;
   RatioD, RatioE, RatioN  : real;

Procedure InitCount;
var
   m : integer;
begin
```

```
     for m := 1 to 84 do
     begin
         CountD[m] := 0;
         CountE[m] := 0;
         CountN[m] := 0;
     end;
end;

begin

InitCount;

Reset(TwelveWeekBase);
Reset(Shift12WBase);

While Not Eof(TwelveWeekBase) and Not Eof(Shift12WBase) do
begin
     j := 1;
     While Not Eoln(Shift12WBase) do
     begin
          read(Shift12WBase,Shift12);
          for i:=j to j + 6 do
          begin
               read(TwelveWeekBase, Pattern12);
               if (Shift12 = 'D') and (Pattern12 = '1') then
                  CountD[i] := CountD[i] + 1
               else
               if (Shift12 = 'E') and (Pattern12 = '1') then
                  CountE[i] := CountE[i] + 1
               else
               if (Shift12 = 'N') and (Pattern12 = '1') then
                  CountN[i] := CountN[i] + 1;
          end;
          j := j + 7;
     end;
readln (Shift12WBase);
readln (TwelveWeekBase);
end;

Close(TwelveWeekBase);
Close(Shift12WBase);

Assign(PrintStaff12,'LPT1');
Rewrite(PrintStaff12);

writeln(PrintStaff12,'     STAFF LIST REPORT:                ');
writeln(PrintStaff12,'                                       ');
writeln(PrintStaff12);
writeln(PrintStaff12,
        '     ------------------------------------------------------',
        '------------');
writeln(PrintStaff12,'     Day :                ','    Day-Shift',
        '    ',' Evening-Shift','    ',' Night-Shift');
writeln(PrintStaff12,
        '     ------------------------------------------------------',
        '------------');

TotalD := 0;
TotalE := 0;
TotalN := 0;

for l := 1 to 84 do
begin
    if (l=1) or (l=8) or (l=15) or (l=22) or (l=29) or (l=36) or (l=43)
       or (l=50) or (l=57) or (l=64) or (l=71) or (l=78) then
```

```
        write(PrintStaff12,'     Monday   ');
    if (l=2) or (l=9) or (l=16) or (l=23) or (l=30) or (l=37) or (l=44)
        or (l=51) or (l=58) or (l=65) or (l=72) or (l=79) then
        write(PrintStaff12,'     Tuesday  ');
    if (l=3) or (l=10) or (l=17) or (l=24) or (l=31) or (l=38) or (l=45)
        or (l=52) or (l=59) or (l=66) or (l=73) or (l=80) then
        write(PrintStaff12,'     Wednesday');
    if (l=4) or (l=11) or (l=18) or (l=25) or (l=32) or (l=39) or (l=46)
        or (l=53) or (l=60) or (l=67) or (l=74) or (l=81) then
        write(PrintStaff12,'     Thursday ');
    if (l=5) or (l=12) or (l=19) or (l=26) or (l=33) or (l=40) or (l=47)
        or (l=54) or (l=61) or (l=68) or (l=75) or (l=82) then
        write(PrintStaff12,'     Friday   ');
    if (l=6) or (l=13) or (l=20) or (l=27) or (l=34) or (l=41) or (l=48)
        or (l=55) or (l=62) or (l=69) or (l=76) or (l=83) then
        write(PrintStaff12,'     Saturday ');
    if (l=7) or (l=14) or (l=21) or (l=28) or (l=35) or (l=42) or (l=49)
        or (l=56) or (l=63) or (l=70) or (l=77) or (l=84) then
        write(PrintStaff12,'     Sunday   ');

    writeln (PrintStaff12,
                '     ',l:2,'     ',CountD[l]:8,'         ',CountE[l]:8,
             '           ',CountN[l]:8);


    TotalD := TotalD + CountD[l];
    TotalE := TotalE + CountE[l];
    TotalN := TotalN + CountN[l];

end;

    AveD := (TotalD / 84);
    AveE := (TotalE / 84);
    AveN := (TotalN / 84);
    RatioD := (TotalD) / (TotalD + TotalE + TotalN);
    RatioE := (TotalE) / (TotalD + TotalE + TotalN);
    RatioN := (TotalN) / (TotalD + TotalE + TotalN);

writeln(PrintStaff12,'     ------------------------------------------',
                     '-------------------------');
writeln(PrintStaff12,'     Total   ','             ',TotalD:8,
                     '         ',TotalE:8,'           ',TotalN:8);
writeln(PrintStaff12,'     Average','              ',AveD:8:3,
                     '         ',AveE:8:3,'           ',AveN:8:3);
writeln(PrintStaff12,'     Ratio Actual ','        ',RatioD:8:3,
                     '         ',RatioE:8:3,'           ',RatioN:8:3);
writeln(PrintStaff12,'     Ratio Ideal ','         ',DayRatio:8:3,
                   '       ',EveRatio:8:3,'           ',NiteRatio:8:3);

writeln(PrintStaff12,'     ==========================================',
                     '=========================');

Close(PrintStaff12);


end; {*********Procedure PrintStaffList12Week*****************************}


{*************************************************************************}


Procedure StaffList12Week;

Var
   CountD,
```

```
   CountE,
   CountN                  : Array[1..84] of Longint;
   Shift12,
   Pattern12               : Char;
   i,j,k,l                 : integer;
   TotalD, TotalE, TotalN  : Longint;
   AveD, AveE, AveN        : real;
   RatioD, RatioE, RatioN  : real;

Procedure InitCount;
var
   m : integer;
begin
     for m := 1 to 84 do
     begin
         CountD[m] := 0;
         CountE[m] := 0;
         CountN[m] := 0;
     end;
end;

begin

InitCount;

Reset(TwelveWeekBase);
Reset(Shift12WBase);

While Not Eof(TwelveWeekBase) and Not Eof(Shift12WBase) do
begin
     j := 1;
     While Not Eoln(Shift12WBase) do
     begin
          read(Shift12WBase,Shift12);
          for i:=j to j + 6 do
          begin
               read(TwelveWeekBase, Pattern12);
               if (Shift12 = 'D') and (Pattern12 = '1') then
                  CountD[i] := CountD[i] + 1
               else
               if (Shift12 = 'E') and (Pattern12 = '1') then
                  CountE[i] := CountE[i] + 1
               else
               if (Shift12 = 'N') and (Pattern12 = '1') then
                  CountN[i] := CountN[i] + 1;
          end;
          j := j + 7;
     end;
readln (Shift12WBase);
readln (TwelveWeekBase);
end;

Close(TwelveWeekBase);
Close(Shift12WBase);

Assign(StafList12,'STAF12.DAT');
Rewrite(StafList12);

writeln(StafList12,'     STAFF LIST REPORT:                ');
writeln(StafList12,'                                       ');
writeln(StafList12);
writeln(StafList12,
        '     ------------------------------------------------------',
        '----------------');
writeln(StafList12,'     Day :               ','   Day-Shift',
```

```
        '    ','Evening-Shift','    ','Night-Shift');
writeln(StafList12,
        '     ------------------------------------------------------',
        '----------------');

TotalD := 0;
TotalE := 0;
TotalN := 0;

for l := 1 to 84 do
begin
    if (l=1) or (l=8) or (l=15) or (l=22) or (l=29) or (l=36) or (l=43)
       or (l=50) or (l=57) or (l=64) or (l=71) or (l=78) then
       write(StafList12,'     Monday   ');
    if (l=2) or (l=9) or (l=16) or (l=23) or (l=30) or (l=37) or (l=44)
       or (l=51) or (l=58) or (l=65) or (l=72) or (l=79) then
       write(Staflist12,'     Tuesday  ');
    if (l=3) or (l=10) or (l=17) or (l=24) or (l=31) or (l=38) or (l=45)
       or (l=52) or (l=59) or (l=66) or (l=73) or (l=80) then
       write(StafList12,'     Wednesday');
    if (l=4) or (l=11) or (l=18) or (l=25) or (l=32) or (l=39) or (l=46)
       or (l=53) or (l=60) or (l=67) or (l=74) or (l=81) then
       write(StafList12,'     Thursday ');
    if (l=5) or (l=12) or (l=19) or (l=26) or (l=33) or (l=40) or (l=47)
       or (l=54) or (l=61) or (l=68) or (l=75) or (l=82) then
       write(Staflist12,'     Friday   ');
    if (l=6) or (l=13) or (l=20) or (l=27) or (l=34) or (l=41) or (l=48)
       or (l=55) or (l=62) or (l=69) or (l=76) or (l=83) then
       write(StafList12,'     Saturday ');
    if (l=7) or (l=14) or (l=21) or (l=28) or (l=35) or (l=42) or (l=49)
       or (l=56) or (l=63) or (l=70) or (l=77) or (l=84) then
       write(StafList12,'     Sunday   ');

    writeln (StafList12,
             '     ',l:2,'     ',CountD[l]:8,'         ',CountE[l]:8,
             '           ',CountN[l]:8);


    TotalD := TotalD + CountD[l];
    TotalE := TotalE + CountE[l];
    TotalN := TotalN + CountN[l];

end;

    AveD := (TotalD / 84);
    AveE := (TotalE / 84);
    AveN := (TotalN / 84);
    RatioD := (TotalD) / (TotalD + TotalE + TotalN);
    RatioE := (TotalE) / (TotalD + TotalE + TotalN);
    RatioN := (TotalN) / (TotalD + TotalE + TotalN);

writeln(StafList12,'     ------------------------------------------',
                   '--------------------');
writeln(StafList12,'       Total   ','             ',TotalD:8,'         ',
                   TotalE:8,'           ',TotalN:8);
writeln(StafList12,'       Average','              ',AveD:8:3,'         ',
                   AveE:8:3,'           ',AveN:8:3);
writeln(StafList12,'       Ratio Actual ','        ',RatioD:8:3,
                   '         ',RatioE:8:3,'           ',RatioN:8:3);
writeln(StafList12,'       Ratio Ideal ','         ',DayRatio:8:3,
                   '         ',EveRatio:8:3,'           ',NiteRatio:8:3);

writeln(StafList12,'     ==========================================',
                   '====================');
```

```
Close(StafList12);

end; {*********Procedure StaffList12Week*********************************}

{************************************************************************}

Procedure Length4; { Generating Four-Week Schedules }

begin

ClrScr;

{ User can choose either to use default value for Penalty Cost or        }
{ to customize them.                                                     }

Writeln;
Writeln('       Default Values of Penalty Costs:                        ');
Writeln;
Writeln;
Writeln('       Penalty Cost for having 3 consecutive days off    :=  5 ');
Writeln('       Penalty Cost for having 4 consecutive days off    := 10 ');
Writeln('       Penalty Cost for working 6 consecutive days       := 10 ');
Writeln('       Penalty Cost for working 7 consecutive days       := 30 ');
Writeln('       Penalty Cost for working 8 consecutive days       := 40 ');
Writeln('       Penalty Cost for having single day off            := 10 ');
Writeln('       Penalty Cost for having single day on             := 10 ');
Writeln('       Penalty Cost for working on Saturday, Sunday off  :=  5 ');
Writeln('       Penalty Cost for working on Sunday, Saturday off  :=  5 ');
Writeln('       Penalty Cost for working on Week End              := 10 ');
Writeln;
Writeln;
Writeln('       MAXIMUM PENALTY COST can be set equal to the highest    ');
Writeln('       value of penalty cost. The higher the maximum penalty   ');
Writeln('       cost the less strict the schedules.                     ');
Writeln;
Write('       Use default value for Penalty Costs? (Y/N): ');
Readln(DefaultValue);

if (DefaultValue = 'Y') or (DefaultValue = 'y') then
        DefaultPCost
     else
        CustomizePCost;

{ Then, user will be asked the maximum Penalty Cost that can be tolerated }

Writeln;
Write('       Enter maximum penalty cost:  ');
Readln (MaxPCost);

{ User will be asked how many pattern s/he wants to generate }

ClrScr;
Writeln;
Write('       Enter number of patterns to be generated:  ');
Readln (NumOfPat);
{NumOfPat := 1000;}


CreatePattern;
CreateShiftBase;
ReorderShift;

repeat
```

```
ClrScr;
Writeln;
Write('       Enter number of schedules to be selected',
      '  (Maximum = ',NumOfPat:4,' ):  ');

Readln(ScheduleNum);

PickFourWeekPattern;
CreateShift;

ClrScr;

Writeln;
Write('       Display schedules on the screen? (Y/N)   ');
Readln (SeeSchedule);

If (SeeSchedule = 'Y') or (SeeSchedule = 'y') then
   begin ClrScr; ScreenView; end;

ClrScr;
Writeln;
Write('       Display Staff List Report? (Y/N)   ');
Readln(ViewReport);

If (ViewReport = 'Y') or (ViewReport = 'y') then
   ViewStaffList4Week;

Writeln;
Write('       Save the schedules? (Y/N)   ');
Readln (SaveSchedule);

If (SaveSchedule = 'Y') or (SaveSchedule = 'y') then
   SaveTheSchedule;

Writeln;
Write('       Save Staff List Report? (Y/N)   ');
Readln(SaveReport);

If (SaveReport = 'Y') or (SaveReport = 'y') then
   StaffList4Week;

Writeln;
Write('       Print the schedules? (Y/N)   ');
Readln(PrintSchedule4);

If (PrintSchedule4 = 'Y') or (PrintSchedule4 = 'y') then
   PrintSchedule;

Writeln;
Write('       Print StaffListReport? (Y/N)   ');
Readln(PrintReport4);

If (PrintReport4 = 'Y') or (PrintReport4 = 'y') then
   PrintStaffList4Week;

Writeln;
Write('       Select another set of schedules? (Y/N)   ');
Readln(PickAgain);

until (PickAgain = 'N') or (PickAgain = 'n');
end; { Procedure Length4 }

{*************************************************************************}

Procedure Length8;
```

```
begin

ClrScr;

{ User can choose either to use default value for Penalty Cost or          }
{ to customize them.                                                       }

Writeln;
Writeln('       Default Values of Penalty Costs:                          ');
Writeln;
Writeln;
Writeln('       Penalty Cost for having 3 consecutive days off    :=  5 ');
Writeln('       Penalty Cost for having 4 consecutive days off    := 10 ');
Writeln('       Penalty Cost for working 6 consecutive days       := 10 ');
Writeln('       Penalty Cost for working 7 consecutive days       := 30 ');
Writeln('       Penalty Cost for working 8 consecutive days       := 40 ');
Writeln('       Penalty Cost for having single day off            := 10 ');
Writeln('       Penalty Cost for having single day on             := 10 ');
Writeln('       Penalty Cost for working on Saturday, Sunday off  :=  5 ');
Writeln('       Penalty Cost for working on Sunday, Saturday off  :=  5 ');
Writeln('       Penalty Cost for working on Week End              := 10 ');
Writeln;
Writeln;
Writeln('       MAXIMUM PENALTY COST can be set equal to the highest    ');
Writeln('       value of penalty cost. The higher the maximum penalty   ');
Writeln('       cost the less strict the schedules.                     ');
Writeln;
Write('       Use default value for Penalty Costs? (Y/N): ');
Readln(DefaultValue);

if (DefaultValue = 'Y') or (DefaultValue = 'y') then
        DefaultPCost
     else
        CustomizePCost;

{ Then, user will be asked the maximum Penalty Cost that can be tolerated }

Writeln;
Write('       Enter maximum penalty cost:  ');
Readln (MaxPCost);

{ User will be asked how many pattern s/he wants to generate }

ClrScr;
Writeln;
Write('       Enter number of patterns to be generated:  ');
Readln (NumOfPat);

CreatePattern;
CreateShiftBase;
ReorderShift;

repeat

ClrScr;
Writeln;
Write('       Enter number of schedules to be selected:  ',
      ' (Maximum = ',NumOfPat:4,' ):  ');

Readln(ScheduleNum);

PickFourWeekPattern;
CreateShift;
EightWeekPattern;
CreateSecondShift;
```

```
Create8WeekShift;

ClrScr;
Writeln;
Write('        Display schedules on the screen? (Y/N)  ');
Readln (SeeSchedule);

If (SeeSchedule = 'Y') or (SeeSchedule = 'y') then
   begin ClrScr; ViewEightSchedule; end;

ClrScr;
Writeln;
Write('        Display Staff List Report on screen? (Y/N)  ');
Readln (Display8Staff);

If (Display8Staff = 'Y') or (Display8Staff = 'y') then
   ViewStaffList8Week;

Writeln;
Write('       Save the schedules ? (Y/N)  ');
Readln (SaveSchedule);

If (SaveSchedule = 'Y') or (SaveSchedule = 'y') then
   SaveEightSchedule;

Writeln;
Write('       Save Staff List Report? (Y/N)   ');
Readln(SaveReport);

If (SaveReport = 'Y') or (SaveReport = 'y') then
   StaffList8Week;

Writeln;
Write('       Print the schedules? (Y/N)   ');
Readln(PrintSchedule8);

If (PrintSchedule8 = 'Y') or (PrintSchedule8 = 'y') then
   PrintEightSchedule;

Writeln;
Write('       Print Staff List Report? (Y/N)   ');
Readln(PrintReport8);

If (PrintReport8 = 'Y') or (PrintReport8 = 'y') then
   PrintStaffList8Week;

Writeln;
Write('       Select another set of schedules? (Y/N)   ');
Readln(PickAgain);

until (PickAgain = 'N') or (PickAgain = 'n');

end; { Procedure Length8 }

{*************************************************************************}

Procedure Length12;

begin

ClrScr;

{ User can choose either to use default value for Penalty Cost or        }
{ to customize them.                                                     }
```

```
Writeln;
Writeln('       Default Values of Penalty Costs:                          ');
Writeln;
Writeln;
Writeln('       Penalty Cost for having 3 consecutive days off    :=  5 ');
Writeln('       Penalty Cost for having 4 consecutive days off    := 10 ');
Writeln('       Penalty Cost for working 6 consecutive days       := 10 ');
Writeln('       Penalty Cost for working 7 consecutive days       := 30 ');
Writeln('       Penalty Cost for working 8 consecutive days       := 40 ');
Writeln('       Penalty Cost for having single day off            := 10 ');
Writeln('       Penalty Cost for having single day on             := 10 ');
Writeln('       Penalty Cost for working on Saturday, Sunday off  :=  5 ');
Writeln('       Penalty Cost for working on Sunday, Saturday off  :=  5 ');
Writeln('       Penalty Cost for working on Week End              := 10 ');
Writeln;
Writeln;
Writeln('       MAXIMUM PENALTY COST can be set equal to the highest    ');
Writeln('       value of penalty cost. The higher the maximum penalty   ');
Writeln('       cost the less strict the schedules.                     ');
Writeln;
Write('       Use default value for Penalty Costs? (Y/N): ');
Readln(DefaultValue);

if (DefaultValue = 'Y') or (DefaultValue = 'y') then
         DefaultPCost
     else
         CustomizePCost;

{ Then, user will be asked the maximum Penalty Cost that can be tolerated }

Writeln;
Write('       Enter maximum penalty cost:  ');
Readln (MaxPCost);

{ User will be asked how many pattern s/he wants to generate }

ClrScr;
Writeln;
Write('       Enter number of patterns to be generated:  ');
Readln (NumOfPat);

CreatePattern;
CreateShiftBase;
ReorderShift;

repeat

ClrScr;
Writeln;
Write('       Enter number of schedules to be selected:  ',
      ' (Maximum = ',NumOfPat:4,' ):  ');

Readln(ScheduleNum);

PickFourWeekPattern;
EightWeekPattern;
TwelveWeekPattern;
CreateShift;
CreateSecondShift;
CreateThirdShift;
Create8WeekShift;
Create12WeekShift;


ClrScr;
```

```
Writeln;
Write('        Display schedules on the screen? (Y/N)  ');
Readln (SeeSchedule);

If (SeeSchedule = 'Y') or (SeeSchedule = 'y') then
   begin ClrScr; ViewTwelveSchedule; end;

ClrScr;
Writeln;
Write('        Display Staff List Report on screen? (Y/N)  ');
Readln (Display12Staff);

If (Display12Staff = 'Y') or (Display12Staff = 'y') then
   ViewStaffList12Week;

Writeln;
Write('       Save the schedules ? (Y/N)  ');
Readln (SaveSchedule);

If (SaveSchedule = 'Y') or (SaveSchedule = 'y') then
   SaveTwelveSchedule;

Writeln;
Write('       Save Staff List Report? (Y/N)   ');
Readln(SaveReport);

If (SaveReport = 'Y') or (SaveReport = 'y') then
   StaffList12Week;

Writeln;
Write('       Print the schedules? (Y/N)   ');
Readln(PrintSchedule12);

If (PrintSchedule12 = 'Y') or (PrintSchedule12 = 'y') then
   PrintTwelveSchedule;

Writeln;
Write('       Print Staff List Report? (Y/N)   ');
Readln(PrintReport12);

If (PrintReport12 = 'Y') or (PrintReport12 = 'y') then
   PrintStaffList12Week;

Writeln;
Write('       Select another set of schedules? (Y/N)   ');
Readln(PickAgain);

until (PickAgain = 'N') or (PickAgain = 'n');

end; { Procedure Length12 }

{*************************************************************************}

Procedure ThesisTitle;

VAR

   GraphDriver, GraphMode : Integer;
   xMax, yMax : Integer;

BEGIN

   GraphDriver := Detect;
   InitGraph( graphDriver, graphMode, '');
```

```
   SetBkColor(Blue);

   SetColor( White );
   SetLineStyle(SolidLn, 0, NormWidth);
   xMax := GetMaxX; yMax := GetMaxY;
   Rectangle(0,0, xMax, yMax );
   Rectangle(3,3, (xMax-3), (yMax-3));
   SetColor( Red );
   SetFillStyle( SolidFill, Yellow );

   SetTextStyle( SansSerifFont, HorizDir, 2);
   MoveTo( 198, 50);
   SetColor( LightRed );
   OutText( 'Decision Support System');

   MoveTo( 300, 80);
   OutText( 'for' );

   MoveTo( 45, 125);
   SetColor(White);
   SetTextStyle( TriplexFont, HorizDir, 7);
   OutText( 'Nurse Scheduling');

   SetTextStyle( SansSerifFont, HorizDir, 2);
   MoveTo( 250,210);
   OutText( 'Developed by');

   MoveTo( 198,235);
   SetColor(Yellow);
   SetTextStyle( GothicFont, HorizDir,4);
   OutText( 'Darwin Sitompul');

   SetTextStyle( SansSerifFont, HorizDir, 2);
   MoveTo( 244,325);
   OutText('Supervised by');

   SetColor(White);
   MoveTo( 206,350);
   OutText('Sabah Randhawa, PhD');

   SetColor(LightRed);
   MoveTo(197,400);
   OutText('Oregon State University');

   REPEAT UNTIL KeyPressed;
   CloseGraph

END;

begin {****************************Main Program*****************************}


ThesisTitle;

If (ReadKey = #13) or KeyPressed then

begin
TextBackground(1);
TextColor(14);
end
else
TextBackground(1);
TextColor(14);

ClrScr;
```

```
writeln('                                                              ');
writeln('          Welcome...                                          ');
writeln('                                                              ');
writeln('          This program generates schedules for nurses         ');
writeln('          in hospital using a heuristic method called         ');
writeln('          "Best-First Search Technique".                      ');
writeln('                                                              ');
writeln('          The program provides three options for the length   ');
writeln('          of schedule period and several options for          ');
writeln('          the hospital shift policies, but you may define     ');
writeln('          specific hospital shift policy to be used by        ');
writeln('          the program.                                        ');
writeln('                                                              ');
writeln('          The program will try to satisfy both nurses and     ');
writeln('          hospital requirements by minimizing the penalty     ');
writeln('          costs of the nurses'' and hospital''s constraints.  ');
writeln('                                                              ');
writeln('          You may select to either use the default penalty    ');
writeln('          costs built in the program, or to assign your       ');
writeln('          own penalty costs reflecting the specific work      ');
writeln('          environment.                                        ');
writeln('                                                              ');
writeln('Press <Enter> to continue...                                 ');
readln;
ClrScr;
writeln('                                                              ');
writeln('          ...                                                 ');
writeln('          You will first be asked to specify the number of    ');
writeln('          patterns you want to generate. It is recommended    ');
writeln('          that for flexibility you specify this number to be  ');
writeln('          greater than the number of schedules required.      ');
writeln;
writeln('          Once the program generates a set of patterns,       ');
writeln('          it will provide you with several set of schedules   ');
writeln('          from which you may select the number you need.      ');
writeln('                                                              ');
writeln('          You may view the schedules and the staff list       ');
writeln('          report before you decide to pick and save schedules ');
writeln('          or to discard the schedules and generate another    ');
writeln('          set.                                                ');
writeln;
writeln('          If you think that the patterns you have generated   ');
writeln('          do not satisfy your needs, you may generate         ');
writeln('          another pattern set.                                ');
writeln;
writeln('          Good Luck...                                        ');
writeln;
writeln('Press any key to continue or <Esc> to terminate ...          ');

if ReadKey = #27 then Exit
else
ClrScr;

{UserQuits := False;}

repeat

ClrScr;

writeln('                                                              ');
writeln('                                                              ');
writeln('                                                              ');
writeln('                                                              ');
writeln('                                                              ');
writeln('                                                              ');
```

```
writeln('                                LENGTH OF SCHEDULE :                ');
writeln('                                ___________________                 ');
writeln('                                                                    ');
writeln('                                1. Four-week.                       ');
writeln('                                2. Eight-week.                      ');
writeln('                                3. Twelve-week.                     ');
writeln('                                4. Exit                             ');
writeln('                                                                    ');
write('                               Select 1, 2 or 3 (4 for exit) :      ');
readln(LengthOfSchedule);

{UserQuits := False;}
Case LengthOfSchedule of

     1: Length4;
     2: Length8;
     3: Length12;
     4: Exit;

     else

     writeln('Error, try again! ');

end; {of Case}
ClrScr;
writeln;
writeln('          If the patterns generated do not satisfy the          ');
writeln('          staff requirement, the program can generate           ');
writeln('          another patterns and schedules.                       ');
writeln;
writeln;
writeln;
write('            Generate another patterns? (Y/N)  ');

readln(UserAnswer);

until  (UserAnswer = 'N') or (UserAnswer = 'n');

end.
```