1987

Doctor's office management system

Yong-gi Kim

The University of Montana

Follow this and additional works at: https://scholarworks.umt.edu/etd

Let us know how access to this document benefits you.

Recommended Citation
https://scholarworks.umt.edu/etd/2095

This Thesis is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact scholarworks@mso.umt.edu.
COPYRIGHT ACT OF 1976

This is an unpublished manuscript in which copyright subsists. Any further reprinting of its contents must be approved by the author.

Mansfield Library
University of Montana
Date: 1987
DOCTOR'S OFFICE MANAGEMENT SYSTEM

BY

Yong-Gi Kim
B.S., Seoul National University, 1978

Presented in partial fulfillment of the requirements
for the degree of
Master of Science
University of Montana
1987

Approved by

[Signatures]
Chairman, Board of Examiners
Dean, Graduate School
Date
July 27, 1987
The purpose of this project is to develop a software package for a Doctor's Office Management System. In implementation thereof, I will use a four step software life cycle; Requirements and Specification, Design, Implementation and Test, and Maintenance.

For Requirements and Specification, the methodology of Structured Analysis is used. This is represented with Data Flow Diagrams, a Data Dictionary, and a Process Description. The methodology used in the Design phase is Structured Design. The result of the Structured Design is a Structure Chart. Also, a Pascal-like module description is used to describe the algorithms. The program dBASE III is used to implement the system on a microcomputer. The testing of the system is done each time a module is completed during the implementation phase.

Most of the time and effort that was spent on this project fell in the Requirements and Specification phase. This stage follows every step of the Structured Analysis methodology. Once this analysis had been completed, the Design and Implementation phase were completed in a relatively short time.
Table of Contents

Abstract ii

Table of Contents iii

1. Introduction ............................................. 1

2. Requirements and Specification ................................ 7

   2.1. Methodology for Requirement and Specification .............. 8

       2.1.1. Procedure of Structured Analysis .................. 8

       2.1.2. The tools for Structured Analysis .................. 10

   2.2. Current Physical Model .................................. 14

       2.2.1. What is the Current Physical Model? ................. 14

       2.2.2. Production of the Current Physical Model .......... 14

                       for this project

   2.3. Current Logical Model .................................. 18

       2.3.1. Methodology for the Current Logical Model .......... 18

       2.3.2. Derivation the Current Logical Model .......... 19

                       for this project

   2.4. New Logical Model ....................................... 23

       2.4.1. Producing the New Logical Data Flow Diagram .......... 23

       2.4.2. New Logical Data Flow Diagram .................. 24
2.5. New Physical Model (Future System) ........................................ 25

2.5.1. Producing the New Physical Data Flow Diagram .......... 25
2.5.2. Context Diagram ................................................................. 26
2.5.3. Level-0 Data Flow Diagram ................................................. 27
2.5.4. Data Dictionary ................................................................. 29
2.5.5. Process Description ............................................................. 30

3. Design ....................................................................... 31

3.1. What is Structured Design? .............................................. 33
3.2. Structure Chart ................................................................. 34
3.3. Transaction Analysis ......................................................... 37
3.4. Module Description ............................................................ 41

4. Implementation ................................................................. 44

5. Test and maintenance .......................................................... 47

6. Conclusion ..................................................................... 50

Bibliography ....................................................................... 56
1. Introduction
METHODOLOGY USED IN THIS PROJECT

This project has involved the development of software which will automate some of the functions at Superior Clinic, Superior, Montana, which are being done manually. Among the functions of the clinic, appointment scheduling, billing, and maintenance of patient's records will be automated by this software.

The new system has been developed through the software lifecycle which is:

1. Requirements and Specification.
2. Design.
3. Implementation.
4. Testing and maintenance.

The methodology used in analysis phase is Structured Analysis developed by De Marco (1979). It is represented by Data Flow Diagrams, a Data Dictionary, and Process Descriptions. To produce a structured specification of the analysis phase of the project, the methodology requires seven steps. These steps are

1. Producing a Current Physical Model.
2. Producing a Current Logical Model.
3. Producing a New Logical Model.
5. Applying a cost-benefit study.
6. Selecting an option.
7. Packaging the specification.

The product of the first step is a Current Physical Data Flow Diagram which shows the current manual system. In order to produce the Current Physical Data Flow Diagram, determination of context, user interviews, collection of data files and forms, and walkthroughs with users should be executed.

The product of the second step is a Current Logical Data Flow Diagram. In order to produce the Current Logical Data Flow Diagram, the Current Physical Data Flow Diagram is expanded to remove physical characteristics from the Current Physical Data Flow Diagram by replacing it with its logical equivalent. Then, the current physical files and data stores are replaced by the logical file equivalents which are derived through the eight steps which will described in section 2.3.4. The Current Logical Data Flow Diagram, the product of the second step of the analysis phase, is constructed by combining the expanded Current Physical Data Flow Diagram in which all physical items are replaced by their logical equivalents, with the logical file equivalents.

The product of the third step of the analysis phase is the New Logical Data Flow Diagram. The New Logical Data Flow Diagram may add several more features to the Current Logical Data Flow Diagram for the future system.
In the next step, the man-machine boundary is selected. The man-machine boundary determines how much of this system will be automated and how much will remain manual. The man-machine boundary is determined through a cost-benefit study. The product of this step is a New Physical Data Flow Diagram.

The next step is applying a cost-benefit study. In this step, various kinds of hardware may be selected. Usually, a particular selection among the various kinds of hardware is made at the beginning of design phase.

The last step of the analysis phase is packaging the structured specification. The structured specification for this project consists of the Data Flow Diagrams, Data Dictionary, Process Descriptions and screen designs for the clinic management system.

The tools of each element of the structured specification will be described specifically in the next section.

When the Requirements and Specification is completed, the next step is the Design phase of the system. To produce code for the system, it is necessary to design the system since the code cannot be produced directly from the Data Flow Diagrams. The first reason why the code cannot be produced directly is that a Data Flow Diagram provides flow of data but not of control. Therefore, all of the processes in the diagrams seem to be independent of each other. Since such control is not represented in the
Data Flow Diagram, Data Dictionary or Process Description, the control information must be provided before the implementation phase. The second reason why the code cannot be produced directly is that the Data Flow Diagram and related specifications ignore most processing for exception or error conditions. So, in the Design phase, error handling is included. And finally, specifications never show the end of any input data stream. The details of the necessary steps for the end of the input condition are described in the design phase.

The methodology used in the design phase of this project is called Structured Design. Structured Design produces a top-down design. A structure chart is a graphic documentation tool for representing the results of a top-down design. There are two major techniques in Structured Design methodology used to derive the top-down design. These are transform analysis and transaction analysis. The next step after the structure chart is the specification of each of the modules in the chart. This is called the low level design. The technique used for the specification of each module for this project is a Pascal-like module description. For each of the primary modules in the structure charts, there is a module description of the task or function provided by that module. The module description almost appears as a programming language so that this can easily be changed to coding in the implementation phase. The Pascal-like description of the module facilitates the actual coding since the program is written directly from the module description. Each module of the structure chart has one module description and the module
description provides additional information. Therefore, the module description connects the structure chart and the actual implementation.

Several factors are examined to come up with a good design. The major factors to be considered to get a good design are coupling, cohesion, and the size of the modules. Coupling is the degree of interconnection between modules; therefore, the coupling should be as low as possible for maintenance of the modules. Cohesion is a measure of the strength of relatedness of the elements within a module; therefore, it is better to maximize the cohesion. The size of the modules should be small enough to minimize its complexity.

After the design phase is completed, the next step is implementation. The language which is used for this project is determined after considering several items, for example, size of the clinic and characteristics of the system. The selected language to implement this system is dBASE III. dBASE III allows an almost direct translation of the module description into implementation, and the language dBASE III provides high maintainability.

The testing of the system is done during the implementation phase. Top-down and Incremental Testing is used for the testing phase. This technique allows top level modules to be tested prior to the implementation of the bottom level design.
2. Requirements and Specification
2.1. Methodology for Requirements and Specification

2.1.1. Procedure of Structured Analysis

The methodology used in the Requirements and Specification phase for this project is Structured Analysis developed by Tom De Marco. This Structured Analysis methodology consists of seven components as shown in Fig. 2.1. According to Structured Analysis and System Specification of De Marco (1979), the seven steps of the analysis phase are:

b. Producing a Current Logical Data Flow Diagram through removing the physical items from the Current Physical Data Flow Diagram.
c. Producing a New Logical Data Flow Diagram by adding some new features to the Current Logical Data Flow Diagram for the future computerized system.
e. Applying a cost-benefit study to the set of Logical Data Flow Diagrams.
f. Selecting one New Physical Data Flow Diagram among the possible tentative New Physical Data Flow Diagrams through the cost-benefit
**Fig. 2.1. Process of Structured Analysis**

From *Structured Analysis and Systems Specification* (De Marco, 1979, p. 26)

1. **STUDY CURRENT ENVIRON.**
   - Current Physical Data Flow Diagram

2. **DERIVE LOGICAL EQUIVALENT**
   - New Logical D.F.D.

3. **MODEL NEW LOG. SYSTEM**
   - Data Dictionary

4. **ESTABLISH MAN_MACHINE I/F**
   - Physical D.F.D.

5. **QUANTIFY EACH OPTION**
   - Cost/Benefit Studies

6. **SELECT OPTION**
   - Physical Req.
   - Budget & Schedule

7. **PACKAGE SPEC**
   - Selected D.F.D.

---

**Steps in the Process:**

1. Study Current Environment
2. Derive Logical Equivalent
3. Model New Logical System
4. Establish Man-Machine I/F
5. Quantify Each Option
6. Select Option
7. Package Spec

**Feasibility Document**

**Structured Specification**

**User Req.**
study of step e.

The major outputs of this procedure are the Current Physical Model, the Current Logical Model, the New Logical Model, and the New Physical Model, which will be described later in this section.

2.1.2. The tools for Structured Analysis

Structured Analysis requires three types of analysis phase tools - a Data Flow Diagram, a Data Dictionary and a tool to describe logic and policy.

Data Flow Diagrams are made up of four basic elements as shown in Fig. 2.2. These elements are:

a. a named vector representing a data flow, which shows a data path by "-->".

b. a circle or a bubble representing a process, which shows transformations of data.

c. parallel straight lines representing a file or a database: ---

---

d. a box representing a data source and sinks, which show a net
originator or receiver of data - typically a person or an organization outside the domain.

The Data Flow Diagram shows flows of data, not of control. Therefore, in order to implement the project, we need a design which shows the control of data after this analysis phase and before the implementation.

Fig. 2.2. Example of Data Flow Diagram
A Data Dictionary is an ordered set of definitions of terms used in the Data Flow diagram. An example of a Data Dictionary is

\[
\text{Patient\_file} = \{ \underline{\text{Patient\_Name}} \ + \\
\hspace{1cm} \underline{\text{Patient\_Number}} \\
\hspace{1cm} \text{Address} \\
\hspace{1cm} (\text{Phone}) \\
\hspace{1cm} \text{Date\_of\_Birth} \\
\hspace{1cm} \text{Sex} \\
\hspace{1cm} \text{Doctor} \}\n\]

According to De Marco (1979, p. 133), the notations being used in the Data Dictionary are:

a. \( = \) means IS EQUIVALENT TO
b. \( + \) means AND
c. \( [ \) means EITHER-OR
d. \( ] \) means ITERATIONS OF the component enclosed
e. \( ( ) \) means that the enclosed component is OPTIONAL
f. \( \underline{\text{---}} \) means that the underlined item is an indexed key of the file or data base.

The tool used to describe logic and policy is Structured English. Each bubble on the Data Flow Diagram is followed by a description of what it does in Structured English. Thus, for each bubble there follows a detailed
description. An example of Structured English follows.

------------------------------------------
Schedule_Appointment
Process 1.5
------------------------------------------

Accept Date from screen.
Access Appointment_file using date as key.
For each date of records of Appointment_file:
    Display the record of Appointment_file on the screen.
Accept Patient_Name, Time, Doctor, and Reason.
Accept Schedule_Confirm.
    IF Schedule_Confirm is true
    THEN Add a new record to Appointment_file using
        the Patient_Name, Date, Time, and Doctor.

------------------------------------------
2.2. Current Physical Model

2.2.1. What is the Current Physical Model?

The Current Physical Data Flow Diagram is the output of the first step of the procedure of the Structured Analysis methodology. To produce the Current Physical Data Flow Diagram, the current physical environment of the current system should be inspected and transformed into a data flow diagram. In other words, the Current Physical Data Flow Diagram shows the data flows of the current system. In this Current Data Flow Diagram, physical characteristics, such as department names, locations, names of related persons, files or data stores are included. To make up such a current physical study, determination of context, user interviews, collection of sample data types, files, and forms should be executed before the Current Physical Data Flow diagram is produced.

2.2.2. Production of the Current Physical Model for this project

The current manual system of the clinic consists of largely four functions. These are the functions of the front desk of the clinic, the doctor, the nurse, and the accountant. When a patient visits the clinic, he or she fills out a form dealing with his or her vital statistics. The
information is copied onto a report form in triplicate, henceforth called a diagnosis paper for purpose of this project. On these papers, the doctor enters his specific description of the diagnosis. One copy will be sent to an accountant for calculating the service charge and billing, and the second copy will be sent to the front desk of the clinic for filing as a permanent history file of the patient. This is referred to by the doctor whenever the patient visits the clinic. The third copy is sent to a hospital nearby if the patient is moved to that hospital. Since the clinic is attached to the hospital, it has use of all their laboratory equipment. In the billing process, an insurance form is mailed to the patient’s insurance company. The difference between the payment from the insurance company and the service charge will be mailed to the patient in the form of a bill which is issued once a month. Appointment scheduling, however, is a separate function from this process. During the process from a patient’s visiting the clinic to payment of the service charge, some information is used by the system more than two or three times.

In order to produce the Current Physical Model, necessary forms such as an appointment book (used for scheduling appointments), a patient information card (filled out by the patient when he or she visits the clinic for the first time), a diagnosis form, an accounting book (showing all patient’s accounting statuses), an insurance form, and a statement were collected from the clinic. To collect this information, it was necessary to interview the users, the receptionist for the front desk, a nurse, an
accountant, and the doctor. But the size of the clinic is small enough for
the doctor to understand the functions of each job. Therefore, the doctor
can determine the context of this system, and the necessary information
required for computerization of the clinic was obtained through the doctor.

In order to derive the Physical Data Flow Diagram, the context should
first be determined. Then, circles for the four functions of the clinic,
which were described previously, with each function name inside the
circle, should be drawn. The next step is to identify the net inputs and net
outputs. The net inputs are incoming data flows from outside the system
into the system. The net inputs of this system are:

- appointment inquiry : from patient to front desk.
- patient information : from patient to front desk.
- payment : from patient to accountant.
- account inquiry : from patient to accountant.
- payment : from insurance company to accountant.

The net outputs are outgoing data flows from the system to outside the
system. The net outputs of this system are:

- appointment : from front desk to patient.
- marked diagnosis paper : from doctor to hospital.
- account information : from accountant to patient.
- bill : from accountant to patient.
- insurance form : from accountant to insurance company.
The person(s) or originator outside the system who receives the data from the system or dispatches the data into the system is represented by a rectangular box with his or her name in the box. Then the functions of the system, represented by circles, and the sources outside the system, represented by rectangular boxes, are connected with named vectors. The vector is a data flow. The next step is drawing the vectors between the circles, which are data flows between the functions inside the system. Next is the drawing of files accesses or data storage. When a function accesses a file to get data, then the arrow of the vector is from the file to the function; when the function accesses the file to store data, then the arrow is from the function to the file. A double headed arrow between the function and the file or data storage is drawn when the function accesses the file to get and to store data.

The Data Flow Diagram and Data Dictionary for the Current Physical Model is shown in Appendix A.
2.3. Current Logical Model

2.3.1. Methodology for Current Logical Model

The Current Logical Data Flow Diagram is the product of the second step of the procedure of the Structured Analysis methodology shown in Fig. 2.1. The Current Logical Data Flow Diagram is produced by replacing each physical item in the Current Physical Data Flow Diagram with its logical equivalent. De Marco's definition of the words "physical" and "logical" in his *Structured Analysis and System Specification* (1979) is as follows. The word "physical" refers to a description that is implement-dependent, while the word "logical" refers to one that is implementation-independent. In other words, the logical-physical distinction is the difference between what is accomplished, and how it is accomplished. According to the methodology, the derivation of the Current Logical Model from the Current Physical Model consists of three steps. The first step is building an expanded Data Flow Diagram to remove the highest level(s) of physical characteristics. The second step is decomposing and normalizing the file structures into their most logical forms, and the last step is combining the expanded data flow diagram and the logicalized files. The derivation of the Current Logical Model from the Current Physical model for the Doctor's Office Management System is described in the next section.
2.3.2. Derivation of Current Logical Model for this project

The first procedure of "logicalization" is the development of an expanded Data Flow Diagram which removes physical characteristics of the Current Physical Data Flow Diagram by replacing bubbles at level \( n \) with an \( n+1 \) level network and replacing the physical items with their logical equivalents. Fig. 2.3 is a part of the Current Physical Data Flow Diagram of the current system, and Fig. 2.4 is the expanded Data Flow Diagram of Fig. 2.3. Process 1 of the Current Physical Data Flow Diagram of the current system, which is shown in Fig. 2.3, is expanded in Fig. 2.4 to remove physical items of the jobs of the front desk of the clinic and replace the items with logical equivalents, which are shown in Fig. 2.4. The jobs of the front desk of the clinic are: scheduling new appointments, finding an existing appointment to answer an inquiry from a patient, canceling an existing appointment, modifying an existing appointment, getting a patient information card filled out by the patient, and filling out the diagnosis papers by copying patient information from the patient information card, which will be sent to the doctor for use during the diagnosis of the patient. The "logicalized" items of the front desk job can be expressed as a Current Logical Data Flow Diagram as shown in Fig. 2.4. The whole expanded Data Flow Diagram from the Current Physical Data Flow Diagram is in Appendix B.
Fig. 23 A part of the Current Physical DFD
Fig. 2.4. Expanded Data Flow Diagram from Fig. 2.3
The second procedure of "logicalization" is deriving logical file equivalents from the current physical files or data storages. De Marco's (1979) Structured Analysis methodology describes this procedure in eight steps. These are:

1. Classifying all accesses to data storage. For each access, data flow and direction is recorded. As shown in the expanded data flow diagram of Appendix C, all references to stored data are assigned a unique number.

2. Classifying all accesses into incoming and outgoing data flows. All required data elements and key elements of the flow are recorded.

3. Decomposing the file structure into private component files.

4. Eliminating internal repeating groups in each file structure.

5. Separating the field of the record from its correlations.

6. Combining the same items into common component files.

7. Removing any stored data element that can be derived by reference to other stored data elements.

8. Packaging the normalized file structure.

The application of this procedure to derive logical file equivalents from the Current Physical Model of the Doctor's Office Management system is shown step by step in Appendix C. The results of this application to current physical files and data storages are six logicalized files: Appointment_file, Patient_file, Diagnosis_file, Account_file, Payment_file, and Fee_file. The components of these files are shown in
The Current Logical Data Flow Diagram can finally be drawn by combining the expanded Data Flow diagram shown in Appendix B with the logicalized files. This Current Logical Data Flow diagram is shown in Appendix D.

2.4. New Logical Model

2.4.1. Producing the New Logical Data Flow Diagram

Some more features are added to the Current Logical Model for the future computerized system. This new enhanced Data Flow Diagram is called a New Logical Data Flow Diagram. In order to build the New Logical Data Flow Diagram for the Doctor's Office Management System, several commercially available systems for clinic management have been examined. Most of them have similar functions in meeting a typical clinic's requirements, which are appointment scheduling, billing, insurance form printing, practice management, and medical procedures. Several features, which the current manual system of the clinic does not have and should be needed in the future computerized system, were extracted from this study. These are:

- printed appointment schedule
open time appointment schedule
A/R aging (Aging Report)
account summary report for clients
analysis of practice
permanent history file update.

However, the commercially available systems could not be applied directly
to the clinic since the clinic is in a special situation: It is attached to a
hospital so that it uses all laboratory equipment of the hospital, and the
clinic is comparatively too small in size.

2.4.2. New Logical Data Flow Diagram

The New Logical Data Flow Diagram for this project is shown in
Appendix E.
2.5. New Physical Model

2.5.1. Producing the New Physical Data Flow Diagram

Since the New Logical Data Flow Diagram does not distinguish between automated and manual procedures, a man-machine boundary must be set up for the future system. Therefore, in the future Data Flow Diagram, the manual part should be eliminated from the New Logical Data Flow diagram. This Data Flow Diagram from which the manual procedure has been deleted is called the New Physical Data Flow Diagram. The man-machine boundary is determined by separating out components of the New Logical Data Flow Diagram. Thus, process 4.1 of that diagram is eliminated in the Data Flow Diagram for the future system. Therefore, the process of calculating the service charge will remain as a manual procedure from a cost-benefit perspective. However, the very common service charges, which are about 80 percent of the total service of the clinic, are determined and stored by the computer system.

The New Physical Model consists of the Data Flow Diagram shown in Appendix F, the Data Dictionary shown in Appendix G, the Process Description shown in Appendix H, and the Screen design for the New system shown in Appendix I.
2.5.2. Context Diagram

The Context Diagram of the New Physical Model is shown in Appendix F. De marco (1979) defines the Context Diagram as a "formal declaration of the domain" (p. 75) of the system. The domain of this project is shown by a circle named as SYSTEM. In the Context Diagram, incoming arrows are net inputs to the system, and outgoing arrows are net outputs of the system to the user. The net inputs to the SYSTEM are patient service charge, payment from patient and/or insurance company of the patient, account inquiry from the patient, appointment inquiry, patient information which is filled out by the patient and shows the vital statistics of the patient, and a patient's history update after service. The net outputs of the SYSTEM are the aging report which shows all of the patient's aged service charges for a given month, the monthly accounting report which shows all of the patient's accounting statuses for a given month, the insurance form which will be mailed to the patient's insurance company, a print-out of all appointment statuses for a given date, a print-out of the result of the history update of a patient after service, bills which will be mailed to the patients, appointment confirmations, and answers to account inquiries from patients which show the patient's cumulative accounting statuses.
2.5.3. Level-0 Data Flow Diagram

The Level-0 Data Flow Diagram of this system is shown in Appendix F. The Level-0 Data Flow diagram consists of five processes which are Appointment, Billing, Diagnosis, Maintaining_history, and Backup. Among these processes, the Appointment process is separated from the others, and the Appointment_file is accessed only by this Appointment process.

In the Appointment process, an Appointment_Inquiry is accepted by a patient. This Appointment_Inquiry is classified into five functions as shown in DIAGRAM 0 of Appendix F: scheduling a new appointment, finding an existing appointment, canceling an existing appointment, modifying an existing appointment, and printing existing appointments for the day. The outputs of the Appointment process are the appointment itself (which is the answer to the inquiry from the user) and a printed appointment_sheet. These decomposed functions are shown in DIAGRAM 1 of Appendix F, the under level diagram of the DIAGRAM 0.

The Billing process is the key process of this system. This process accepts Patient_Charge, Payment from a patient or his insurance company, and Account_Inquiry from a patient as inputs. The outputs of this billing process are the monthly accounting report, the accounting status information of a patient, the bill, the insurance form, and the aging report of all patient service charges.

The main functions of the Billing process are:
1. Finding the patient number - since all communication with the system will be done through the patient number.

2. Entering the patient's service charge into the system.

3. Printing the insurance form to send to the insurance company of the patient.

4. Printing bills to send to the patients.

5. Entering the payments from the patient and/or the patient's insurance company.


7. Printing all aged charges of all patients for a month.

8. Printing a monthly accounting report which shows the accounting status of all patients of the clinic of a given month.

These functions of the billing process are decomposed into DIAGRAM 2 of Appendix F.

The process 3.0 of the Level-0 Data Flow Diagram is for entering patient information into the system when a patient visits the clinic for the first time, and for searching a patient's permanent history, which has already been input in the Maintaining_History process and which will be referred to by the doctor during his diagnosis of the patient. The inputs to this process are patient information, and the patient history request. The outputs are the diagnosis paper (which involves the patient information
and which will be used by the doctor for diagnosis of the patient) and the patient history, which is stored in the Diagnosis_file for answering patient history requests. This Diagnosis process is decomposed as shown in DIAGRAM 3 of Appendix F.

The Maintaining_History process accepts updated history of a patient and accesses the Diagnosis_file to update the information of the diagnosis, and to print the updated history. Also, this process produces a monthly service report which shows the practice statistics of the clinic during the given month. This Maintaining_History process is decomposed as shown in DIAGRAM 4 of Appendix F.

2.5.4. Data Dictionary

The Data Dictionary is a set of definitions of data flows and files. Each of the data flows, data elements, and files of the Data Flow Diagram are explained rigorously in the Data Dictionary. Here, the data element is an element of a data flow or of a file, and the data element cannot be decomposed into subordinate data flows. The symbols used in the Data dictionary are described in section 2.1.2.

The Data dictionary for the Doctor's Office Management system is shown in Appendix G.
2.5.5. Process Description

The Process Description describes each process of the Data Flow Diagram. According to De Marco (1979), there are three kinds of tools to describe logic and policy which are better than narrative text. These are the Decision Tree, the Decision Table, and Structured English. Among these tools, Structured English was selected for this project. Structured English is a specification language that makes use of a formal grammar and limited words which consists of imperative verbs, terms defined in the Data Dictionary, and certain reserved words for logic formulation. The logical constructs of Structured English consist of three building blocks: sequence, repetition, and decision. The structured English for this project is shown in Appendix H.
3. Design
Design is a process that produces a method of solution for the system to achieve the best implementation. A design of a system determines the major characteristics of the system, and facilitates the implementation of the system. To produce the code for the system, it is necessary to have a design of the system completed since the code cannot be derived directly from the Data Flow Diagram and the Data Dictionary, which are the products of the analysis phase. The design should be constructed for ease of maintenance and testing of the system since the cost of manpower continues to grow and machine costs are decreasing, and most of the cost of lifetime software falls into the maintenance of the system. For ease of maintenance, some design features are considered. The main features are that:

1. Module size should be small.

2. Each module should be independent of the others.

The methodology used in the design phase of the Doctor's Office Management System is Structured Design. The main reason for selecting the methodology is that the analysis phase of this project was developed using Structured Analysis, and the Structured Design methodology is a strategy to convert the Data Flow Diagram, which is a product of Structured Analysis, into the structure chart. The structure chart, which is the documentation tool for Structured Design, and the Pascal-like module description, which is a low level design of the module, will be described in this section.
3.1. What is Structured Design?

Structured Design is a strategy for producing a highly maintainable, easily tested top-down design. According to Yourdon's (1979) *Structured Design*, the definition of Structured Design is as this: "Structured Design is the art of designing the components of a system and the interrelationship between those components in the best possible way" (p. 8).

Structured Design produces a document, the structure chart, by converting the Data Flow Diagram. To make up a good Structured Design, one should consider design heuristics, such as coupling and cohesion. Coupling is a measure of interconnection between modules. When the degree of coupling is high, the change of one module affects the functioning of another. Therefore, coupling must be minimized so that the structure can be maintained easily without hurting the functioning of another module. According to Youdon (1979), there are four major considerations of computer systems which affect coupling. These are:

1. Strength of connection between modules. When the modules are connected with low interdependence, the coupling is low.
2. Complexity of the interface. The number of items which are being passed between modules affects the coupling. The more the number, the higher the coupling.
3. Type of information flow along the connection.
4. Binding time of the connection.
The other factor which should be considered to make up a good Structured Design is cohesion. Yourdon's (1979) definition of cohesion is "the degree of functional relatedness of processing elements within a single module" (P. 447). When a module does only one task, or does several related tasks which are grouped by using the same data items, the module has high cohesion. The cohesion should be maximized to come up with a good design. However, cohesion and coupling are interrelated. When cohesion of individual modules in the system is high, coupling between modules will be low.

In order to produce the structure chart, which is the result of the Structured Design, transform analysis and transaction analysis techniques are used. The techniques derive the structure chart from the Data Flow Diagram, which is a product of the analysis phase. These techniques and their application to this project will be described briefly later in this section.

3.2. Structure Chart

A structure chart is a graphic tool for representing the results of the top-down design. The basic elements used in the structure chart for the design of this project are:

a. The module, represented by a rectangular box with a name on the inside.
A module is a named set of statements.

b. The connection, represented by a vector joining two modules. A connection is any reference between modules.

c. The couple, represented by a short arrow with a circular tail. A couple is a data item that moves between modules.

d. The module, (which has already been produced,) represented by a circle.

e. The selection, represented by a diamond. The selection chooses one of the modules which are under the diamond. The diamond is the transaction center, which will be described in section 3.4.

f. The loop, represented by a half circle. The loop means repetition of the modules under the half circle.

These notations used in the structure chart for this project are shown in Fig. 3.1.

An example of the structure chart is in Fig. 3.2 and the explanation follows. The Option is accepted from module 4.1, and module 4.0 selects module 4.2 or 4.3 depending on the Option accepted from module 4.1. Then, the selected module is executed. This process will be repeated until the Option meets the end condition. However, this structure chart needs more explanation, for example, about the identity of the end condition, and what the modules do. The specific explanation of each module is described in the module description, which is called a low level design, in section 3.5. This structure chart is derived from the Data Flow Diagram, which is
Fig. 3.1 Notations in the Structure Chart

Fig. 3.2 An example of the Structure Chart
the product of the analysis phase, using the transform analysis and transaction analysis techniques. For this project, the transaction analysis technique was primarily used to derive the structure chart, since the Doctor's Office Management System is a menu driven system. The transaction analysis technique will be explained in the following section.

3.3. Transaction Analysis

"A transaction is any element of data, control, signal, event, or change of state that causes, triggers, or initiates some actions or sequence of actions"(Yourdon, p. 224), and transaction analysis is "a design strategy in which the structure of a system is derived from an analysis of the transactions the system is required to process"(Yourdon, p. 463). The Doctor's Office Management System is a menu driven system, and in order to select any function to be executed, the control to select the function must be accepted from the screen. Therefore, this project has made considerable use of the transaction analysis technique to convert the Data Flow Diagram into the structure chart.

An example which shows the application of the transaction analysis technique from the Doctor's Office Management System is in Fig. 3.3.1 and Fig. 3.3.2. The Fig. 3.3.1 is a level-1 Data Flow Diagram of the system, and the Fig. 3.3.2 is the structure chart derived from the level-1 Data Flow Diagram of Fig. 3.3.1 using the transaction analysis technique. The first
step in deriving the structure chart from the Data Flow Diagram is to determine the transaction center. The transaction center gets a control item, analyzes it, and dispatches the control to the transaction level, which is below the transaction center, depending on the analysis of the control item. In the Data Flow Diagram of Fig. 3.3.1, Process 1.1 becomes the transaction center in the structure chart of Fig. 3.3.2. The transaction center is located on the top level in the shape of a rectangular box with a diamond. In addition, the top level itself, as a transaction center, has a transaction level, which receives a control item, below the transaction center. In the next step, all the processes, 1.2, 1.3, 1.4, 1.5, and 1.6, are located under the diamond, and the transaction center dispatches control to their corresponding modules. During the process, all of the circles in the Data Flow Diagram, and the newly created module which receives the control, are changed to rectangular boxes. The top level and each transaction level rectangular box are connected with a straight line. Then, all the items and controls which are being passed from one module to another are represented by a short arrow with a circular tail.
Fig 3.3.1. Data Flow Diagram for the Structure Chart of Fig 3.3.2

1.2 Find Appointment

1.3 Cancel Appointment

1.4 Modify Appointment

1.5 Schedule Appointment

1.6 Get Patient Info

1.1 Classify Appointment

Appointment

Appointment

Appointment

Appointment

Appointment

Appointment

Appointment

Appointment
Fig. 3.3.2. Structure Chart derived from Fig. 3.3.1
using Transaction Analysis
3.4. Module Descriptions

However, in some cases, it is hard to produce a programming source code directly from the structure chart since the structure chart does not show, for example, the end of the input stream or error handling. Therefore, we need a more detailed low level design tool which can be transformed into a code almost directly. For this purpose, the Pascal-like module description is written for this project. Each module of the structure chart has one module description. An example of the structure chart is in Fig. 3.4, and the module description of the structure chart is shown in the following Fig. 3.5.

Fig. 3.5. Module description derived from the structure chart of Fig. 3.4

MODULE NAME: System MODULE NUMBER: 0.0

procedure System;
begin
   Display_Menu(Option);
   case Option of
      'A': Appointment;
'B': Billing;
'D': Diagnosis;
'M': Maintain History;
'E': End;

end case;
end;

The module description for the Doctor's Office Management System is in Appendix K.
Fig. 3.4. Structure Chart for the Module Description of Fig. 3.5
4. Implementation
Once the analysis and design are done, the implementation is relatively simple. The module description, the low level design of the system, is almost a programming language; therefore, the transformation to coding is a kind of converting from one programming language to another unless the design involves some structural error.

The programming language for implementing the system is selected after considering several fundamental items. The selected implementation tool is dBASE III. The main reasons for selecting dBASE III are:

a. The size of the clinic is relatively small, having about 1000 patients. Therefore, a microcomputer can handle the data of the clinic.

b. The Doctor's Office Management System is a totally business-related system, and the system should provide high maintainability and easy access to data files.

c. The Doctor's Office Management System is a menu driven system. The language of dBASE III is designed conveniently to handle control from the screen.

d. dBASE III uses data base files in the system. One of the advantages of using this data base is that we can index more than one key in one record.
The source code for implementing this project is in Appendix L.

The user's manual of this system is in Appendix M.
5. Test and Maintenance
The testing strategies used in the development of this project are the incremental testing strategy and the top-down testing.

The incremental testing strategy can be applied to the development of the system in the following order as stated by Yourdon (1979):
1. Design, code, and test one module by itself.
2. Add another module.
3. Test and debug the combination.
4. Repeat the steps 2 and 3. (p. 377-378).
This approach adds only one new module to the system at a time.

The top-down testing strategy approaches the system level by level from the top to the bottom level. In the example of Fig. 5, the order of testing is A, B, C, D, E, F, and G. Because of the order of the testing, the dummy module is a necessary one whenever an upper level module is tested. In other words, when an upper level module calls a lower level module, and the lower level module does not yet actually exist, then the dummy module is a necessary one for the lower level module.

The testing of this system is done during development of this system through the two testing strategies.
Fig. 5. Example of a system for testing.
6. Conclusion
The main purpose of this project was to apply Structured Analysis and Structured Design methodology to the real world, especially with respect to the Requirements and Specification phase. I tried to apply the Structured Analysis methodology developed by De Marco as closely as possible. Thus, most of the time spent on this project fell into the Requirements and Specification phase. Once the analysis phase had been completed, the Design and Implementation phase took relatively little time. The result of the Design phase could be transformed to coding without difficulty. The primary cause of bugs in the coding phase was due not to design error, but to lack of familiarity with the implementation tool, dBASE III, itself.

This project was initiated in the summer of 1986, and it carried through the summer of 1987. In my proposal for this thesis project, it was projected to be done by the end of winter quarter of 1987. The schedule in the proposal was as follows:

- Requirements and Specification... September, 1986
- Design ... December, 1986
- Coding and Testing ... February, 1987

But, actually, it took much more time than the scheduled time to finish the Requirements and Specification phase. The reason why I could not finish the analysis phase on schedule is that I proceeded to the analysis and design phase without following the Structured Analysis methodology step
by step. I started the Requirements and Specification phase in July, 1986, and the Analysis phase and Design phase were done November, 1986 and January, 1987. At that point, my first thesis committee meeting was held, and the committee decided that I should follow the Structured Analysis methodology step by step for educational purposes. So, my analysis phase was to restart in January of 1987. In this reworking of the analysis phase, I followed every step of the analysis methodology. During this phase, my committee, consisting of Professor Spence L. Manlove, Dr. Lee Tangedahl, Dr. James Ullrich, and Professor Craig Schiedermayer, met almost every week, and inspected my work which had been done during the past week. With the help of the committee, I collected and referred to some magazines and pamphlets for computerization of medical systems. The major items which were added to my system by the aid of committee meetings were

- printed appointment schedule
- open time schedule
- A/R aging
- account summary report for clients
- structured analysis of practice.

Through the repetition of inspections and approvals of each step from the committee meetings of every week, the analysis phase was completed at the end of April of 1987. The major outputs of this phase were the Current Physical Model, the Current Logical Model, the New Logical Model, the New Physical Model, the Data Dictionary, and the Process Description.
One problem during the development of the Requirements and Specification phase was that the clinic is located about 60 miles from Missoula; therefore, whenever I had some questions about the system, I had to wait for available time to visit the clinic or call the clinic to get information or solve problems. It was the other reason for delay of completion of the analysis phase. The person whom I have contacted for information about the requirement of the system is mainly Dr. Park of the clinic. The clinic is small enough for one to understand the management system very well and he was very cooperative in helping me to finish this project. After the analysis phase had been done, it did not take much time to modify the design which had already been done before the rework of the analysis phase. In the design phase, I decomposed each module into such small components that each one could have almost only one function in accordance with the top-down design principle. And I tried to minimize coupling by reducing the number of items being passed between modules, and to maximize cohesion by making modules do less than two or three related tasks which use the same data items. This phase was done in early May of 1987. Implementation of the system following the design phase took place at the same time as the rework of the analysis phase. In the implementation phase, I found that the highly decomposed design was not convenient to use with dBASE III to implement the system directly. So some of the modules were re-composed for convenient implementation of
the system. This phase was done at the end May of 1987. But with the writing of the report and the user's manual of this project, the program has been modified continuously until June of 1987.

This project was chosen for my thesis project, since the project looked appropriate for the application of the Structured Analysis and the Structured Design methodology, and seemed to be of an appropriate size to be selected as master's degree thesis project. But, as De Marco indicated in his book Software Project Controlling (1982) that most system analysts tend to underestimate the size of a system before they start to develop the project, I found I had underestimated the size of this project during development. I was supposed to finish the project by the end of March, 1987, but it took much more time. Even though I scheduled and started this project aware of the problem of underestimation, it still happened. Therefore, one of major things which I learned from the development of this project is never to underestimate the size of the project to be developed. Also, when we estimate the project to be developed, we had better apply Boehm's (1981) COCOMO estimates, or at least, estimate the schedule or size bigger than it looks.

The other thing I learned from the development of this project is that the language dBASE III does not allow a big single source code larger than about 300 lines, since when the source code is more than that much, the
code will be lost. But, very often, we need a large source code. To solve this problem, each module should be decomposed into a small size, and before that, we should apply the top-down design so that each module has a small sized code.

Even though I met some problems during the development of the system, those are not due to the defects of the analysis or the design methodology, but largely to unfamiliarity with the methodologies. If I had an opportunity to develop this kind of project once more, I would develop the system with using the analysis and the design methodologies again, but more smoothly and in shorter time. My experience of the development of this project says the Structured Analysis and the Structured Design methodologies will fit any kind of system development as long as they are being used jointly.

However, I was able to apply the Structured Analysis and Structured Design methodologies successfully to this project, with many other possible uses, from a software engineering perspective. And the users were satisfied with the output of this project even though the clinic does not yet have a computer to implement this system. This project was a most valuable and educational experience for me.
Bibliography
De Marco, T. *Structured Analysis and Systems Specification.*

Yourdon, E., and Constantine, L. *Structured Design.*

Robbins, J., and Braly, K. *Expert dBASE.*

Castro, L., Hanson, J., and Rettig, T. *Advanced Programmer's Guide,*
featuring *dBASE III* and *dBASE II.* ASHTON.TATE: 1985.

Boehm, B. *Software Engineering Economics.*

De Marco, T. *Software Project Controlling.*
Appendix A.

Current Physical Data Flow Diagram and Data Dictionary
Physical DFD of Current System

1. Front Desk
   - Appointment Inquiry
   - Patient Information
   - Appointment file
   - Patient Information file

2. Nurse
   - Patient History Request
   - Patient History
   - Permanent History file

3. Doctor
   - Diagnosis Paper (3 copies)
   - Marked Diagnosis Paper 1
   - Marked Diagnosis Paper 2
   - Marked Diagnosis Paper 3

4. Accountant
   - Fee Book
   - Patient Information
   - Payment
   - Insurance
   - Accounting Book(file)
   - Insurance Company

Hospital

Patient Information
Data Dictionary of Physical Model of the Current system

1. Appointment:

   Patient_Name • Date • Time • Doctor

2. Appointment_Inquiry

   Patient_Name • Inquiry_Type • Appointment_Date • Doctor

3. Patient_Information

   Patient_Name + Date_of_Birth + Address + (Phone) • S.s* + Marital_Status + Payment_Person + Parent/Spouse + Employer + emergency_Call + Rel_Patient • phone_No + Insurance_Info + Insurance_Company + policy* + Wife_or_Mother_Employed_by

4. Diagnosis_Paper

   Patient_Name + Date_of_Birth + Sex + Address + Phone + S.s* + Spouse/Parent + Insurer + Other_Insurer + Medicare + Visit_Date + Ins_Cert* + Ins_Grp*

5. Marked_Diagnosis_Paper

   Patient_Name + Date_of_Birth + Sex + address + Phone + S.s* + Spouse/Parent + Insurer + Other_Insurer + Medicare + Visit_Date + Ins_Cert* + Ins_Grp* + (Diagnosis) + Clinical_Notes

6. Patient_History_Request

   Patient_Name

7. Patient_History

   (Patient_Name + Date_of_Birth + Sex + Address + Phone + S.s* + Spouse/Parent + Insurer + Other_Insurer + Medicare + Visit_Date + Ins_Cert* + Ins_Grp* + (Diagnosis) + Clinical_Notes) (Permanent History file)
8. Account_Inquiry

Patient_Name

9. Account_Info

Patient_Name + S.s* + Address + Phone + (Date + Charge) +
(Payment_Date + Payment) + Total_Charge + Total_Payment + Balance + Resp_Party

10. Bill

Patient_Name + Address + (Visit_Date + Service + Charge) +
(Payment_Made) + Total_Charge + Total_Payment + Balance

11. Payment

Date + Payment_Type + Payment

12. Insurance_form

Patient_Name + Medicare + Date_of_Birth + Insured_Name + address +
Sex + S.s* + (Diagnosis) + (Visit_Date + Service) + Charge) +
Total_Charge + Total_Payment + Balance + Previous_Illness +
Date_Patient_Ableto_Return_To_Work + Date_of_Total_Disability +
Doctor + Hospitalization_Dates
Data Elements of Physical Model of the Current System

1. Patient_Name
   Name of the patient. Last name and first name.

2. Date
   Date of appointment. In the form of MM/DD/YY.

3. Time
   Time of appointment.

4. Doctor
   Name of the doctor of the patient.

5. Inquiry_Type
   Find, Schedule, Modify or Cancel an appointment.

6. Appointment_Date
   Date of the appointment.

7. Date_of_Birth
   Date of birth of the patient. In the form of MM/DD/YY.

8. Address
   Current address of the patient including zip code.

9. Phone
   Phone number of the patient.

10. S.s*
    Social security number of the patient.

11. Marital_Status
    Marital status of the patient. Single, married, widow, divorced or child.

12. Payment_Person
    Person who is responsible for the payment of medical fee.
13. Parent/Spouse
   Name of parents of the patient if child or full name of spouse if married.

14. Employer
   Name and address of employer of the patient.

15. Wife_or_Mother_employed_by
   Name and address of the employer of the parent or spouse of the patient.

16. Emergency_Call
   Phone number in case of emergency.

17. Rel_Patient
   Relationship to the patient.

18. Phone_No
   Phone number of the person who wrote the patient information.

19. Insurance_Info
   Insurance information for the patient.

20. Insurance_Company
   Name and address of the insurance company of the patient.

21. Policy*
   Insurance policy number of the patient.

22. Sex
   Sex of the patient. Male or female.

23. Insurer
   Name and address of the insurance company which will pay the medical fee for the patient.
24. Other_Insurer
   Name and address of the insurance company which will pay the medical fee for the patient besides insurer of 23.

25. Medicare
   Medicare number.

26. Visit_Date
   Date the patient visited the clinic for the disease.

27. Ins_Cert*
   Insurance certificate number.

28. Ins_Grp*
   Insurance group number.

29. Diagnosis
   Result of diagnosis including blood pressure, temperature, weight, and pulse. In case of common disease, it is marked on the table of the diagnosis_paper.

30. Clinical_Notes
   Specific description of the diagnosis made by the doctor.

31. Charge
   Dollar amount.

32. Payment_Date
   Date the payment is made.

33. Payment
   Dollar amount.

34. Total_Charge
   Dollar amount. Total service charge of the patient so far.
35. Total_Payment
   Dollar amount. Total payment of the patient so far.

36. Balance
   Dollar amount. Difference between Total_Charge and Total_Payment.

37. Resp_Party
   The name of the person who is responsible for the payment of the medical fee of the patient.

38. Service
   The type of service performed by the doctor.

39. Payment_Made
   Payment made by the patient so far.

40. Payment_Type
   Type of payment. Cash, check or credit.

41. Date_Patient_Able_to_Return_To_Work
   Expected date which the patient is able to return to work.

42. Date_of_Total_Disability
   Dates of disability of the patient because of the disease.

43. Hospitalization_Dates
   Dates of hospitalization.
Appendix B.

Expanded Data Flow Diagram from
Current Physical Data Flow Diagram
Expanded Data Flow Diagram for Current Logical Model

With Physical files 1

1.2 Find_Appointment

1.3 Cancel_Appointment

1.4 Modify_Appointment

1.5 Schedule_Appointment

1.6 Get_Patient_Info

Patient Information

Diagnosis_Paper

Appointment

Appointment_Inquiry

Find_Inquiry

Cancel_Inquiry

Appointment_book

Appointment_book

Schedule_Inquiry

Modify_Inquiry

Classify_Appointment
Expanded Data Flow Diagram for Current Logical Model

With Physical files 2

- 3.1 Request Patient History
- 2.1 Search History
- 2.2 Maintain History
- 3.2 Make Diagnosis
- Diagnosis Paper
- Patient History Request
- Permanent History file
- Diagnosis for Hospital
- Diagnosis for Billing
Expanded Data Flow Diagram for Current Logical Model

With Physical files 3

4.1 Calculate Charge

4.2 Setup New Account

4.3 Enter Charge

4.4 Make Insurance form

4.5 Enter Payment

4.6 Bill Patient

4.7 Bill Inquiry

Diagnosis for Billing

Fee Book

Account Inquiry

Account Info

Charged Diagnosis

Charged Diagnosis

Accounting Book

Payment

Permanent History file

Insurance form
File Structure of Current Physical Model

1. Appointment-book

   { Name + Date + Time + Doctor }

2. Permanent-History-file

   { Name + Social-Security + Date_of_Birth + Sex + Address + Phone + Spouse/Parent + Insurer + Other_Insurer + Medicare + Visit_Date + Ins_Cert + Ins_Grp + (Diagnosis) + Clinical_Notes }

3. Accounting-book

   { Name + Social-Security + (Charge-Date + Service + Charge) + (Payment-Date + Payment-Type + Payment) + Balance }
Appendix C.

Deriving Logical File Equivalent from Current Physical Model
Expanded Data Flow Diagram
for Deriving Logical File Equivalent
for Deriving Logical File Equivalent

3.1 Request Patient History

2.1 Search History

2.2 Maintain History

5

3.2 Make Diagnosis

Diagnosis for Hospital

Diagnosis for Billing

Diagnosis File

Patient History File

Patient History Request

Diagnosis Paper
Expanded Data Flow Diagram 3
for Deriving Logical File Equivalent

4.1 Calculate Charge

4.2 Setup New Account

4.3 Enter Charge

4.4 Make Insurance Form

4.5 Enter Payment

4.6 Bill Patient

4.7 Bill Inquiry

Fee Book

Account Inquiry

Account Info

Bill

Diagnosis

Payment

Insurance Form

Permanent History File

Charged Diagnosis

Charged Diagnosis

Charged Diagnosis
1. First step: Compile a census of all references to stored data.

**Census of physical accesses**

<table>
<thead>
<tr>
<th>ACCESS</th>
<th>PROCESS</th>
<th>READ/WRITE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2</td>
<td>Read</td>
</tr>
<tr>
<td>2</td>
<td>1 3</td>
<td>Write</td>
</tr>
<tr>
<td>3</td>
<td>1 4</td>
<td>Write</td>
</tr>
<tr>
<td>4</td>
<td>1 5</td>
<td>Write</td>
</tr>
<tr>
<td>5</td>
<td>2.1</td>
<td>Read</td>
</tr>
<tr>
<td>6</td>
<td>2.2</td>
<td>Write</td>
</tr>
<tr>
<td>7</td>
<td>4.2</td>
<td>Write</td>
</tr>
<tr>
<td>8</td>
<td>4.3</td>
<td>Write</td>
</tr>
<tr>
<td>9</td>
<td>4.4</td>
<td>Read</td>
</tr>
<tr>
<td>10</td>
<td>4.5</td>
<td>Write</td>
</tr>
<tr>
<td>11</td>
<td>4.6</td>
<td>Read</td>
</tr>
<tr>
<td>12</td>
<td>4.7</td>
<td>Read</td>
</tr>
<tr>
<td>13</td>
<td>4.4</td>
<td>Read</td>
</tr>
</tbody>
</table>

2. Second Step: Replace each procedural data flow in the access census by the minimum set of data elements required to accomplish the purpose of the access. Associate required key information with each access.

**CENSUS OF LOGICAL READS (Incoming data flows)**

<table>
<thead>
<tr>
<th>ACCESS</th>
<th>KEY</th>
<th>LOGICAL DATA FLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Patient_Name</td>
<td>Date + Time + Doctor</td>
</tr>
<tr>
<td>5</td>
<td>Patient_Name + Patient_Number</td>
<td>Date_of_Birth + Visit_Date + (Diagnosis) + Clinical_Notes</td>
</tr>
<tr>
<td>9</td>
<td>Patient_Name + Patient_Number</td>
<td>S.s* + (Visit_Date + Service + Charge) + Total_Charge + Total_Payment + Balance</td>
</tr>
<tr>
<td>ACCESS</td>
<td>KEY</td>
<td>LOGICAL DATA FLOW</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>2</td>
<td>Patient_Name</td>
<td>Date + Time + Doctor</td>
</tr>
<tr>
<td>3</td>
<td>Patient_Name</td>
<td>Date + Time + Doctor</td>
</tr>
<tr>
<td>4</td>
<td>Patient_Name</td>
<td>Date + Time + Doctor</td>
</tr>
<tr>
<td>6</td>
<td>Patient_Name + Patient_Number</td>
<td>Date_of_Birth + Visit_Date + Diagnosis + Clinical_Notes</td>
</tr>
<tr>
<td>7</td>
<td>Patient_Name + Patient_Number</td>
<td>Medicare + date_of_Birth + Insured_Name + Address + Sex + S.s*</td>
</tr>
<tr>
<td>8</td>
<td>Patient_Name + Patient_Number</td>
<td>{Visit_Date + Charge + Service}</td>
</tr>
<tr>
<td>10</td>
<td>Patient_Name + Patient_Name</td>
<td>{Payment_Date + Payment_Type + Payment}</td>
</tr>
</tbody>
</table>
3. Third step: Decompose the file structure into private component files, each tailored to the requirement of a logical access.

PCF1 = (Patient_Name + Date + Time + Doctor)

PCF2 = (Patient_Name + Date + Time + Doctor)

PCF3 = (Patient_Name + Date + Time + Doctor)

PCF4 = (Patient_Name + Date + Time + Doctor)

PCF5 = (Patient_Name + Patient_Number + Date_of_Birth + Visit_Date + (Diagnosis) + Clinical_Notes)

PCF6 = (Patient_Name + Patient_Number + Date_of_Birth + Visit_Date + (Diagnosis) + Clinical_Notes + S.s* + Sex + Address + Phone + Spouse/Parent + Insurer + Other_Insurer + Medicare + Ins_Cert* + Ins_Grp*)

PCF7 = (Patient_Name + Patient_Number + Medicare + date_of_Birth + Insured_Name + address + Sex + Resp_Party + S.s*)

PCF8 = (Patient_Name + Patient_Number + (Visit_date + Charge + Service))

PCF9 = (Patient_Name + Patient_Number + (Visit_Date + Service + Charge) + Total_Charge + Total_Payment + Balance)

PCF10 = (Patient_Name + Patient_Number + (Payment_Date + Payment_Type + Payment))

PCF11 = (Patient_Name + Patient_Number + Address + Phone + (Visit_Date + Charge + Service) + Total_Charge + Payment_Made + Balance)

PCF12 = (Patient_Name + Patient_Number + Address + Phone + (Visit_Date + Charge) + (Payment_Date + Payment) + Total_Charge + Total_Payment + Balance)

PCF13 = (Patient_Name + Patient_Number + S.s* + Date_of_Birth + Sex + Address + Insurer + Other_Insurer + Ins_Cert* + Ins_Grp*)
4. Fourth step: Normalize the file structure by eliminating internal repeating groups.

NPCF1 = * Equivalent to PCF1 *
    = (Patient_Name + Date + Time + Doctor)

NPCF2 = * Equivalent to PCF2 *
    = (Patient_Name + Date + Time + Doctor)

NPCF3 = * Equivalent to PCF3 *
    = (Patient_Name + Date + Time + Doctor)

NPCF4 = * Equivalent to PCF4 *
    = (Patient_Name + Date + Time + Doctor)

NPCF5 = * Derived from PCF5 *
    = (Patient_Name + Patient_Number + Date_of_Birth + Visit_Date + Clinical_Notes)

NPCF6 = * Derived from PCF5 *
    = (Patient_Name + Patient_Number + Diagnosis)

NPCF7 = * Derived from PCF6 *
    = (Patient_Name + Patient_Number + Date_of_Birth + Visit_Date + Clinical_Notes + S.s* + Sex + Address + Phone + Spouse/Parent + Insurer + Other_Insurer + Medicare + Ins_Cert* + Ins_Grp*)

NPCF8 = * Derived from PCF6 *
    = (Patient_Name + Patient_Number + Diagnosis)

NPCF9 = * Equivalent to PCF7 *
    = (Patient_Name + Patient_Number + Medicare + Date_of_Birth + Insured_Name + Address + Sex + Resp_Party + S.s*)

NPCF10 = * Derived from PCF8 *
    = (Patient_Name + Patient_Number)

NPCF11 = * Derived from PCF8 *
    = (Patient_Name + Patient_Number + Visit_Date + Service + Charge)
NPCF12 = * Derived from PCF9 *
    = (Patient_Name + Patient_Number + S.s* + Total_Charge + Total_payment + Balance)

NPCF13 = * Derived from PCF9 *
    = (Patient_Name + Patient_Number + Visit_Date + service + Charge)

NPCF15 = * Derived from PCF10 *
    = (Patient_Name + Patient_Number)

NPCF16 = * Derived from PCF10 *
    = (Patient_Name + Patient_Number + Payment_Date + Payment_Date + payment_type + Payment)

NPCF17 = * Derived from PCF11 *
    = (Patient_Name + Patient_Number + Address + Phone + Total_Charge + Payment_Made + Balance)

NPCF18 = * Derived from PCF11 *
    = (Patient_Name + Patient_Number + Visit_Date + Service + Charge)

NPCF19 = * Derived from PCF12 *
    = (Patient_Name + Patient_Number + Address + Phone + Total_Charge + Total_Payment + Balance + Resp_Party + s.s*)

NPCF20 = * Derived from PCF12 *
    = (Patient_Name + Patient_Number + Visit_Date + Charge)

NPCF21 = * Derived from PCF12 *
    = (Patient_Name + Patient_Number + Payment_Date + Payment)

NPCF22 = * Equivalent to PCF13 *
    = (Patient_Name + Patient_Number + S.s* + Date_of_Birth + Sex + Address + Insurer + Other_Insurer + Ins_Cert* + Ins_Grp*)
5. Fifth step: Separate attributes from correlations. Associate attributes with the key attribute for the object they describe. Isolate correlations in pairs.

DNPCF1 = * Equivalent to NPCF1 *
= (Patient_Name + Date + Time + Doctor)

DNPCF2 = * Equivalent to NPCF2 *
= (Patient_Name + Date + Time + Doctor)

DNPCF3 = * Equivalent to NPCF3 *
= (Patient_Name + Date + Time + Doctor)

DNPCF4 = * Equivalent to NPCF4 *
= (Patient_Name + Date + Time + Doctor)

DNPCF5 = * Derived from NPCF5 *
= (Patient_Name + Patient_Number)

DNPCF6 = * Derived from NPCF5 *
= (Patient_Number + Date_of_Birth + Visit_Date + Clinical_Notes)

DNPCF7 = * Derived from NPCF6 *
= (Patient_Name + Patient_Number)

DNPCF8 = * Derived from NPCF6 *
= (Patient_Number + Diagnosis)

DNPCF9 = * Derived from NPCF7 *
= (Patient_Name + Patient_Number)

DNPCF10 = * Derived from NPCF7 *
= (Patient_Number + Date_of_Birth + Visit_Date + Clinical_Notes + S.s + Sex + Address + Phone + Spouse/Parent + Insurer + Other_Insurer + Medicare + ins_Cert + ins_Grp)

DNPCF11 = * Derived from NPCF8 *
= (Patient_Name + Patient_Number)
DNPCF12 = * Derived from NPCF8 *
          = (Patient_Number + Diagnosis)

DNPCF13 = * Derived from NPCF9 *
          = (Patient_Name + Patient_Number)

DNPCF14 = * Derived from NPCF9 *
          = (Patient_Number + Medicare + Date_of_Birth + Insured_Name +
             Address + Sex + Resp_Party + S.s*)

DNPCF15 = * Equivalent to NPCF10 *
          = (Patient_Name + Patient_Number)

DNPCF16 = * Derived from NPCF11 *
          = (Patient_Name + Patient_Number)

DNPCF17 = * Derived from NPCF11 *
          = (Patient_Number + Visit_Date)

DNPCF18 = * Derived from NPCF11 *
          = (Visit_Date + Service + Charge)

DNPCF19 = * Derived from NPCF12 *
          = (Patient_Name + Patient_Number)

DNPCF20 = * Derived from NPCF12 *
          = (Patient_Number + S.s* + Total_Charge + Total_Payment +
             Balance)

DNPCF21 = * Derived from NPCF13 *
          = (Patient_Name + Patient_Number)

DNPCF22 = * Derived from NPCF13 *
          = (Patient_Number + Visit_Date)
DNPCF23 = * Derived from NPCF13 *
   = (Visit_Date + Service + Charge)

DNPCF26 = * Equivalent to NPCF15 *
   = (Patient_Name + Patient_Number)

DNPCF27 = * Derived from NPCF16 *
   = (Patient_Name + Patient_Number)

DNPCF28 = * Derived from NPCF16 *
   = (Patient_Number + Payment_Date)

DNPCF29 = * Derived from NPCF16 *
   = (Payment_Date + Payment_Type + Payment)

DNPCF30 = * Derived from NPCF17 *
   = (Patient_Name + Patient_Number)

DNPCF31 = * Derived from NPCF17 *
   = (Patient_Number + Address + Phone + Total_Charge + Payment_Made + Balance)

DNPCF32 = * Derived from NPCF18 *
   = (Patient_Name + Patient_Number)

DNPCF33 = * Derived from NPCF18 *
   = (Patient_Number + Visit_Date)

DNPCF34 = * Derived from NPCF18 *
   = (Visit_Date + Service + Charge)

DNPCF35 = * Derived from NPCF19 *
   = (Patient_Name + Patient_Number)
DNPCF36 = * Derived from NPCF19 *
        = (Patient_Number + Address + Phone + Total_Charge +
           Total_Payment + Balance + Resp_Party + S.s#)

DNPCF37 = * Derived from NPCF20 *
        = (Patient_Name + Patient_Number)

DNPCF38 = * Derived from NPCF20 *
        = (Patient_Number + Visit_Date)

DNPCF39 = * Derived from NPCF20 *
        = (Visit_Date + Charge)

DNPCF40 = * Derived from NPCF21 *
        = (Patient_Name + Patient_Number)

DNPCF41 = * Derived from NPCF21 *
        = (Patient_Number + Payment_Date)

DNPCF42 = * Derived from NPCF21 *
        = (Payment_Date + payment)

DNPCF43 = * Derived from NPCF22 *
        = (Patient_Name + Patient_Number)

DNPCF44 = * Derived from NPCF22 *
        = (Patient_Number + S.s# + Date_of_Birth + Sex + Address +
           Insurer + Other_Insurer + Ins_Cert# + Ins_Grp#)
6 Sixth step: Combine attributes of the same object into common component files.

CCF1 = * Composed of DNPCF1, 2, 3, and 4 *
= (Patient_Name + Date + Time + Doctor)

CCF2 = * Equivalent to DNPCF5, 7, 9, 11, 13, 15, 16, 19, 21, 26, 27, 30, 32, 35, 37, 40, and 43 *
= (Patient_Name + Patient_Number)

CCF3 = * Composed of DNPCF6, 8, 10, 12, 14, 31, 36, and 44 *
= (Patient_Number + Date_of_Birth + Visit_Date + Clinical_Notes + Diagnosis + Medicare + Insured_Name + Address + Phone + Sex + S.s* + Total_Charge + Total_Payment + Balance + Resp_Party + Spouse/Parent + Insurer + Other_Insurer + Ins_Cert* + Ins_Grp*)

CCF4 = * Equivalent to DNPCF17, 22, 33, and 38 *
= (Patient_Number + Visit_Date)

CCF5 = * Composed of DNPCF18, 23, 34, and 39 *
= (Visit_Date + Service + Charge)

CCF6 = * Equivalent to DNPCF28, and 41 *
= (Patient_Number + Payment_Date)

CCF7 = * Composed of DNPCF29, and 42 *
= (Payment_Date + Payment_Type + Payment)
7. Seventh step: Remove any stored data element that can be derived by reference to other stored data elements.

DRNGLF1 = (Patient_Name + Date + Time + Doctor)

DRNGLF2 = (Patient_Name + Patient_Number)

DRNGLF3 = (Patient_Number + Date_of_Birth + Visit_Date + Clinical_Notes + Diagnosis + Address + Phone + Sex + S.s* + Doctor + Resp_Party + Insured_Name + Medicare + Spouse/Parent + Insurer + Other_Insurer + Ins_Cert* + Ins_Grp*)

DRNGLF4 = (Patient_Number + Visit_Date)

DRNGLF5 = (Visit_Date + Service + Charge)

DRNGLF6 = (Patient_Number + Payment_Date)

DRNGLF7 = (Payment_Date + Payment_Type + Payment)

a. Appointment_file = \{Patient_Name + Date + Time + Doctor\}

b. Patient_file = \{Patient_Name + Patient_Number + Address + Phone
+ Sex + Date_of_Birth + S.s# + Doctor + Resp_Party
+ Insured_Name + Medicare + Spouse/Parent +
Insurer + Other_Insurer + Ins_Cert# + Ins_Grp#\}

c. Diagnosis_file = \{Patient_Number + Visit_Date + Clinical_Notes +
Diagnosis\}

d. Account_file = \{Patient_Number + Date + Service + Charge\}

e. payment_file = \{Patient_Number + Payment_Date + Payment_Type +
Payment\}
Appendix D.

Current Logical Data Flow Diagram
Current Logical DFD: Level 0

1.0 Appointment

2.0 Maintain History

3.0 Diagnosis

4.0 Billing

Appointment file

Patient file

Diagnosis file

Patient Information

Diagnosis for Hospital

Diagnosis for Billing

Account file

Payment file

Fee book

Billing

Accounts into

Insurance form
Current Logical Data Flow Diagram
With Equivalent Logical Files

1.1 Classify Appointment
1.2 Find Appointment
1.3 Cancel Appointment
1.4 Modify Appointment
1.5 Schedule Appointment

Appointment
Appointment file
Appointment
Appointment Inquiry
Cancel Inquiry
Modify Inquiry
Schedule Inquiry
Appointment Inquiry
Appointment
Appointment file
Current Logical DFD 2
With Equivalent Logical files

3.1 Get_Patient_Info

3.2 Request_Patient_History

3.3 Search_History

3.4 Make_Diagnosis

Patient_Number assign

Patient_file

Diagnosis_file

Diagnosis_Paper

Patient_History_Request

Patient_History

Diagnosis_for_Billing

Diagnosis_for_Hospital
Current Logical Model 3
With Equivalent Logical Files

4.1 Calculate Charge
4.2 Find Patient Number
4.3 Enter Charge
4.4 Make Insurance form
4.5 Enter Payment
4.6 Bill Patient
4.7 Bill Inquiry

Diagnosis_for_Billing
Fee_Book
Account_Inquiry
Account_Info
Payment_file
Patient_file
Account_file
Patient_Number
Charged_Diagnosis
Charged_Diagnosis_form
Payment_file
Bill
Insurance_form
Payment
Appendix E.

New Logical Data Flow Diagram
New Logical DFD 3

3.1 Get Patient Info
3.2 Request Patient History
3.3 Search History
3.4 Make Diagnosis

Patient Information
Patient Number assign
Patient file
Diagnosis file
Diagnosis for Hospital
Diagnosis for Billing
New Logical DFD 4

4.1 Calculate Charge
4.2 Find Patient Number
4.3 Enter Charge
4.4 Make Insurance Form
4.5 Enter Payment
4.6 Bill Patient
4.7 Bill Inquiry
4.9 Report Monthly Account

- Account file
- Payment file
- Patient file
- Charged Diagnosis
- Account Inquiry
- Account Info
- Monthly Accounting
- Diagnosis for Billing
- Fee file
- Payment file
- Invoice
- Billing
- Payment
- Insurance Form
- Charge
- Patient Number
- Charged Diagnosis w/Number
- Client
- Aging Report
- Fee file
- Charged Diagnosis w/Number
New logical DFD 5

 Backup

 Appointment_file

 Payment_file

 Patient_file

 Account_file

 Diagnosis_file

 New Disk
Files for New Logical Model

1 Appointment_file

(Patient_Name + Date + Time + Doctor)

2 Patient_file

(Patient_Name + Patient_Number + Address + Phone + Sex +
  Date_of_Birth + S.s# + Doctor + Resp_Party + Insured_Name +
  Medicare + Spouse/Parent + Insurer + Other_Insurer + Ins_Cert# +
  Ins_Grp#)

3 Diagnosis_file

(Patient_Number + Visit_Date + Diagnosis_Code + Diagnosis +
  Clinical_Note)

4 Account_file

(Patient_Number + Charge_Date + Service + Charge)

5 Payment_file

(Patient_Number + Payment_Date + Payment_Type + Payment)

6 Fee_file

(Diagnosis_Code + Diagnosis + Fee)
Appendix F.

Data Flow Diagram for future system
4.1 Update_History

4.2 Produce_History_Update_Sheet

4.3 Report_Service_Type

Diagram 4

History_Update

Diagnosis_file

Patient_file

Service_Report

History_Update_Sheet
Appendix G.

Data Dictionary
FILE OR DATA BASE NAME: Account_file

ALIASES: none

COMPOSITION:

```plaintext
{ Patient_Number +
  Date +
  Service +
  Charge
}
```

ORGANIZATION: indexed by Patient_Number and Date

NOTES:

FILE OR DATABASE NAME: Appointment_file

ALIASES: none

COMPOSITION:

```plaintext
{ Patient_Name +
  Date +
  Time +
  Min +
  Apm +
  Doctor +
  Reason
}
```

ORGANIZATION: indexed by Patient_Name

NOTES:
FILE OR DATABASE NAME: Payment_file
COMPOSITION:

{ Patient_Number +
  Date +
  Payment_Type +
  Payment }

FILE OR DATABASE NAME: Diagnosis_file
COMPOSITION:

{ Patient_Number +
  Visit_Date +
  Diagnosis_Code +
  Diagnosis +
  Clinical_Notes }
FILE OR DATABASE NAME: Fee_file

ALIASES: none

COMPOSITION:

( Code +
  Service +
  Fee )

ORGANIZATION: Indexed by Code

NOTES:
FILE OR DATABASE NAME: Patient_file

COMPOSITION:

{ Patient_Name +
  Patient_Number +
  Address +
  Phone +
  Date_of_Birth +
  Sex +
  Doctor +
  S.s* +
  Resp_Party +
  Insured_Name +
  Medicare +
  Spouse/Parent +
  Insurer +
  Other_Insurer +
  Ins_Cert* +
  Ins_Grp* }

ORGANIZATION: indexed by Patient_Name and Patient_Number

NOTES:
DATA FLOW NAME: Aging_Report

ALIASES: none

COMPOSITION:

{ Patient_Name +
  Patient_Number +
  { Aged_Charge}
} +
Total_Aged_Charge +
Sum_Charge
DATAFLOW NAME: Account_Info

ALIASES: none

COMPOSITION:

Patient_Name +
Patient_Address +
Patient_Phone +
Patient_Number +
[ Date +
  Service +
  Charge ] +
[ Payment_Date +
  Payment_Type +
  Payment ] +
Total_Charge +
Total_Payment +
Balance

NOTES:

DATA FLOW NAME: Diagnosis_Paper

ALIASES: none

COMPOSITION:

Patient_Name +
Patient_Number +
Date_of_Birth +
Sex +
Doctor

NOTES:
DATAFLOW_NAME: Appointment_Inquiry

COMPOSITION:

Patient_Name +
Inquiry_Type +
Appointment_Date +
Doctor +
Reason

NOTES:

DATAFLOW_NAME: Bills

COMPOSITION:

[ Patient_Name +
  Patient_Number +
  Address +
  Phone +
  { Visit_Date +
    Charge +
    Service
  } +
  Total_Charge +
  { Payment_Date +
    Payment_Type +
    Payment
  } +
  Total_Payment +
]
NOTES:
*****************************************************************************
DATAFLOW NAME: Appointment
*****************************************************************************
ALIASES: none
*****************************************************************************
COMPOSITION:

Patient_Name +
Date +
Time +
Doctor +
Reason

*****************************************************************************
NOTES:
*****************************************************************************
*****************************************************************************
DATA FLOW NAME: Appointment_Sheet
*****************************************************************************
ALIASES: none
*****************************************************************************
COMPOSITION:

{ Patient_Name +
  Date +
  Time +
  Doctor +
  Reason }

*****************************************************************************
NOTES:
*****************************************************************************
*****************************************************************************
DATAFLOW NAME: History_Update

ALIASES: none

COMPOSITION:

{ Patient_Number +
  Visit_Date +
  Diagnosis_Code +
  Diagnosis +
  Clinical_Notes
}

NOTES:

DATAFLOW NAME: History_Update_Sheet

ALIASES: none

COMPOSITION:

{ Patient_Name +
  Visit_Date +
  Diagnosis_Code +
  Diagnosis +
  Clinical_Notes
}

NOTES:
DATA FLOW NAME: Monthly_Accounting

ALIASES: none

COMPOSITION:

[ Month +
  Month_Total_Charge +
  Month_Total_Payment +
  Month_Total_Balance
]

NOTES:
DATAFLOW NAME: Patient_Charge

ALIASES: none

COMPOSITION:

{ Patient_Name +
  { Date +
    Service +
    Charge
  }
}

NOTES:

DATAFLOW NAME: Insurance_Form

ALIASES:

COMPOSITION:

Patient_Name +
Address +
Date_of_Birth +
Sex +
S.s* +
Insured_Name +
Medicare +
Insurer +
Ins_Cert* +
Ins_Grp* +
{ Date +
  Service +
  Charge +
  Diagnosis } +
DATAFLOW NAME: Patient_History_Request

ALIASES: none

COMPOSITION:

Patient_Name

DATAFLOW NAME: Patient_History

ALIASES: none

COMPOSITION:

Patient_Name +
Patient_Number +
(Visit_Date +
Diagnosis_Code +
Diagnosis +
Clinical_Notes)

NOTES:
DATA FLOW NAME: Patient_Information

ALIASES: none

COMPOSITION:

(Patient_Name +
 Address +
 Phone +
 Date_of_Birth +
 Sex +
 Doctor +
 S.s* +
 Resp_Party +
 Insured_Name +
 Medicare +
 Spouse/Parent +
 Insurer +
 Other_Insurer +
 Ins_Cert* +
 Ins_Grp* )

NOTES:
DATAFLOW NAME: Payment

ATLAS: none

COMPOSITION:

- Date
- Payment_Type
- Payment

NOTES:

**********************************************************

DATAFLOW NAME: Account_Inquiry

ATLAS: none

COMPOSITION:

- Patient_Name

NOTES:

**********************************************************
DATA FLOW NAME: Service_Report

COMPOSITION:

{ Visit_Date +
   Diagnosis +
   Clinical_Notes }

NOTES:
DATA ELEMENT NAME: Address
ALIASES: Patient_Address
VALUES AND MEANINGS:
  40 character string
NOTES: patient's address

DATA ELEMENT NAME: Age
VALUES AND MEANINGS:
  three digit integer
NOTES: Age of patient

DATA ELEMENT NAME: Apm
VALUES AND MEANINGS:
  AM or PM. Two character string.
NOTES: none
DATA ELEMENT NAME: Aged_Charge
ALIASES: none
VALUES AND MEANINGS:

Dollar Amount.
Aged Charge of a patient. Current, 30-60 days, 60-90 days, over 90 days.
DATA ELEMENT NAME: Appointment_Date
ALIASES: Date, Visit_Date, Last_Visit
VALUES AND MEANINGS:
   Date in the form of MM/DD/YY
NOTES:

DATA ELEMENT NAME: Balance
ALIASES: none
VALUES AND MEANINGS:
   Dollar amount
NOTES:
   Balance due (difference between Total_Charge and Total_Payment)

DATA ELEMENT NAME: Charge
ALIASES: none
VALUES AND MEANINGS:
   Dollar amount
NOTES: service charge
DATA ELEMENT NAME: Clinical_Notes

VALUES AND MEANINGS:

Specific description of diagnosis made by the doctor.
50 character string.

NOTES:

DATA ELEMENT NAME: Date

VALUES AND MEANINGS:

Date in the form of MM/DD/YY

NOTES:

DATA ELEMENT NAME: Date_of_Birth

VALUES AND MEANINGS:

Date in the form of MM/DD/YY

NOTES: patient's date of birth
DATA ELEMENT NAME: Code

VALUES AND MEANINGS;

Service code for common service such as immunization, physical exam, and laboratory fees.
5 character string.

NOTES:
DATA ELEMENT NAME: Diagnosis
VALUES AND MEANINGS:
Diagnosis made by the doctor.
40 character string.
NOTES:

DATA ELEMENT NAME: Diagnosis_Code
VALUES AND MEANINGS:
Diagnosis code. 10 character string.
NOTES: none
DATA ELEMENT NAME: Fee

VALUES AND MEANINGS:

Dollar amount

NOTES:
DATA ELEMENT NAME: Ins_Cert*
ALIASES: none
VALUES AND MEANINGS:

Insurance certificate number

NOTES:

DATA ELEMENT NAME: Ins_Grp*
ALIASES: none
VALUES AND MEANINGS:

Insurance group number.

NOTES:

DATA ELEMENT NAME: Insured_Name
ALIASES: none
VALUES AND MEANINGS:

Insured name: 20 character string.
DATA ELEMENT NAME: Insurer
 ALIASES: none
 VALUES AND MEANINGS:

 Name and address of the insurance company which will pay the medical fee for the patient.

 NOTES:

 ELEMENT NAME: Medicare
 ALIASES: none
 VALUES AND MEANINGS:

 Medicare number: 10 digit integer.

 NOTES:

 DATA ELEMENT NAME: Min
 ALIASES: none
 VALUES AND MEANINGS:

 Minute in the form of mm, 2 digit integer.

 NOTES:
DATA ELEMENT NAME: Month

VALUES AND MEANINGS:

Month of Monthly_Accounting. In the form of 99.

NOTES:

-------------------------

DATA ELEMENT NAME: Month_Total_Charge

VALUES AND MEANINGS:

Dollar amount.
Sum of Total_Charge of the month.

NOTES:
DATA ELEMENT NAME: Month_Total_Payment
VALUES AND MEANINGS:
   Dollar amount.
   Sum of Total_Payment of the month.

DATA ELEMENT NAME: Month_Total_Balance
VALUES AND MEANINGS:
   Dollar Amount.
   Difference between Month_Total_Charge and Month_Total_Payment.
DATA ELEMENT NAME: Other_insurer
ALIASES: none
VALUES AND MEANINGS:

Other insurance company. 30 character string.

NOTES:

DATA ELEMENT NAME: Patient_Address
ALIASES: Address
VALUES AND MEANINGS:

40 character string

NOTES:

DATA ELEMENT NAME: Patient_Name
ALIASES: none
VALUES AND MEANINGS:

20 character string.

NOTES: patient's first and last name
check, cash, or credit card.

VALUES AND MEANINGS:

ALIASES: none

DATA ELEMENT NAME: Payment Type

NOTES:

15 character string

VALUES AND MEANINGS:

ALIASES: Phone

DATA ELEMENT NAME: Patient Phone

NOTES:

Four digit character

VALUES AND MEANINGS:

ALIASES: none

DATA ELEMENT NAME: Patient Number
DATA ELEMENT NAME: Payment
VALUES AND MEANINGS:
   Dollar amount
NOTES: amount of a patient's payment

DATA ELEMENT NAME: Phone
VALUES AND MEANINGS:
   Phone number is in the form of (***)***-***-***.
   13 character string.
NOTES: patient's phone number

DATA ELEMENT NAME: Reason
VALUES AND MEANINGS:
   The reason for visit.
   40 character string.
NOTES:
DATA ELEMENT NAME: Resp_Party

VALUES AND MEANINGS:

The name of the person who is responsible for the payment of the medical fee of the patient. 20 character string.

NOTES:

DATA ELEMENT NAME: Service

VALUES AND MEANINGS:

The type of service performed by the doctor.

NOTES:

DATA ELEMENT NAME: Sex

VALUES AND MEANINGS:

Sex of the patient.

1 character string, "F" or "M".
DATA ELEMENT NAME: Spouse/Parent
ALIASES: none
VALUES AND MEANINGS:
Name of parents of the patient if child or full name of spouse if married.
20 character string.

NOTES:

DATA ELEMENT NAME: S.s*
ALIASES: none
VALUES AND MEANINGS:
Social security number 9 character string.

NOTES:

DATA ELEMENT NAME: Time
ALIASES: none
VALUES AND MEANINGS:
Time in the form of HH, 2 digit integer.

NOTES:
DATA ELEMENT NAME: Total_Aged_Charge
ALIASES: none
VALUES AND MEANINGS:

Dollar Amount.
Sum of each Aged_Charge of all patients.
Current, 30-60 days, 60-90 days, over 90 days.

NOTES:

DATA ELEMENT NAME: Sum_Charge
ALIASES: none
VALUES AND MEANINGS:

Dollar amount.
Sum of each Total_Aged_Charge.
DATA ELEMENT NAME: Total_Charge
**********************************************************
ALIASES: none
**********************************************************
VALUES AND MEANINGS:

Dollar amount. Total service charge of the patient so far.

**********************************************************
NOTES:
**********************************************************
**********************************************************

DATA ELEMENT NAME: Total_Payment
**********************************************************
ALIASES: none
**********************************************************
VALUES AND MEANINGS:

Dollar amount. Total payment of the patient so far

**********************************************************
NOTES:
**********************************************************
**********************************************************

DATA ELEMENT NAME: Visit_Date
**********************************************************
ALIASES: Date
**********************************************************
VALUES AND MEANINGS:

Date in the form of MM/DD/YY

**********************************************************
NOTES:
**********************************************************
Appendix H.

Process Description
Appointment ---- Process 1

Classify Appointment_Inquiry
Process 1.1

Select Appointment_Inquiry from screen:

  Case 1: (Find_Inquiry)
      Call Find_Appointment.

  Case 2: (Cancel_Inquiry)
      Call Cancel_Appointment.

  Case 3: (Modify_Inquiry)
      Call Modify_Appointment.

  Case 4: (Schedule_Inquiry)
      Call Schedule_Appointment.

  Case 5: (Print_Inquiry)
      Call Print_Appointment.

Find_Appointment
Process 1.2

Accept Patient_Name from screen.
Access Appointment_file by the Patient_Name.
Display Appointment_file_Record on the screen.
Cancel_Appointment
Process 1.3

Accept Patient_Name from screen.
Access Appointment_file by the Patient_Name.
Display the Appointment_file_Record on the screen.
Accept cancel_Confirm.
   IF cancel_Confirm is true
       THEN delete the Appointment_file_Record.

Modify_Appointment
Process 1.4

Accept Patient_Name from screen.
Access Appointment_file by the Patient_Name.
Display the Appointment_file_Record on the screen.
Select Modify_Option:

Case 1: (Patient_Name)
        Accept Patient_Name.
        Replace Patient_Name in Appointment_file with new one.

Case 2: (Date)
        Accept Date.
        Replace Date in Appointment_file with new one.

Case 3: (Time)
        Accept Time.
        Replace Time in Appointment_file with new one.
Case 4: (Doctor)
Accept Doctor.
Replace Doctor in Appointment_file with new one.

Case 5: (Reason)
Accept Reason.
Replace Reason in Appointment_file with new one.

Schedule_Appointment
Process 1.5

Accept Date from screen.
Access Appointment_file using date as key.
For each Appointment_file_Record of the date:
   Display the Appointment_file_Record on the screen.

Accept Patient_Name, Time, Doctor, and Reason.
Accept Schedule_Confirm.
   IF Schedule_Confirm is true
   THEN Add a new record to Appointment_file using the Patient_Name,
       Date, Time, and Doctor.
Print_Appointment_Schedule
Process 1.6

Accept Date from screen.
Access Appointment_file using Date as key.

For each Appointment_file_Record:
    Print the Appointment_file_Record onto Appointment_Sheet.
Find_Patient_Number
Process 2.1

Accept Patient_Name from screen.
Access Patient_file using the Patient_Name as key

IF found
Then Display Patient_Number, Address on screen.

Enter_Charge
Process 2.2

Accept patient_Numbers from screen.

For each Patient_Number:
Accept dates.

For each Dates:
Accept Service.
Accept Charge.

Add a new record to Account_file using the Patient_Number, Date,
service and Charge.
Print_Insurance_Form
Process 2.3

Accept Patient_Numbers from screen.
For each Patient_Number:

Access Patient_file using the Patient_Number as key
Print Patient_Name, Patient_Number, Address, Sex, Date_of_Birth,
       S.s#, Insured_Name and Medicare onto Insurance_form.

Access Account_file using the Patient_Number as key
For each Date of the Patient_Number:

Print Date, service, and Charge onto Insurance_form.

Enter_Payment
Process 2.4

Accept Patient_Numbers from screen.
For each Patient_Number:

Accept dates.
For each Date:

Accept payment_Type.
Accept Payment.

Add a new record to Payment_file using the Patient_Number, Date,
       payment_Type, and Payment.
Accept Bill Date, Month, and Year from screen.

Access Account_file.
Sort the Account_file using Patient_Number as key.

For each Patient_Number of the sorted Account_file:

Access Patient_file using the Patient_Number as key.
Print Doctor, Patient_Name, Address, and Phone.
For each Date of the Patient_Number:

Calculate Date_Count using Date and accepted Bill Date.
Add Charge of the record to Curr_Age, Age_30, Age_60, or Age_90 depending on the Date_count.
SET Balance TO Balance plus Charge.
IF Month and Year of Date is equal to the accepted Bill Month and Year from screen
THEN
Print Date, Service, and Charge.

Access Payment_file using the Patient_Number as key.
For each Date of the Patient_Number:

Subtract the Payment from in the order of Age_90, Age_60, Age_30, or Age_90 while the values are greater than 0.
SET Balance TO Balance minus Payment.
IF Month and Year of Date is equal to the accepted Bill Month and Year from screen
THEN
Print Date, Payment_Type, and Payment.

Print Balance.
Print Curr_Age, Age_30, Age_60, and Age_90.
Accept patient_Name from screen.
Access Patient_file using the Patient_Name as key.
For each Patient_Number of the Patient_Name:
  Display patient_Number, Address, Phone, date_of_Birth on screen.
  Accept Confirm_Ok.
  IF Confirm_Ok is true
    Then
      Access Account_file using the Patient_Number as key.
      For each Date of the Patient_Number:
        Display Date, service, and Charge on screen.
        Set Total_Charge to Total_Charge plus Charge.

      Access Payment_file using the Patient_Number as key.
      For each Date of the Patient_Number:
        Display date, payment_Type, and Payment on screen.
        Set Total_Payment to Total_Payment plus Payment.

      Set Balance to Total_Charge minus Total_Payment.
      Display Total_Charge, Total_Payment, and Balance on screen.
Aging_Charge
Process 2.7

Accept Current_Date ( C_Year, C_Mon, C_Date ) from screen.

Sort Account_file using Patient_Name as sort key.
Access Sorted_Account_file.
For each record of Sorted_Account_file:

IF Patient_Name is not equal to Name
THEN
Access Payment_file using Name as key.
For each Payment_Date of the Patient_Name:

SET Pay TO Payment of the record.

IF Pay is greater than 90_Age
THEN
SET Pay TO Pay minus 90_Age.
SET 90_Age TO 0.

IF Pay is greater than 60_Age
THEN
SET Pay TO Pay minus 60_Age.
SET 60_Age TO 0.

IF Pay is greater than 30_Age
THEN
SET Pay TO Pay minus 30_Age.
SET 30_Age TO 0.

ELSE
SET 30_Age TO 30_Age minus Pay.

ELSE
SET 30_Age TO 30_Age minus Pay.
ELSE ** ELSE 3 **
    SET 60_Age TO 60_Age minus Pay.

ELSE ** ELSE 2 **
    SET 90_Age TO 90_Age minus Pay.

Print Name, Curr_Age, 30_Age, 60_Age, and 90_Age onto Aging_Report.
SET Total_Curr_Age TO Total_Curr_Age plus Curr_Age.
SET Total_30_Age TO Total_30_Age plus 30_Age.
SET Total_60_Age TO Total_60_Age plus 60_Age.
SET Total_90_Age TO Total_90_Age plus 90_Age.
SET Name TO Patient_Name.
SET Curr_Age, 30_Age, 60_Age, and 90_Age TO 0. ** END IF 1 **

IF C_Year greater than or equal to Year of the Date of the record of 
    Sorted_Account_file ** Cont. of For 1 **
THEN
    SET Y_Day TO ((C_Year - Year of the record) * 365).

IF C_Mon greater than or equal to Month of the Date of the record of 
    Sorted_Account_file
THEN
    SET M_Day TO ((C_Mon - Month of the record) * 30 ).
ELSE
    SET M_Day TO (((C_Mon - Month of the record) * 30) * 365).
    SET Y_Day TO (Y_Day - 365).

IF C_Date greater than or equal to Date of the record of 
    Sorted_Account_file
THEN
    SET D_Day TO (C_Date - Date of the record).
ELSE
    SET D_Day TO ((C_Date - Date of the record) * 30).
    SET M_Day TO (M_Day - 30).
SET Date_Count TO (Y_Day + M_Day + D_Day).

CASE 1: (Date_Count is less than 31)
SET Curr_Age TO Curr_Age plus Charge.

CASE 2: (Date_Count is greater than 30 and less than 61)
SET 30_Age TO 30_Age plus Charge.

CASE 3: (Date_Count is greater than 60 and less than 91)
SET 60_Age TO 60_Age plus Charge.

CASE 4: (Date_Count is greater than 90)
SET 90_Age TO 90_Age plus Charge.

SET Sum_Charge TO Sum_Charge plus (Total_Curr_Age + Total_30_Age +
Total_60_Age + Total_90_Age).

Print Total_Curr_Age, Total_30_Age, Total_60_Age, and Total_90_Age onto Aging_Report.

Print Sum_Charge.
Accept Month from screen.

Sort Account_file using Patient_Number as key.

Access the sorted Account_file.
For each Patient_Number of the sorted Account_file:

    For each Date of the Patient_Number:
        IF Month of Date of the record is equal to Month
            THEN
                SET Month_Total_Charge TO Month_Total_Charge plus Charge.
                SET Total_Charge TO Total_Charge plus Charge.
                SET Month_Total_Balance TO Month_Total_Balance plus Charge.
                SET Total_Balance TO Total_Balance plus Charge.

Access Payment_file using the Patient_Number as key.
For each Date of the Patient_Number:
    IF Month of Payment_Date of the record is equal to Month
    THEN
        SET Month_Total_Payment TO Month_Total_Payment plus Payment.
        SET Total_Payment TO Total_Payment plus Payment.
        SET Month_Total_Balance TO Month_Total_Balance minus Payment.
        SET Total_Balance TO Total_Balance minus Payment.

Print Total_Charge, Total_Payment, and Total_Balance.
SET Total_Charge, Total_Payment, Total_Balance TO 0.

Print Month, Month_Total_Charge, Month_Total_Payment, and
    Month_Total_Balance onto Monthly_Accounting.
Get_Patient_Info
Process 3.1

Accept Patient_Name, Patient_Number, Address, Phone, Sex,
   Date_of_Birth, Doctor, S.s*, Insured_Name, Resp_party,
   Spouse/Parent, Insurer, Other_Insurer, Ins_Cert*, Ins_Grp*, and
   Medicare from screen.
Accept Information_Ok.
   IF Information_Ok is true
     Then Add a new record to Patient_file using the information accepted.
     Print the new record on the Diagnosis_Paper.

Search_History
Process 3.2

Accept Patient_Name from screen.
Accept Patient_file using Patient_Name as key
For each Patient_Number of the Patient_Name:
   Display Patient_Number, Address, Date_of_Birth, and Doctor
Accept Search_Ok.

IF Search_Ok is true
Then Access Diagnosis_file using the Patient_Number as key
For each Visit_Date of the Patient_Number:
   Print Patient_Name, Patient_Number, Visit_Date, Date_of_Birth,
   Doctor, Diagnosis_Code, Diagnosis, and Clinical_Notes onto
   History_Sheet.
Maintain History ------ Process 4

Update History
Process 4.1

Accept Patient_Number, Visit_Date, Diagnosis_Code, Diagnosis, Clinical_Notes from screen.

Access Diagnosis_file using the Patient_Number as key
IF found
    THEN
        Replace the record with the informations accepted
ELSE
    Add a new record to Diagnosis_file using the informations accepted.
Produce_Update_History_Sheet
Process 4.2

Access Patient_file using Patient_Name as key.
For each Patient_Number of the Patient_Name:
  Display Patient_Number, Address, S.s#, Date_of_Birth, and Doctor.
  Accept History_Ok.

  If History_OK is true
  Then
    Access Diagnosis_file using the Patient_Number as key.
    For each Visit_Date of the Patient_Number:

      Display Patient_Number, Visit_Date, Diagnosis, and Clinical_Notes.
      Accept Print_Ok.
      If Print_Ok is true
      Then
        Print Patient_Name, Patient_Number, Date_of_Birth, Doctor, Visit_Date, Diagnosis, and Clinical_Notes onto History_Update_Sheet.

Report_Service_Type
Process 4.3

Accept Date from screen.

Access Diagnosis_file using the Date as key.
For each Diagnosis of the Date:
  Add 1 to the Number of the Diagnosis.

Print Numbers of Diagnosis of the Dates.
Backup ----- Process 5

Backup
Process 5.0

Copy Appointment_file to Backup_Disk.
Copy Patient_file to Backup_Disk.
Copy Diagnosis_file to Backup_Disk.
Copy Account_file to Backup_Disk.
Copy Payment_file to Backup_Disk.
Appendix I.

Screen for New System
WELCOME TO CLINIC MANAGEMENT SYSTEM

1 => APPOINTMENT
2 => BILLING
3 => PATIENT HISTORY
4 => MAINTAINING HISTORY
5 => END

Enter Option Number: _

SCREEN 1

APPOINTMENT SCHEDULING

F => Find appointment
C => Cancel appointment
M => Modify appointment
S => Schedule appointment
P => Print appointment schedule
E => Return to main screen

Enter selection number: _

SCREEN 2
PATIENT NAME:
DATE OF APPOINTMENT:
TIME OF APPOINTMENT:
DOCTOR:
REASON:

Hit "Return" to continue

SCREEN 3

Enter Patient Name:_____________________

* Type first name and last name.

SCREEN 4
PATIENT NAME:
DATE OF APPOINTMENT:
TIME OF APPOINTMENT:
DOCTOR:
REASON:

Are you sure to cancel?(Y/N) _

SCREEN 5

MODIFY APPOINTMENT
1 => Patient Name:
2 => Date of Appointment:
3 => Time of Appointment:
4 => Doctor:
5 => Reason:

Enter Selection Number: _
( Type "E" to return appointment menu )

SCREEN 6
BILLING
1 => Set Up New Account
2 => Input Patient Charge
3 => Input Patient Payment
4 => Bill Patient
5 => Billing Inquiry from Patient
6 => Print Insurance Form
7 => Aging Charges of all Patients
8 => Report Monthly Account
E => Return to Main menu

Enter Selection Number: _

SCREEN 7

PATIENT INFORMATION
1.PATIENT NAME:
2.PATIENT NUMBER:
3.ADDRESS:
4.PHONE NUMBER:
5.DATE OF BIRTH:
6.SEX:
7.DOCTOR:
8.S.s*:
9.RESP_PARTY:
10.INSURED NAME:
11.MEDICARE

-----IS THIS CORRECT INFORMATION?(Y/N): _

SCREEN 8
INPUT CURRENT PATIENT CHARGE

1. PATIENT NUMBER: ----
2. DATE: --/--/--
3. SERVICE: --------------
4. CHARGE: """"""

-----------

SCREEN 9

INPUT PAYMENT

-------------------

1. PATIENT NUMBER: ----
2. DATE: --/--/--
3. PAYMENT TYPE: -
   (H: check, A: cash, R: credit card)
4. PAYMENT: ----

-------------------

SCREEN 10
<table>
<thead>
<tr>
<th>NAME:</th>
<th>NUMBER:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHARGE</th>
<th>PAYMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>SERVICE CHARGE</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
</tr>
</tbody>
</table>

TOTAL CHARGE: ------  TOTAL PAY: ------  BALANCE: ------

SCREEN 11

<table>
<thead>
<tr>
<th>NUMBER:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME:</td>
</tr>
<tr>
<td>DATE OF BIRTH:</td>
</tr>
<tr>
<td>SEX:</td>
</tr>
<tr>
<td>DOCTOR:</td>
</tr>
<tr>
<td>VISIT DATE:</td>
</tr>
<tr>
<td>DIAGNOSIS CODE:</td>
</tr>
<tr>
<td>DIAGNOSIS:</td>
</tr>
<tr>
<td>CLINICAL NOTES:</td>
</tr>
</tbody>
</table>

SCREEN 12
MAINTAINING PATIENT HISTORY

U => Update Patient History
P => Print Updated History
S => Report Service Type
R => END

Select Option: -

SCREEN 13

1. NAME:
2. NUMBER:
3. D.O.B:
4. DOCTOR:
5. VISIT DATE:
6. DIAGNOSIS CODE:
7. DIAGNOSIS:
8. CLINICAL NOTES:

----------------------------------------
Hit "Return" to continue.

SCREEN 14
PATIENT HISTORY

I -> Input Patient Information
D -> Search Patient History
R -> Return to Main menu

Enter Selection Number: -
Appendix J.

Structure Chart - High Level Design
0.0 SYSTEM

1.0 Display_Menu

2.0 Appointment

3.0 Billing

4.0 Diagnosis

5.0 Maintain_History

1 Option
2.0 Appointment

- 2.1 Display Appointment Menu
- 2.2 Find Appointment
- 2.3 Schedule Appointment
- 2.4 Modify Appointment
- 2.5 Cancel Appointment
- 2.6 Print Appointment

1: Appointment Option
2.1
Display_Appointment_Menu

2.1.1
Show_Appointment_Menu

2.1.2
Get_Appointment_Option

1: Appointment_Option
2.2
Find_Appointment

2.2.1
Get_Patient_Name

2.2.2
Find_Appointment_Record

2.2.3
Display_Appointment_Record

1: Patient_Name
2: Appointment
2.3
Schedule
Appointment

2.3.1
Display_Appointment

2.3.2
Schedule
2.3.2
Schedule

2.3.2.1 Get_Appointment_Info

2.3.2.2 Verify_Info

2.3.2.3 Make_Appointment

1 Appointment_Info
2 Appointment_Ok
2.4 Modify_Appointment

2.2.1 Get_Patient_Name

2.4.2 Display_Modify_Menu

2.4.3 Make_Modification

2.4.4 UserModification_Confirm

1: Patient_Name
2: Modify_Option
3: Modify_OK
2.5
Cancel_Appointment

2.2.1
Get_Patient_Name

1: Patient_Name

2.5.2
Remove_Appointment

1: Patient_Name
2.5.2 Remove_Appointment

2.5.2.1 Display_Cancel_Record

2.5.2.2 User_Cancel_Confirm

2.5.2.3 Remove_Record

1 Patient_Name
2 Cancel_OK
3.0
Billing

3.1 Display Account Menu
3.2 Setup New Account
3.3 Input Charge

3.4 Input Payment
3.5 Bill Patient
3.6 Bill Inquiry

3.7 Aging Charge
3.8 Report monthly Account
3.9 Print Insurance Form

1: Account Option
3.1
Display_Account_Menu

3.1.1
Show_Account_Menu

3.1.2
Get_Account_Option

1: Account_Option
3.3 Input_Charge

3.3.1 Charge_Patient_Number

3.3.2 Read_Charge_Info

3.3.3 Add_Charge

1. Patient_Number
2. Charge_Info
3.4.1 Read Patient Number

3.4.2 Read Payment Info

3.4.3 Add Payment

1: Patient_Number
2: Payment_info
3.5  Bill.Patient

3.5.1  Get.Bill.Date
3.5.2  Read.Account.File
3.5.3  Read.Payment.File
3.5.4  Read.Payment.File
3.5.5  Print.Bill

1.Bill.Date
2.Patient.Number
3.Charge_Info
4.Payment_Info
5.Patient_Info
2.2.1 Get_Patient_Name
3.6 Bill_Inquiry
3.6.2 Find_Bill_Info
1: Patient_Name
3.6.2 Find_Bill_Info

3.6.2.1 Display_Patient_Info

3.6.2.2 Display_Charge

3.6.2.3 Display_Payment

3.6.2.4 Write_Balance

1: Patient_Name
2: Patient_Number
3: Total_Charge
4: Total_Payment
4.0 Diagnosis

4.1 Dis_Diag_Menu

4.2 Input_Patient_information

4.3 Do_Diagnosis

1: Diag_Option
4.2
Input_Patient_Information

4.2.1
Get_New_Information

4.2.2
Build_New_Information
4.3 Do_Diagnosis

2.2.2 Get_Patient_Number

4.3.1 Request_Patient_History

4.3.2 Display_Patient_History

1: Patient_Number
2: Patient_History
5.2 Update_History

5.2.1 Read_History_Update

5.2.2 Add_Diagnosis_File

1: History_Update
5.3 Print_Update_History

2.2.1 Get_Patient_Name

5.3.2 Print_Patient_File

5.3.3 Print_History

1: Patient_Name
2: Patient_Number
Appendix K.

Module Description - Low Level Design
procedure Add_Charge( in: Charge_Info );

begin

Move Charge_Info to a new record.
Add the new record to Account_file using Patient_Number as key.

end;

MODULE NAME: Add_Diagnosis_File
MODULE NUMBER: 5.2.2

procedure Add_Diagnosis_file( in: History_Update );

begin

Add a record in Diagnosis_file using History_Update.

end;
MODULE NAME: Add_Payment

procedure Add_Payment( in: Payment_Info);

begin

    Move Payment_Info a new record.
    Add the new record to Payment_file using Patient_Number as key.

end;

MODULE NAME: Appointment

procedure Appointment

begin

    Display_Appointment_Menu( Appointment_Option );

    case Appointment_Option of
        'F': Find_Appointment;
        'S': Schedule_Appointment;
        'M': Modify_Appointment;
        'C': Cancel_Appointment;
        'P': Print_Appointment;
        'E': escape;
    end case;

end
module Aging_Charge

procedure Aging_Charge;

begin

accept current date(C_Year, C_Mon, C_Date) from screen;
sort Account_file using P_Number as sort key;
access the Sorted_Account_file;

Number := P_Number;
DO WHILE not end of file

if P_Number<>Number then
    access Payment_file using Number as key;

    DO WHILE not end of file

    if P_Number= Number then
        Pay := Payment;
        if Pay>Age_90 then
            Pay := Pay - Age_90;
            Age_90 := 0;
        if Pay>Age_60 then
            Pay := Pay - Age_60;
            Age_60 := 0;
        if Pay>Age_30 then
            Pay := Pay - Age_30;
            Age_30 := 0;
        else
            Age_30 := Age_30 - Pay;
        endif /*Age_30*/
    else
        Age_60 := Age_60 - Pay;
    endif /*Age_60*/
else
    Age_90 := Age_90 - Pay;
endif /*Age_90*/

ENDDO /*inner WHILE*/
print Number, Curr_Age, Age_30, Age_60, Age_90;
Curr_Age, Age_30, Age_60, Age_90 := 0;
endif;

if (C_Year >= Year of Date of record of account_file)
   Y_Day := (C_Year - Year of the Date) * 365
endif;
if (C_Mon >= MONTH of Date of record of account_file)
   M_Day := (C_Mon - Month of the Date) * 30
else
   M_Day := (C_Mon - Month of the Date) * 30 + 365
   Y_Day := Y_Day - 365;
endif;
if (C_Date >= Day of Date of record of account_file)
   D_Day := (C_Date - Day of the Date);
else
   D_Day := (C_Date - Day of the Date) + 30;
   M_Day := M_Day - 30;
endif;

Date_count := Y_Day + M_Day + D_Day;

Case Date_count of
   Date_count<31 : Curr_Age := Curr_Age + Charge;
   30<Date_count<61 : Age_30 := Age_30 + Charge;
   60<Date_count<91 : Age_60 := Age_60 + Charge;
   Date_count>90 : Age_90 := Age_90 + Charge;
endcase;

END WHILE *outer while*

end;
procedure Billing;

begin

Display_Account_Menu(Account_Option);

case Account_Option of
'1': Setup_New_Account;
'2': Input_Charge;
'3': Input_Payment;
'4': Bill_Patient;
'5': Bill_Inquiry;
'6': Print_Insurance;
'7': Aging_Charge;
'8': Monthly_Report;
'R': return to main menu;
end case;

end;

-------------------------------------------------------------------------------------------------------------------------------------

procedure Bill_Inquiry;

begin

Get_Patient_Name(Patient_Name);
Find_Bill_Info(Patient_Name);

end;
MODULE NAME: BILL_Patient

procedure BILL_Patient;

begin

  Get_BILL_Date( BILL_Date);
  WHILE Number DO
    Read_Account_File( Number, Charge_Info);
    Read_Payment_File( Number, Payment_Info);
    Read_Patient_File( Number, Patient_Info);
    Print_Bill( BILL_Date, Charge_Info, Payment_Info,
               Patient_Info);

  ENDDO

end;

******************************************************************************

MODULE NAME: Build_New_Information

procedure Build_New_Information( in: New_Info );

begin

  Add a new record to Patient_file with a new Patient_Number;
  Copy New_Info to the new record;

end;
MODULE NAME: Cancel_Appointment

procedure Cancel_Appointment

begin

    Get_Name( Patient_Name );
    Remove_Appointment( Patient_Name );

end;

MODULE NAME: Charge_Patient_Number

procedure Charge_Patient_Number( out: Patient_Number );

begin

    read Patient_Number from screen 9;

end;
MODULE NAME: Diagnosis

procedure Diagnosis;

begin

    Dis_Diag_Menu( Diag_Option );

    case Diag_Option of
        'I': Input_Patient_Information;
        'D': Do_Diagnosis;
    end case;

end;

MODULE NAME: Dis_Diag_Menu

procedure Dis_Diag_Menu( out: Diag_Option );

begin

    Display menu for selecting Diag_Option;

    repeat

        accept Diag_Option from screen;

        until Diag_Option not in (I, D, R);

    end;

end;
procedure Display_Account_Menu(out: Account_Option);

begin

    Show_Account_Menu;

    repeat

        Get_Account_Option(Account_Option);

        until Account_Option not in (1,2,3,4,5,6,7,8,R);

    end;

******************************************************************************

MODULE NAME: Display_Appointment

procedure Display_Appointment(out: Avail);

begin

    Accept Date from screen;
    Access Appointment_file using the Date as key;

    for each Patient_Name of the Date:
        Display the Appointment record;

    Accept Avail;

end;
procedure Display_appointment_Menu(out: Appointment_Option);
  begin
    Show_Appointment_Menu;
    repeat
      Get_Appointment_Option(Appointment_Option);
    until Appointment_Option not in (F,S,M,C,P,E);
  end;

MODULE NAME: Display_appointment_Record MODULE NUMBER: 2.2.3

procedure Display_Appointment_Record(in: Appointment);
  begin
    Display the appointment record in the form of SCREEN 3;
  end;
MODULE NAME: Display_Cancel_Record

procedure Display_Cancel_Record( in: Patient_Number);

begin

    find Appointment_Record in the Appointment_file;
    display the record in the form of SCREEN 5;

end;

MODULE NAME: Display_Charge

procedure Display_Charge( in: Patient_Number, out: Total_Charge);

begin

    repeat

        read next record in Account_file using Patient_Number as key;
        display Date, Service, Charge on screen 11;
        add Charge to Total_Charge;

    until Patient_Number is false;

end;
MODULE NAME: Display_menu

procedure Display_Menu( out: Option );

begin

    Show_Menu;

    repeat

        Get_Option( Option );

        until Option not in (A,B,D,M,E);

end;

MODULE NAME: Display_Modify_menu

procedure Display_Modify_Menu( in: Patient_Name, out: Modify_Option );

begin

    display Appointment Modification Menu in the form of screen 6;
    access Appointment file using the Patient_Name as key;
    display the Appointment Record on the screen.
    read Modify_Option;

end;
MODULE NAME: Display_Patient_Info
MODULE NUMBER: 3.6.2.1

procedure Display_Patient_Info( in: Patient_Name, out: Patient_Number );
begin
  find a record in Patient_file using Patient_Name as key;
  display Patient_Name, Patient_Number, Address, Phone on screen 11;
  copy Patient_Number of the record to Patient_Number;
end;

MODULE NAME: Display_Payment
MODULE NUMBER: 3.6.2.3

procedure Display_Payment( in: Patient_Number, out: Total_Payment);
begin
  repeat
    read next record in Payment_file using Patient_Number as key;
    display date, Payment_Type, Payment on screen 11;
    add Payment to Total_Payment;
  until Patient_Number is false;
end;
MODULE NAME: Display_Patient_History

begin
    display Patient_History in the form of SCREEN 12;
end;

MODULE NAME: Display_Update_Menu

begin
    display SCREEN 13;
    prompt and read Update_Option;
end;
MODULE NAME: Do_Diagnosis

procedure Do_Diagnosis;

begin

    Get_Patient_Number( Patient_Number);
    Request_Patient_History( Patient_History);
    Display_Patient_History( Patient_History);

end;

MODULE NAME: Find_Appointment

procedure Find_Appointment;

begin

    Get_Patient_Name( Patient_Name);
    Find_Appointment_Record( Appointment);
    Display_Appointment_Record( Appointment);

end;
procedure Find_Appointment_Record( out: Appointment );

begin

    find a record in Appointment_file using Patient_Name as key;

end;

MODULE NAME: Find_Bill_Info
MODULE NUMBER: 3.6.2

procedure Find_Bill_Info( in: Patient_Name );

begin

    Display_Patient_Info( Patient_Name, Patient_Number );
    Display_Charge( Patient_Number, Total_Charge );
    Display_Payment( Patient_Number, Total_Payment );
    Write_balance( Total_Charge, Total_Payment );

end;
MODULE NAME: Get_Account_Option
 MODULE NUMBER: 3.1.2

procedure Get_Account_Option( out: Account_Option );

begin

    read account_Option);

end;


MODULE NAME: Get_Appointment_Info
 MODULE NUMBER: 2.3.2.1

procedure Get_appointment_Info( out: Appointment_Info );

begin

    read Patient_Name;
    read Date;
    read Time;
    read Min;
    read doctor;
    read Reason;

end;
MODULE NAME: Get_Appointment_Option
MODULE NUMBER: 2.1.2

procedure Get_Appointment_Option( out: Appointment_Option );

begin

    write( 'Enter Selection Number: ');
    read Appointment_Option;

end;

***************************************************************************************

MODULE NAME: Get_New_Information
MODULE NUMBER: 4.2.1

procedure Get_New_Information( out: New_Info );

begin

    display SCREEN 8;

    repeat

        read Patient_Name, Patient_Number, Address, Phone, Date_of_Birth, Sex, Doctor, S.s#, Insured_Name, Resp_Party, and Medicare;
        ask user about correctness of the information(Y/N);

        until User_answer = Y;

        copy the information to New_Info;

end;
MODULE NAME: Get_Option  

procedure Get_Option( out: Option );

begin

    write( 'Enter Option Number :');
    read Option;

end;

MODULE NAME: Get_Patient_Name  

procedure Get_Patient_Name( out: Patient_Name );

begin

    display SCREEN 4;
    read Patient_Name;

end;
MODULE NAME: Get_Bill_Date

procedure Get_Patient_Number( out: Bill_Date );

begin

    accept Bill date, Month, and Year from screen;

end;

*****************************************************************************

MODULE NAME: Input_Charge

procedure Input_Charge;

begin

    display SCREEN 9;

    repeat

        Charge_Patient_Number( Patient_Number );
        repeat

            Read_Charge_Info( Patient_Number, Charge_Info );
            Add_Charge( Charge_Info );
            accept More_Charge;

        until More_Charge is false;

        accept More_Patient;

    until More_Patient is false;

end;
MODULE NAME: Input_Patient_Information  MODULE NUMBER: 4.2

procedure Input_Patient_Information;

begin

    Get_New_Information( New_Info );
    Build_New_Information( New_Info );

end;

*****************************************************************************

MODULE NAME: Input_Payment  MODULE NUMBER: 3.4

procedure Input_Payment;

begin

    display SCREEN 10;
    repeat

        Read_Patient_Number( Patient_Number );
        repeat
            Read_Payment_Info( Patient_Number, Payment_Info );
            Add_Payment( Payment_Info );
            accept More_Payment;
        until More_Payment is false;

        accept More_Patient;

    until More_Patient is false;

end;
MODULE NAME: Maintain_History
MODULE NUMBER: 5.0

procedure Maintain_History;

begin

    Display_Update_Menu( Update_Option );

    case Update_Option of
        'U': Update_History;
        'P': Print_Update_History;
        'R': Report_Service;
        'E': end;
    case end;

end;

MODULE NAME: Make_Appointment
MODULE NUMBER: 2.3.2.2

procedure Make_Appointment( in: Appointment_Info );

begin

    add a new record to Appointment_file;
    copy the Appointment_Info into the new record;

end;
procedure Modify_Appointment;
begin
  repeat
    Get_Patient_Name( Patient_Name );
    Display_Modify_Menu( Modify_Option );
    Make_Modification( Modify_Option );
    User_Modification_Confirm( Modify_OK );
  until Modification_OK;
end;

MODULE NAME: Make_Modification
MODULE NUMBER: 2.4.3
procedure Make_Modification( in: Modify_Option );
begin
  case Modify_Option of
    '1': read Patient_Name;
       replace Patient_Name in appointment_file with new one read;
    '2': read Date;
       replace Date in appointment_file with new one read;
    '3': read Time and Min;
       replace Time and Min in appointment_file with new ones read;
    '4': read Doctor;
       replace Doctor in Appointment_file with new one read;
    '5': read Reason;
       replace Reason in appointment_file with new one read;
    'R': return to SCREEN 2;
  end case;
end;
MODULE NAME: Print_Bill

procedure Print_Bill( in: Bill_Date, Charge_Info, Payment_Info, Patient_Info);

begin

    print Bill using Bill_Date, Patient_Info, Charge_Info, and Payment_Info;

end;

MODULE NAME: Print_History

procedure Print_History( in: Patient_Number );

begin

    while Patient_Number do

        read next record in Diagnosis_file using Patient_Number as key;
        print the record;

    end while;

end;
module name: Print_Patient_File
module number: 5.3.2

procedure Print_Patient_File( in: patient_Name, out: patient_Number );

begin

find a record in Patient_file using Patient_Name as key;
copy Patient_Number of the record to Patient_Number;
print Patient_Name, Patient_Number, S.s#, Address, Date_of_Birth,
and Doctor;

end;

module name: Print_Update_History
module number: 5.3

procedure Print_Update_History;

begin

repeat

Get_Patient_Name( Patient_Name );
Print_Patient_Name( Patient_Name, patient_Number );
Print_History( Patient_Number );

until Patient_Name is false;

end;
MODULE NAME: Read_Account_File

procedure Read_Account_File( out: Number, Charge_Info );
begin
    while Number = Patient_Number do
        read next record of Account_file;
        add Date, Service, Charge to Charge_Info;
        Number = Patient_Number;
    end while;
end;

MODULE NAME: Read_Charge_Info

procedure Read_Charge_Info( in: Patient_Number, out: Charge_Info );
begin
    read Date, Service, Charge from SCREEN 9;
    copy Patient_Number, Date, Service, and Charge to Charge_Info;
end;
MODULE NAME: Read_History_Update

procedure Read_History_Update(out: History_Update);

begin

    display SCREEN 14;
    accept Patient_Number, Vst_Date, D_Code, Diagnosis, and C_Notes
                     from screen;
    copy the information into History_Update;

end;

MODULE NAME: Read_Patient_File

procedure Read_Patient_File(in: Number, out: Patient_Info);

begin

    find a record in Patient_file using Patient_Number as key;
    copy Doctor, Patient_Name, Address and Phone of the record into
                     Patient_Info;

end;
procedure Read_Patient_Number(out: Patient_Number);
begin
  read Patient_Number from SCREEN 10;
end;

procedure Read_Payment_File(in: Number, out: Payment_Info);
begin
  while Patient_Number = Number do
    read next record of Payment_file;
    add Payment_Date, Payment_Type, and Payment to Payment_Info;
  end while;
end;
MODULE NAME: Read_Payment_Info

procedure Read_payment_Info( in: Patient_Number, out: payment_Info );

begin

    read Date, Payment_Type, and Payment from SCREEN 10;
    copy Patient_Number, Date, Payment_Type, and Payment into
        Payment_Info;

end;

MODULE NAME: Remove_Appointment

procedure Remove_Appointment( in: Patient_Name );

begin

    Display_Cancel_Record( Patient_Name );
    User_Cancel_Confirm( cancel_Ok );
    if Cancel_Ok is true then Remove_Record( Patient_Name );

end;
MODULE NAME: Remove_Record
MODULE NUMBER: 2.5.2.3

procedure Remove_Record( in: Patient_Name );

begin

    find a record in Appointment_file using Patient_Name as key;
    remove the record from Appointment_file;

end;

MODULE NAME: Report_Service
MODULE NUMBER: 5.4

procedure Report_Service;

begin

    accept Date from screen;

    while Date do
        read next record in Diagnosis_file using Date as key;
        add 1 to number of the Diagnosis;
    end while;
    print each Diagnosis, number of the Diagnosis onto Service_Report;

end;
procedure Report_Monthly_Account;

begin

   accept A_Year, A_Mon, A_day from screen;
   access Account_file;

   DO WHILE not end of file

      if (Month of Date of the record = A_Month and Year = A_Year) then
         T_Charge := T_Charge + Charge;
      
     END WHILE;

   access Payment_file;

   DO WHILE not end of file

      if (Month of Date of the record = A_Month and Year = A_Year) then
         T_Payment := T_Payment + Payment;
      
     END WHILE;

   T_Balance := T_Charge - T_Payment;

   print T_Charge, T_Payment, T_balance;

   end;
MODULE NAME: Request_patient_History
MODULE NUMBER: 4.3.1

procedure Request_Patient_History( in: Patient_Number,
                                  out: Patient_History);
begin

  find a record in Patient_file using Patient_Number as key;
  copy Patient_Name, Patient_Number, Date_of_Birth, Sex, Doctor into
  Patient_History;
  find a record in Diagnosis_file using Patient_Number as key
  while Patient_Number do
    read next record in Diagnosis_file using Patient_Number as key;
    add Vst_Date, D_Code, Diagnosis, and C_Notes to Patient_History;
  end while;
end;


MODULE NAME: Schedule
MODULE NUMBER: 2.3.2

procedure Schedule;
begin

  repeat
    Get_Appointment_Info( Appointment_Info );
    Verify_Info( Appointment_Info, Appointment_OK );
    until Appointment_OK;
    Make_Appointment( Appointment_Info );
  end;

MODULE NAME: Schedule_Appointment

module Schedule_Appointment;
begin
    Display_Appointment( Avail );
    if Avail is true then
        Schedule;
end;

MODULE NAME: Setup_New_Account

module Setup_New_Account;
begin
    Get_Patient_Name( out: Patient_Name );
    find a record in Patient_file using Patient_name as key;
    display Patient_Number of the record on the screen;
    if not found then display 'The Patient_Number has not been assigned'
end;
MODULE NAME: Show_Account_Menu  

procedure Show_Account_Menu;

begin

display SCREEN 7;

end;

MODULE NAME: Show_Appointment_Menu  

procedure Show_Appointment_Menu(out: Appointment_Option);

begin

display SCREEN 2;

end;
MODULE NAME: Show_Menu

procedure Show_Menu

begin

display SCREEN 1;

end;

MODULE NAME: System

procedure System;

begin

Display_Menu( Option);

case Option of
 'A': Appointment;
 'B': Billig;
 'D': Diagnosis;
 'M': Maintain_History;
 'E': end;
end case;

end;
MODULE NAME: Update_History

procedure Update_History;

begin

repeat

    Read_History_Update(History_Update);
    Add_Diagnosis_File(History_Update);

until Patient_Number is false;

end;

-----------------------------------------------------------

MODULE NAME: User_Cancel_Confirm

procedure User_Cancel_Confirm(out: Cancel_Ok);

begin

    prompt user answer about the cancelation of the appointment;
    if the answer is yes then Cancel_Ok := true;
    else Cancel_Ok := false;

end;
procedure User_Modification_Confirm(out: Modify_Ok);

begin

    prompt user answer about the modified appointment;
    if the answer is yes then Modify_Ok := true;
    else Modify_Ok := false;

end;

MODULE NAME: Verify_Info
MODULE NUMBER: 2.3.2.2

procedure Verify_Info(in: Appointment_Info, out: Appointment_Ok);

begin

    display Appointment_Info;
    write('Is this information correct?(Y/N): ');
    accept answer;
    if the answer is 'Y' then Appointment_Ok := true;
    else Appointment_Ok := false;

end;
procedure Write_Balance(in: Total_Charge, out: Total_Payment);

begin

    balance := Total_Charge - Total_Payment;
    display Balance on SCREEN 11;

end;
Appendix L.

Source Code
* Program : System.prg
* Author : Yong-Gi Kim
* Date : March 10, 1987
* Calls : Display_Menu.prg
*      : Appointment
*      : Billing
*      : Diagnosis
*      : Maintain_History
* File used : none
*
* Display Main screen and select...
*

SET DELIMITER OFF
SET ECHO OFF
SET BELL OFF
SET TALK OFF

Option = ' '
DO WHILE Option <> 'E'

Option = ' '
DO Display_Menu with Option

DO CASE

    CASE Option = '1'
        DO Appointment

    CASE Option = '2'
        DO Billing

    CASE Option = '3'
        DO Diagnosis

    CASE Option = '4'
        DO Maintain_History

    CASE Option = 'E'

ENDCASE

ENDDO
parameter Option

clear

\$ 5,18 say 'WELCOME TO CLINIC MANAGEMENT SYSTEM'
\$ 3,50 say DATE()  
\$ 6,18 say '-----------------------------'
\$ 9,20 say '1 => APPOINTMENT'
\$ 10,20 say '2 => BILLING'
\$ 11,20 say '3 => DIAGNOSIS'
\$ 12,20 say '4 => MAINTAINING HISTORY'
\$ 13,20 say 'E => END'
\$ 15,18 say '-----------------------------'

DO WHILE (Option='1') .AND. (Option='2') .AND.;
(Option='3') .AND. (Option='4') .AND. (Option='E'))

\$ 17,20 say 'Enter Option Number : ' GET Option
READ

ENDDO

return
Program : Appointment.prg
Author  : Yong-Gi Kim
Date    : March 20, 1987
Calls   : Dis_Appoint
          : Find_App
          : Sche_App
          : Modi_App
          : Canc_App
          : Prin_App

Called by : System
File used : none
Notes : Display appointment selection menu and select

SET DELIMITER OFF
SET ECHO OFF
SET BELL OFF
SET TALK OFF

Option = ' '  

DO Dis_Appoint with Option

DO CASE
  CASE Option = 'F'
    DO Find_App
  CASE Option = 'S'
    DO Sche_App
  CASE Option = 'M'
    DO Modi_App
  CASE Option = 'C'
    DO Canc_App
  CASE Option = 'P'
    DO Prin_App
  CASE Option = 'R'
    return
ENDCASE
SET DEVICE TO SCREEN
* Program : Dis_Appoint.prg
* Author : Yong-Gi Kim
* Date : March 20, 1987
* Calls : none
* Called by : Appointment
* File used : none
* Notes : Display Appointment selection menu and select
*

parameter Option

clear

3,29 say 'APPOINTMENT SCHEDULING'
3,50 say DATE()
6,23 say '-----------------------------'
9,26 say 'F => Find Appointment'
10,26 say 'C => Cancel Appointment'
11,26 say 'M => Modify Appointment'
12,26 say 'S => Schedule Appointment'
13,26 say 'P => Print Appointment Schedule'
14,26 say 'R => Return to Main screen'
16,23 say '-----------------------------'

DO WHILE (Option('F') .AND. (Option('C')) .AND. 
(Option('M') .AND. (Option('S')) .AND .
(Option('P') .AND. (Option('R')))

18,26 say 'Enter Selection Number : ' GET Option READ
ENDDO
return
# Find_App.prg

**Program**: Find_App.prg  
**Author**: Yong-Gi Kim  
**Date**: March 21, 1987  
**Calls**: Get_Name  
**Called by**: Appointment  
**File used**: Appointment_file  
**Notes**: Find an existing appointment using patient name as key

```plaintext
Name = '
ret = '

DO Get_Name with Name

USE Appointment_file
INDEX ON P_Name TO APNT

SEEK Name

clear

@ 3,10 say '---------------------------------------------'
@ 8,10 say 'PATIENT NAME : '
@ 9,10 say 'DATE OF APPOINTMENT : '
@ 10,10 say 'TIME OF APPOINTMENT : '
@ 11,10 say 'DOCTOR : '
@ 12,10 say 'REASON : '

@ 8,32 say Name
@ 9,32 say Date
@ 10,32 say Time
@ 10,34 say ':
@ 10,35 say Min
@ 11,32 say Doctor
@ 12,32 say Reason

@17,10 say '---------------------------------------------'

IF P_Name = '
    @15,12 say '*** The name input does not exist in file ***'
ENDIF

@20,15 say 'Hit "RETURN" to continue.' GET ret
READ

USE
erase APNT.NDX

return
```
parameter name

CLEAR

215,15 say 'Please, type first name and last name.'
210,15 say 'Enter Patient Name : ' GET name

READ

return
Program : Canc_App.prg
Author : Yong-Gi Kim
Date : March 27, 1987
Calls : Get_Name
Called by : Appointment
File used : Appointment_file
Notes : Cancel an existing appointment

Name = '
answer = '
rec_num = 0

Do Get_Name with Name

USE Appointment_file
INDEX ON P_Name TO APNT

SEEK Name

    IF P_Name = '
        clear
        310,10 say 'The name input does not exist in file.'
        Try again!' GET answer
        READ
        USE
        erase APNT.NDX
        return
    ENDIF
    clear

    3 10 say '----------------------------------------'
    3 10 say 'PATIENT NAME :
    3 10 say 'DATE OF APPOINTMENT :
    3 10 say 'TIME OF APPOINTMENT :
    3 10 say 'DOCTOR :
    3 10 say 'REASON :

    3 8,32 say Name
    3 9,32 say Date
    3 10,32 say Time
    3 10,34 say ':'
    3 10,35 say Min
    3 11,32 say Doctor
    3 12,32 say Reason

    3 17,10 say '----------------------------------------'
    3 20,15 say 'Are you sure to cancel?(Y/N) : ' GET answer
    READ
rec_num = RECNO()

USE
erase APNT.NDX

IF answer = 'Y'
USE Appointment_file
DELETE RECORD rec_num
ENDIF

USE

return
SET DELIMITER OFF

STORE 2 TO Line
STORE 0 TO P_Time
STORE 0 TO Col, Row
A_Date = ' ',
Avail = ' ',
Ok = ' '

Name = '
STORE 0 TO A_Time, A_Min
Dr = ',
A_reason = ',

\4,20 say '---------------------'
\5,20 say 'APPOINTMENT SCHEDULE'
\7,20 say '---------------------'
\9,25 GET A_Date PICTURE '99/99/99'
READ
STORE CTOD(A_Date) TO A_Date

\1,67 say 'DATE: '
\2,67 say A_Date
\1, 4 say ':00 :15 :30 :45'
DO WHILE (Line < 24)
\Line, 4 say '---------------------------------------------------------------'
\Line+1,4 say ' I I I I I I I I I'
\Line+1,1 say STR(P_Time,2,0)
Line = Line + 2
P_Time = P_Time + 1
IF (P_Time > 12)
    P_Time = P_Time - 12
ENDIF
ENDDO
\Line, 4 say '---------------------------------------------------------------'
USE Appointment_file
GOTO TOP

LOCATE ALL FOR (Date = A_Date)

DO WHILE .NOT. EOF()

Col = Min
  IF (Time > 7)
    Row = (Time - 7)*2 + 1
  ELSE
    ROW = (Time + 5)*2 + 1
  ENDIF

&Row, Col+5 say TRIM(P_Name)

CONTINUE

ENDDO

&21,67 say 'Available?'
&22,67 say '(Y/N): ' GET Avail
READ

USE

IF Avail = 'Y'

  DO SCHEDULE

ENDIF

USE

return
SET DELIMITER OFF

Name =
S_Date =
STORE 0 TO S_Time, S_Min
S_Doctor =
S_Reason =
answer =
clear

DO WHILE (answer('Y'))

GET Name
GET S_Date PICTURE '99/99/99'
GET S_Time PICTURE '99'
GET S_Doctor
GET S_Reason

GET answer(Y/N)

STORE CTOD(S_Date) TO S_Date

USE Appointment_file
APPEND BLANK

REPLACE P_Name with Name, Date with S_Date, Time with S_Time;
Min with S_Min Doctor with S_Doctor, Reason with S_Reason

USE

return
SET DELIMITER OFF
SET TALK OFF
SET BELL OFF
SET ECHO OFF

Name = ' 
M_Date = ' 
STORE 0 TO M_Time, M_Min 
M_Doctor = ' 
M_Reason = ' 
option = ' 

DO Get_Name with Name
USE Appointment_file
INDEX ON P_Name TO APNT.NDX
SEEK NAME

M_Date = DTOC(Date)
M_Time = Time
M_Min = Min
M_Doctor = Doctor
M_Reason = Reason

clear

2 2,25 say '-------------------------'
2 3,25 say 'MODIFY APPOINTMENT'
2 4,25 say '-------------------------'

2 5,10 say '-----------------------------'
2 9,10 say '1. PATIENT NAME : ' 
2 10,10 say '2. DATE (mm/dd/yy) : ' 
2 11,10 say '3. TIME (tt:mm) : ' 
2 12,10 say '4. DOCTOR : ' 
2 13,10 say '5. REASON : ' 
2 17,10 say '-------------------------------'
DO WHILE (option<>'R')
   DO CASE
      CASE option = '1'
         GET Name
      CASE option = '2'
         GET M_Date PICTURE '99/99/99'
      CASE option = '3'
         GET M_Time PICTURE '99'
      CASE option = '4'
         GET M_Doctor
      CASE option = '5'
         GET M_Reason
   ENDCASE
   GET option
   READ
ENDDO
STORE CTOD(M_Date) TO M_Date
REPLACE P_Name with Name, Date with M_Date, Time with M_Time;
               Min with M_Min, Doctor with M_Doctor, Reason with M_Reason
USE
erase APNT.NDX
return
SET DELIMITER OFF
SET ECHO OFF
SET BELL OFF
SET TALK OFF

P_Date = ';
STORE 5 TO count
STORE 1 TO num

clear
@d 8,25 say '-----------------
@d 9,25 say 'PRINT APPOINTMENT'
@d 10,25 say '----------
@d 15,20 say 'Enter Date of Appointment.'
@d 17,20 say 'Month/Day/Year : ' GET P_Date PICTURE '99/99/99'
@d 19,20 say '** PRINTER Should be turned on! **'
READ

STORE CTOD(P_Date) TO P_Date

USE Appointment_file

clear

SET DEVICE TO PRINT
@d 1,15 say '*** APPOINTMENT ***
@d 1,45 say P_Date
@d 3,8 say 'No Patient Name
@d 4,8 say 'Time Doctor Reason'
@d 4,8 say '---

STORE CTOD(P_Date) TO P_Date
SORT TO ST ASCENDING ON APM, Time, Min FOR (Date = P_Date)

USE ST
GOTO TOP
LOCATE ALL FOR (Date = P_Date)

DO WHILE .NOT. EOF()

@ count, 0 say num
@ count,12 say P_Name
@ count,33 say Time
@ count,35 say ':'
@ count,36 say Min
@ count,42 say Doctor
@ count,51 say Reason

STORE count + 1 TO count
STORE num + 1 TO num

CONTINUE
ENDDO
@ count,8 say '--------------------------------------------------------'
@ count + 1,8 say ' ' USE
erase ST.dbf

return
* Program : Billing.prg
* Author  : Yong-Gi Kim
* Date    : March 31, 1987

SET DELIMITER OFF
SET ECHO OFF
SET BELL OFF
SET TALK OFF

Option = ''

DO Dis_Account with Option

DO CASE

CASE Option = '1'
    DO Setup
CASE Option = '2'
    DO In_Cha
CASE Option = '3'
    DO In_Pay
CASE Option = '4'
    DO Bill
CASE Option = '5'
    DO Bill_Inq
CASE Option = '6'
    DO Prin_Ins
CASE Option = '7'
    DO Aging
CASE Option = '8'
    DO Mon_Acc
CASE Option = 'R'
    return

ENDCASE

return
Program: Dis_Account.prg
Author: Yong-Gi Kim
Date: March 31, 1986
Calls: none
Called by: Billing.prg
File used: none
Notes: Display Accounting Menu

parameter Option

clear

@ 5,33 say 'BILLING'
@ 3,50 say DATE()
@ 7,23 say '----------------------------------'
@ 9,26 say '1 => Set Up New Account'
@10,26 say '2 => Input Patient Charge'
@11,26 say '3 => Input Patient Payment'
@12,26 say '4 => Bill Patient'
@13,26 say '5 => Bill Inquiry from Patient'
@14,26 say '6 => Print insurance Form'
@15,26 say '7 => Aging Charges of all Patients'
@16,26 say '8 => Report Monthly Account'
@17,26 say 'R => Return to Main Menu'
@19,23 say '----------------------------------'

DO WHILE ((Option)'1') .AND. (Option)'2') .AND. (Option)'3') .AND.;
(Option)'4') .AND. (Option)'5') .AND. (Option)'6') .AND.;
(Option)'7') .AND. (Option)'8') .AND. (Option)'R')

@21,26 say 'Enter Selection Number : ' GET Option
READ

ENDDO

return
* Program : Setup.prg
* Author : Yong-Ki Kim
* Date : March 31, 1987
* Called by : Billing.prg
* File used : Patient_file.dbf
* Notes : This Setup.prg Confirms Patient_Number which is already assigned in Diagnosis.

If the Patient number is not found return to Diagnosis phase and input patient information (assign number).

Name = ', Ans = ',

DO Get_Name with Name

USE Patient_file
GOTO TOP
LOCATE ALL FOR (P_Name = Name)

DO WHILE ((.NOT. EOF()) .AND. (Ans = 'Y'))

CLEAR
2 5,10 SAY '-------------------------------------------------------------'
2 7,10 SAY 'Patient Name : '
2 8,10 SAY 'Patient Number : '
2 9,10 SAY 'Patient Address : '
210,10 SAY 'Phone : '
211,10 SAY 'Date of Birth : '
212,10 SAY 'Doctor : '
213,10 SAY '-------------------------------------------------------------'

2 7,28 SAY P_Name
2 8,28 SAY P_Number
2 9,28 SAY Address
210,28 SAY Phone
211,28 SAY DOB
212,28 SAY Doctor
215,10 SAY 'Is This Patient Information Correct?(Y/N) : '; GET Ans

READ
CONTINUE
ENDDO

IF Ans = 'Y'
218,10 SAY 'If you cannot find the patient number, Please input'
219,10 SAY 'Patient information and assign the patient number,'
220,10 SAY 'Hit "Return" to continue. ' GET Ans
READ
ENDIF
USE
return
STORE 0 TO Number
I_Date = ' ' 
I_Service = ' ' 
STORE 0.00 TO I_Charge
Ans1 = ' ' 
Ans2 = ' ' 

clear
3,50 say DATE( )
4,17 say '-----------------------------'
5,17 say 'INPUT CURRENT PATIENT CHARGE'
6,17 say '-----------------------------'

DO WHILE (Ans2<>'N')

8,10 say '-------------------------------'
10,10 say '1. PATIENT NUMBER : ' GET Number
DO WHILE (Ans1<>'N')

12,10 say '2. DATE : ' GET I_Date PICTURE '99/99/99'
14,10 say '3. SERVICE : ' 
16,10 say '4. CHARGE : ' 
18,10 say '-------------------------------'
20,10 say 'More charge to input for this patient?(Y/N): '
22,10 say '*** If the SERVICE is common SPECIAL SERVICE, IMMUNIZATION,' 
23,10 say 'PHYSICAL EXAM, or PATHOLOGY & LAB, enter SERVICE CODE ***'
31 GET I_Service
READ

22,10 say ' '
23,10 say ' '

USE Fee_file
LOCATE FOR Code=TRIM(I_Service)

IF EOF() .OR. I_Service= ' '
ELSE
    214,38 say Service
    216,31 say Fee
    I_Service = Service
    I_Charge = Fee
ENDIF

220,57 GET Ans1
READ

IF Number<>0
    STORE CTOD(I_Date) TO I_Date
    USE Account_file
    APPEND BLANK
    REPLACE P_Number with Number, Date with I_Date,;
        Service with I_Service, Charge with I_Charge
    STORE DTOC(I_Date) TO I_Date
    USE
ENDIF
ENDDO

Ans1 = '
222,10 say 'More patient to input charge? (Y/N): ' Get Ans2
READ
  222,10 say '
ENDDO

return
STORE 0 TO Number
P_Date = 
P_Type = 
STORE 0.00 TO P_Payment
Ans1 = 
Ans2 = 

clear
@ 3,50 say DATE()
@ 4,22 say '----------'
@ 5,22 say 'INPUT PAYMENT'
@ 6,22 say '----------'

DO WHILE <Ans2>='N'

@ 8,10 say '-----------------------------------------------------'
@10,10 say '1. PATIENT NUMBER : ' GET Number

DO WHILE <Ans1>='N'

@12,10 say '2. DATE : ' GET P_Date PICTURE '99/99/99'
@14,10 say '3. PAYMENT TYPE : ' GET P_Type
@15,12 say ' (H: Check, A: Cash, R: Credit Card)'
@17,10 say '4. PAYMENT : ' GET P_Payment PICTURE '99999.99'
@19,10 say '-----------------------------------------------------'
@21,10 say 'More payment to input for this patient?(Y/N): '; GET Ans1
READ

IF Number<>0
    STORE CTOD(P_Date) TO P_Date
    USE Payment_file
    APPEND BLANK
    REPLACE P_Number with Number, Date with P_Date,;
    Pay_Type with P_Type, Payment with P_Payment
    STORE DTOC(P_Date) TO P_Date
    USE
ENDIF
ENDDO

Ans1 = ' '
#22,10 say 'More patient to input payment?(Y/N): ' Get Ans2
READ
#22,10 say '

ENDDO
return
* Program : Bill.prg
* Author : Yong-Gi Kim
* Date : April 1, 1987
* Calls : Acc_Date.prg
* Called by : Billing
* File used : Patient_file.dbf, Account_file.dbf, Payment_file.dbf
* Notes : Print Bill to Patient
*
Number = 0
prt = ' ',
title = 'Y'

STORE 13 TO count
Name = '
Adrs = '
STORE 0.00 TO Balance, Pay, Curr_Age, Age_30, Age_60, Age_90
STORE 0 TO Date_Count, Y_Day, M_Day, D_Day
STORE 0 TO B_Year, B_Mon, B_Date, Number, temp

DO Acc_Date with B_Mon, B_Date, B_Year
clear

USE Account_file
SORT TO AST ASCENDING ON P_Number

USE Patient_file
INDEX ON P_Number TO PTNT

SELECT 1
USE AST

SELECT 2
USE Payment_file

SELECT 3
USE Patient_file INDEX PTNT

SELECT 1

Number = P_Number

DO WHILE .NOT. EOF() .OR. EOF()

IF (P_Number <> Number )
    temp = P_Number
    title = 'Y'
SELECT 2

STORE count + 1 TO count
@count, 5 say '-------------------------------'
STORE count + 2 TO count
@count, 5 say 'PAYMENT'
STORE count + 1 TO count
@count, 5 say '-------------------------------'
STORE count + 1 TO count

GOTO TOP
LOCATE ALL FOR (P_Number = Number)

DO WHILE .NOT. EOF()
    STORE Payment TO Pay
    IF (Pay>Age_90)
        Pay = Pay - Age_90
        Age_90 = 0.00
    ENDIF
    IF (Pay>Age_60)
        Pay = Pay - Age_60
        Age_60 = 0.00
    ENDIF
    IF (Pay>Age_30)
        Pay = Pay - Age_30
        Age_30 = 0.0
        Curr_Age = Curr_Age - Pay
    ELSE
        Age_30 = Age_30 - Pay
    ENDIF
    ELSE
        Age_60 = Age_60 - Pay
    ENDIF
    ELSE
        Age_90 = Age_90 - Pay
    ENDIF

    IF ((YEAR(Date)= B_Year) .AND. (MONTH(Date) = B_Mon))
        @count, 5 say Date
        @count, 15 say Pay_Type
        @count, 39 say Payment
        STORE count + 1 TO count
    ENDIF

    STORE Balance - Payment TO balance
CONTINUE
ENDDO
STORE count + 2 TO count
ACCOUNT 5 say 'ACUMMULATIVE BALANCE'
STORE count + 1 TO count
ACCOUNT 5 say '--------------------------------------'
ACCOUNT +1,34 say Balance

STORE count + 2 TO count
ACCOUNT 5 say '--------------------------------------'
STORE count + 2 TO count
ACCOUNT 5 say 'Current > 30days > 60days > 90days,'
STORE count + 1 TO count
ACCOUNT 0 say Curr_Age
ACCOUNT 12 say Age_30
ACCOUNT 25 say Age_60
ACCOUNT 39 say Age_90
ACCOUNT +1,5 say '--------------------------------------'

STORE 0.00 TO Curr_Age, Age_30, Age_60, Age_90, Balance
STORE 13 TO count

Number = temp

SET DEVICE TO SCREEN

ENDIF * P_Number<>Number *

IF title = 'Y'

SET DEVICE TO PRINT
SELEC 3

SEEK NUMBER
ACCOUNT 1,10 say Doctor
ACCOUNT 1,42 say STR<B_Mon,2,0>
ACCOUNT 1,44 say '/'
ACCOUNT 1,45 say STR<B_Year,4,0>
ACCOUNT 2,10 say 'Drawer 378, Brooklyn & Roosevelt, Superior,'
ACCOUNT 3,10 say 'MT 59872. Telephone (406) 822-4803'
ACCOUNT 4,5 say '--------------------------------------'
ACCOUNT 6,5 say 'Name :
ACCOUNT 6,15 say P_Name
ACCOUNT 7,5 say 'Address :
ACCOUNT 7,15 say Address
ACCOUNT 9,5 say '--------------------------------------'
ACCOUNT 11,5 say 'CHARGE'
ACCOUNT 12,5 say '--------------------------------------'
title = 'N'

ENDIF
SELECT 1
IF (B_Year >= YEAR(Date))
    Y_Day = (B_Year - YEAR(Date)) * 365
ENDIF

IF (B_Mon >= MONTH(Date))
    M_Day = (B_Mon - MONTH(Date)) * 30
ELSE
    M_Day = (B_Mon - MONTH(Date)) * 30 + 365
    Y_Day = Y_Day - 365
ENDIF

IF (B_Date >= DAY(Date))
    D_Day = (B_Date - DAY(Date))
ELSE
    D_Day = (B_Date - DAY(Date)) + 30
    M_Day = M_Day - 30
ENDIF

Date_count = (Y_Day + M_Day + D_Day)

DO CASE
    CASE (Date_Count < 31)
        Curr_Age = Curr_Age + Charge
    CASE (Date_Count)30 .AND. Date_Count(61)
        Age_30 = Age_30 + Charge
    CASE (Date_Count)60 .AND. Date_Count(91)
        Age_60 = Age_60 + Charge
    CASE (Date_Count)90)
        Age_90 = Age_90 + Charge
ENDCASE

IF ((YEAR(Date) = B_Year) .AND. (MONTH(Date) =B_Mon))
    count, 5 say Date
    count,15 say Service
    count,40 say Charge
    STORE count + 1 TO count
ENDIF

STORE Balance + Charge TO Balance

IF EOF()
    EXIT
ENDIF

SKIP
ENDDO  *WHILE*
@count+1,5 say ' '  
CLOSE DATABASE  
erase AST.dbf  
erase PTNT.NDX  

SET DEVICE TO SCREEN  
return
Parameter B_Mon, B_Date, B_Year
prt = ''

clear
@ 2,20 say '----------'
@ 3,20 say 'PRINT BILL'
@ 4,20 say '----------'

DO WHILE (prt <> 'Y')
    @ 6,10 say '-----------------------------------------------'
    @ 8,10 say 'Enter Bill Month, Date, Year'
    @10,10 say '1. Month<99> : ' GET B_Mon PICTURE '99'
    @11,10 say '2. Date <99) : ' GET B_Date PICTURE '99'
    @12,10 say '3. Year<9999) : ' GET B_Year PICTURE '9999'
    @14,10 say '-----------------------------------------------'
    @16,10 say 'Is this correct Bill Date?<Y/N)' GET prt
    @22,10 say '*** PRINTER MUST BE TURNED ON ***'
READ
ENDDO

return
Name = 'Name = 'Ans = 'Ans = 'STORE 7 TO cnt1, cnt2
STORE 0.00 TO T_Charge, T_Payment, Balance
STORE 0 TO Number

DO Get_Name with Name

USE Patient_file
GOTO TOP
LOCATE ALL FOR <P_Name = Name>

DO WHILE (NOT. EOF()) .AND. (Ans = 'Y')

clear

1, 1 say 'NAME: '
1, 30 say 'NUMBER: '
2, 1 say 'ADDRESS: '
3, 1 say '-----------------------------------------------
------'
1, 7 say P_Name
1, 40 say P_Number
2, 10 say Address

10, 1 say 'Is this the patient whom you are searching?(Y/N): ';
    GET Ans
READ

IF (Ans = 'Y')
    STORE P_Number TO Number
ENDIF

CONTINUE
ENDDO
210,1 say ' 
IF (Ans = 'Y')

24, 7 say 'C H A R G E 
25, 1 say 'DATE * SERVICE * CHARGE PAYMENT' 
26, 1 say '--------- --------- --------- ---------' 

USE Account_file 
GOTO TOP 
LOCATE ALL FOR (P_Number = Number) 

DO WHILE (.NOT. EOF()) 
  @cnt1, 1 say Date 
  @cnt1, 10 say Service 
  @cnt1, 25 say Charge 
  T_Charge = T_Charge + Charge 
  cnt1 = cnt1 + 1 
  CONTINUE 
ENDDO 

USE Payment_file 
GOTO TOP 
LOCATE ALL FOR (P_Number = Number) 

DO WHILE (.NOT. EOF()) 
  @cnt2, 40 say Date 
  @cnt2, 51 say Pay_Type 
  @cnt2, 62 say Payment 
  T_Payment = T_Payment + Payment 
  cnt2 = cnt2 + 1 
  CONTINUE 
ENDDO 

Balance = T_Charge - T_Payment 
220, 10 say 'TOTAL CHARGE ***** TOTAL PAYMENT ***** BALANCE' 
221, 9 say T_Charge 
221, 34 say T_Payment 
221, 56 say Balance 
ENDIF 
USE 
222, 1 say '-----------------------------------------------; 
--------------' 
STORE ' ' TO Ans 
223, 15 say 'Hit "Return" to continue. ' Get Ans 
READ 
return
STORE 0 TO Number
STORE 0.00 TO L_Charge
prt = ' ',
S_Date = ' ',
S_Place = ' ',
P_Code = ' ',
clear

@ 1,15 say '--------------------------------------------'
@ 2,15 say 'PRINT INSURANCE FORM'
@ 3,15 say '--------------------------------------------'

DO WHILEprt<> 'Y'

@ 5,10 say '--------------------------------------------'
@ 7,10 say '1. PATIENT NUMBER : ' GET Number
@ 8,10 say '2. DATE OF SERVICE : ' GET S_Date PICTURE '99/99/99'
@ 9,10 say '3. PLACE OF SERVICE : ' GET S_Place
@ 10,10 say '4. PROCEDURE CODE : ' GET P_Code
@ 11,10 say '5. LAB. CHARGE : ' GET L_Charge
@ 12,10 say 'OUTSIDE OFFICE '
@ 14,10 say '--------------------------------------------'
@ 16,10 say 'Is this Information Correct?(Y/N) ; ' GET prt
@ 20,10 say '*** PRINTER MUST BE TURNED ON ***'
READ
ENDDO

SET DEVICE TO PRINT

clear

@ 1,10 say '--------------------------------------------'
@ 2,10 say 'HEALTH INSURANCE CLAIM FORM'
@ 3,10 say '--------------------------------------------'
@ 5,4 say '*** PATIENT AND INSURED INFORMATION ***'
USE Patient_file
INDEX ON P_Number TO PTNT

SEEK Number

@ 7, 1 say '1. PATIENT NAME :'
@ 7,18 say P_Name
@ 8, 1 say '2. DATE OF BIRTH :'
@ 8,18 say DOB
@ 9, 1 say '3. INSURED NAME :'
@ 9,18 say Ins_Name
@10, 1 say '4. PATIENT ADDRESS :'
@10,18 say Address
@11, 1 say '5. PATIENT SEX :'
@11,18 say SEX
@12, 1 say '6. INSURED ADDRESS :'
@12,18 say R_Party

@14, 4 say '*** PHYSICIAN OR SUPPLIER INFORMATION ***'
@16, 1 say '7. LAB. CHARGE OUTSIDE OFFICE :'
@16,31 say L_Charge
@17, 1 say '8. PHYSICIAN NAME :'
@17,18 say Doctor

STORE CTOD(S_Date) TO S_Date

USE Diagnosis_file
LOCATE FOR P_Number=Number .AND. Vst_Date=S_Date

@18, 1 say '9. SERVICE DATE :'
@18,20 say Vst_Date
@19, 1 say '10. SERVICE PLACE :'
@19,20 say S_Place
@20, 1 say '11. PROCEDURE CODE :'
@20,20 say P_Code
@21, 1 say '12. SERVICE DESCR. :'
@21,20 say C_Notes
@22, 1 say '13. DIAGNOSIS CODE :'
@22,20 say D_Code

USE Account_file
INDEX ON P_Number + Date TO ACNT

SEEK Number + S_Date

@23, 1 say '14. SERVICE :'
223,20 say Service
224,1 say '15.CHARGE :'
224,20 say Charge
225,1 say ''

CLOSE DATABASE
erase ACNTndx
erase PTNTndx

SET DEVICE TO SCREEN

return
* Program : Aging.prg
* Author : Yong-Gi Kim
* Date : April 30, 1987
* Calls :
* Called by : Billing
* File used : Account_file.dbf, Payment_file.dbf
* Notes : Aging Charges of all patients
*

SET DELIMITER OFF
SET TALK OFF

STORE 0 TO C_Year, C_Mon, C_Date, Number, temp
Name = ' 
STORE 0.00 TO Pay, Curr_Age, Age_30, Age_60, Age_90
STORE 0.00 TO T_Curr_Age, T_30_Age, T_60_Age, T_90_Age
STORE 0.00 TO Sum_Charge
STORE 0 TO Date_Count, Y_Day, M_Day, D_Day
prt = ' 

clear

t 2,15 say '----------------------------------------'
t 3,15 say 'AGING CHARGES OF ALL PATIENTS'
t 4,15 say '----------------------------------------'

DO WHILE Prt<> 'Y'

t 7,15 say 'Enter Current Month(99) :' GET C_Mon PICTURE '99'
t 8,15 say 'Enter Current Date (99) :' GET C_Date PICTURE '99'
t 9,15 say 'Enter Current Year(9999) :' GET C_Year PICTURE '9999'
t11,15 say '----------------------------------------'
t12,15 say 'Is this Correct Date?(Y/N) :' GET prt
	15,15 say '*** PRINTER MUST BE TURNED ON ***'

READ

ENDDO

USE Account_file

SORT TO AST ASCENDING ON P_Number

SELECT 1
USE AST

SELECT 2
USE Payment_file
SELECT 1
SET DEVICE TO PRINT

CLEAR

1, 25 SAY '*** MONTHLY AGED ACCOUNTING ***'
1, 65 SAY STR(C_Mon,2,0)
1, 67 SAY '/'
1, 68 SAY STR(C_Year,4,0)
2, 10 SAY '-------------------------------------------------
3, 10 SAY 'Pat. Number Curr >30days >60days >90days'
4, 10 SAY '-------------------------------------------------

NUMBER = P_Number

DO WHILE .NOT. EOF() OR .EOF()

IF (P_Number = Number)

  TEMP = P_Number

  SELECT 2
  GOTO TOP

  LOCATE ALL FOR (P_Number = Number)

DO WHILE .NOT. EOF()

  STORE Payment TO Pay

  IF (Pay > Age_90)

    Pay = Pay - Age_90
    Age_90 = 0.00

    IF (Pay > Age_60)

      Pay = Pay - Age_60
      Age_60 = 0.00

      IF (Pay > Age_30)

        Pay = Pay - Age_30
        Age_30 = 0.00
        Curr_Age = Curr_Age - Pay

      ELSE

        Age_30 = Age_30 - Pay
      ENDIF

    ELSE

      Age_60 = Age_60 - Pay
    ENDIF

  ELSE

    Age_60 = Age_60 - Pay
  ENDIF

ENDIF
ELSE
  Age_90 = Age_90 - Pay
ENDIF
CONTINUE
ENDDO  *WHILE*

5 say Number
15 say Curr_Age
30 say Age_30
45 say Age_60
60 say Age_90

T_Curr_Age = T_Curr_Age + Curr_Age
T_30_Age = T_30_Age + Age_30
T_60_Age = T_60_Age + Age_60
T_90_Age = T_90_Age + Age_90

Number = temp
STORE 0.00 TO Curr_Age, Age_30, Age_60, Age_90
STORE count + 1 TO count
ENDIF

SELECT 1

IF (C_Year >= YEAR(Date))
  Y_Day = (C_Year - YEAR(Date)) * 365
ENDIF

IF (C_Mon >= MONTH(Date))
  M_Day = (C_Mon - MONTH(Date)) * 30
ELSE
  M_Day = (C_Mon - MONTH(Date)) * 30 + 365
  Y_Day = Y_Day - 365
ENDIF

IF (C_Date >= DAY(Date))
  D_Day = (C_Date - Day(Date))
ELSE
  D_Day = (C_Date - Day(Date)) + 30
  M_Day = M_Day - 30
ENDIF

Date_Count = (Y_Day + M_Day + D_Day)
DO CASE
  CASE (Date_Count < 31)
    Curr_Age = Curr_Age + Charge
  CASE (Date_Count > 30 AND Date_Count < 61)
    Age_30 = Age_30 + Charge
  CASE (Date_Count > 60 AND Date_Count < 91)
    Age_60 = Age_60 + Charge
  Case (Date_Count > 90)
    Age_90 = Age_90 + Charge
ENDCASE

IF EOF() THEN
  EXIT
ENDIF

SKIP
ENDDO * WHILE *

CLOSE DATABASE
erase AST.dbf

Sum_Charge = T_Curr_Age + T_30_Age + T_60_Age + T_90_Age

@count,10 say '-----------------------------------------------'
@count+1,17 say ' TOT_Curr   TOT_30   TOT_60   TOT_90'
@count+2,15 say T_Curr_Age
@count+2,30 say T_30_Age
@count+2,45 say T_60_Age
@count+2,60 say T_90_Age
@count+3,10 say '-----------------------------------------------'
@count+4,15 say 'Total_Charge :'
@count+4,30 say Sum_Charge
@count+5,10 say '-----------------------------------------------'
@count+6,10 say ''

SET DEVICE TO SCREEN

return
STORE 0 TO Month, Year, Number, Temp
STORE 0 TO C_Count, P_Count, Tot_C_Count, Tot_P_Count
STORE 0.00 TO T_Charge, T_Payment, T_Balance
STORE 0.00 TO P_Charge, P_Payment, P_Balance
STORE 6 TO cnt
P_Date = '
prt = ' 

DO WHILE prt(>')'Y'
   212,15 say 'Enter the Month (<99) :
   212,39 GET Month PICTURE '99'
   214,15 say 'Enter the Year (<9999) :
   214,39 GET Year PICTURE '9999'
   216,15 say '-------------------------------'
   218,15 say 'Is this Date Correct?<Y/N>: ' GET prt 
   READ
ENDDO

SET DEVICE TO PRINT

clear
2 1,10 say '*** MONTHLY ACCOUNTING REPORT ***'

2 1,54 say STR<Month,2,0)
2 1,56 say '/'
2 1,57 say STR<Year,4,0)
2 2, 5 say '-------------------------------'
2 3, 5 say 'PATIENT CHARGE PAYMENT BALANCE CHARGE PAYMENT'
2 4, 5 say 'NUMBER COUNT COUNT'
USE Account_file
SORT TO AST ASCENDING ON P_Number

SELECT 1
USE AST

SELECT 2
USE Payment_file

SELECT 1
GOTO TOP

Number = P_Number

DO WHILE .NOT. EOF() .OR. EOF()

IF (P_Number<>Number)
    Temp = P_Number
    SELECT 2
    GOTO TOP

LOCATE ALL FOR ((MONTH(Date)=Month) .AND. (YEAR(Date)=Year);
                 .AND. (P_Number=Number))

DO WHILE (.NOT. EOF())

    STORE P_Payment + Payment TO P_Payment
    STORE T_payment + Payment TO T_Payment
    STORE P_Count + 1 TO P_Count
    STORE Tot_P_Count + 1 TO Tot_P_Count

    CONTINUE

ENDDO  * WHILE .NOT. EOF() of Payment_file *

STORE P_Charge - P_payment TO P_Balance

IF C_Count<>0 .OR. P_Count<>0
    @cnt, 5 say STR(Number,4,0)
    @cnt,10 say P_Charge
    @cnt,22 say P_Payment
    @cnt,35 say P_Balance
    @cnt,45 say C_Count
    @cnt,55 say P_Count

    STORE 0.00 TO P_Charge, P_Payment, P_Balance
    STORE 0 TO C_Count, P_Count
    STORE cnt + 1 TO cnt

ENDIF
Number = Temp
ENDIF * P_Number <> Number *

SELECT 1

IF MONTH(Date)=Month .AND. YEAR(Date)=Year
    STORE P_Charge + Charge TO P_Charge
    STORE T_Charge + Charge TO T_Charge
    STORE C_Count + 1 TO C_Count
    STORE Tot_C_Count + 1 TO Tot_C_Count
ENDIF

IF EOF() THEN
    EXIT
ENDIF

SKIP

ENDDO * WHILE of SELECT 1 *

STORE T_Charge - T_Payment TO T_Balance

\cnt, 5 say '----------------------------------------------'
STORE \cnt + 1 TO \cnt 
\cnt, 5 say 'TOTAL :
\cnt,10 say T_Charge
\cnt,22 say T_Payment
\cnt,35 say T_balance
\cnt,45 say Tot_C_Count
\cnt,55 say Tot_P_Count
\cnt+1,5 say '----------------------------------------------'
\cnt+1,5 say '

CLOSE DATABASE
erase AST.dbf

SET DEVICE TO SCREEN

return
Op = ' ' 

do Dis_Diag with Op 

DO CASE

    CASE Op = '1'
        do In_Info 
    CASE Op = 'D'
        do Do_Diag 
    CASE Op = 'R'
        return

ENDCASE
Parameter Op

clear

\# 4,29 say '----------'
\# 5,29 say 'DIAGNOSIS'
\# 6,29 say '----------'
\# 8,23 say '--------------------------------------------------'
\#10,26 say 'I => Input Patient Information'
\#12,26 say 'D => Search Patient History'
\#14,26 say 'R => Return to Main menu'
\#16,23 say '--------------------------------------------------'

DO WHILE ((Op)= 'I') .AND. (Op)= 'D' .AND. (Op)= 'R')

\#18,26 say 'Enter Selection Number : ' GET Op
READ

ENDDO

return
Program: In_Info.prg
Author: Yong-Gi Kim
Date: April 8, 1987
Calls: None
Called by: Diagnosis
File used: Patient_file.dbf
Notes: Input new patient's information into Patient_file

Ans = ' '
Name = '
STORE 0 TO Number
Adrs = '
Phn = '
Bthday = '
Sx = '
Dr = '
STORE 0 TO S_No
I_Name = '
Resp = '
STORE 0 TO Medi

DO WHILE (AnsO'Y')
clear
3 1,10 say '--------------------------'
3 2,10 say 'INPUT PATIENT INFORMATION'
3 3,10 say '--------------------------'
3 5, 1 say '1.Name : ' GET Name
3 8, 1 say '2.Patient No. : ' GET Number
3 9, 1 say '3.Address : ' GET Adrs
210, 1 say '4.Phone : ' GET Phn
211, 1 say '5.Date of Bth : ' GET Bthday PICTURE '99/99/99'
212, 1 say '6.Sex : ' GET Sx
213, 1 say '7.Doctor : ' GET Dr
214, 1 say '8.S.S No. : ' GET S_No PICTURE '999999999'
215, 1 say '9.Insured Name: ' GET I_Name
216, 1 say '10.Resp Party : ' GET Resp
217, 1 say '11.medicare No.: ' GET Medi
219, 1 say 'Is This Correct Information?(Y/N) : ' GET Ans
READ
ENDDO

STORE CTOD(Bthday) TO Bthday
USE Patient_file
APPEND BLANK

REPLACE P_Name with Name, P_Number with Number, Address with Adrs,
   Phone with Phn, DOB with Bthday, Sex with Sx,
   Doctor with Dr, SS with S_No, Ins_Name with I_Name,
   R_Party with Resp, Med with Medi

USE
return
* Program     : Do_Diag.prg
* Author      : Yong-Gi Kim
* Date        : April 8, 1987
* Calls       :
* Called by   : Diagnosis.prg
* File used   : Diagnosis.dbf
* Notes       : Display the patient's permanent history from
*              : Diagnosis file.

SET DELIMITER OFF
SET ECHO OFF
SET BELL OFF
SET TALK OFF

STORE 0 TO Number

ret = ' '  
Name = ' ' 

clear

? 1,16 say '---------------'
? 2,16 say 'PATIENT HISTORY'
? 3,16 say '---------------'
? 5, 1 say '--------------------------------------------------------'
? 7, 1 say 'Patient Number: ' GET Number
? 8, 1 say 'Patient Name  : ' 
? 9, 1 say 'Date of Birth  : ' 
?10, 1 say 'Sex          : '
?11, 1 say 'Doctor       : '
?12, 1 say 'Visit Date   : '
?13, 1 say 'Diagno. Code : '
?14, 1 say 'Diagnosis    : '
?15, 1 say 'Clinical Notes: '
?19, 1 say '--------------------------------------------------------'
READ

USE Patient_file
INDEX ON P_Number TO PTNT.NDX

SEEK Number

IF EOF()  
  ? 8,17 say '*** The Patient Number Does not exist in Patient_file ***'
ELSE  
  ? 8,17 say P_Name
  ? 9,17 say DOB
  ?10,17 say Sex
  ?11,17 say Doctor
ENDIF

USE
erase PTNT.NDX
USE Diagnosis_file
GOTO TOP
LOCATE ALL FOR (P_Number = Number)

IF EOF()
   318,17 say '*** The Number Does not exist in Diagnosis(History) file ***'
ENDIF

DO WHILE .NOT. EOF()
   312,17 say Vst_Date
   313,17 say D_Code
   314,17 say Diagnosis
   315,17 say C_Notes
   317,12 say '** Hit "Return" for Next Visit Date of History.**' GET ret
   READ
   CONTINUE
ENDDO

USE

320,15 say 'Hit "RETURN" to continue.' GET ret
READ
return
Sel = ' '

DO Dis_maintain with Sel

DO CASE

CASE Sel = 'U'
   DO Up_Hist

CASE Sel = 'P'
   DO Print_Hist

CASE Sel = 'S'
   DO Rep_Serv

CASE Sel = 'R'
   return

ENDCASE
parameter Sel

clear

2 5,32 say '---------------'
2 6,32 say 'MAINTAIN HISTORY'
2 7,32 say '---------------'
2 9,23 say '-----------------------------------------------'
211,26 say 'U => Update Diagnosis_file (< History >)'
213,26 say 'P => Print Updated History'
215,26 say 'S => Monthly Service Report'
217,26 say 'R => Return to Main menu'
219,23 say '------------------------------'

DO WHILE ((Sel<>'U') .AND. (Sel<>'P') .AND. (Sel<>'S') .AND. (Sel<>'R'))

    221,26 say 'Enter Selection : ' GET Sel
    READ

ENDDO

return
STORE 0 TO Number
Visit = '
Diag = '
Code = '
Notes = '
Ans = '
clear

@ 3,25 say '-------------'
@ 4,25 say 'Update History'
@ 5,25 say '-------------'

DO WHILE (Ans <> 'Y')

@ 7, 1 say '---------------------------------------------'
@ 9, 1 say '1.Patient Number : ' GET Number PICTURE '9999'
@10, 1 say '2.Last Visit Date: ' GET Visit PICTURE '99/99/99'
@11, 1 say '3.Diagnosis Code : ' GET Code
@12, 1 say '4.Diagnosis : ' GET Diag
@13, 1 say '5.Clinical Notes : ' GET Notes
@18, 1 say '---------------------------------------------'
@20,15 say 'Is This Information Correct?(Y/N): ' GET Ans
READ

ENDDO

STORE CTOD(Visit) TO Visit
USE Diagnosis_file
APPEND BLANK

REPLACE P_Number with Number, Vst_Date with Visit, Diagnosis with Diag,
D_Code with Code, C_Notes with Notes
USE
return
* Program : Print_Hist.prg
* Author : Yong-Gi Kim
* Date : April 12, 1987
* Called by : Maintain_History.prg
* File used : Patient_file.dbf, Diagnosis_file.dbf
* Notes : Print Updated history
*

Name = '',
Ans = '',
ret = '',
STORE 0 TO Number,
Birth = '',
Dr = '',

Do Get_Name with Name

USE Patient_file
GOTO TOP
LOCATE ALL FOR (P_Name = Name)

DO WHILE (((.NOT. EOF()) .AND. (Ans='Y'))

clear
@ 5,10 say '----------------------------------------'
@ 7,10 say 'Patient Name : '
@ 9,10 say 'Patient Number : '
@11,10 say 'Soc sec Number : '
@13,10 say 'Address : '
@15,10 say 'Date of Birth : '
@17,10 say 'Doctor : '
@19,10 say '----------------------------------------'
@ 7,28 say P_Name
@ 9,28 say P_Number
@11,28 say SS
@13,28 say Address
@15,28 say DOB
@17,28 say Doctor

@21,10 say 'Is this the information which you are looking for?(Y/N) : '
GET Ans
READ
IF Ans = 'Y'
    STORE P_Number TO Number
    STORE DTOC(DOB) TO Birth
    Dr = Doctor
ENDIF
CONTINUE
ENDDO
STORE CTOD(Birth) TO Birth

clear

    2 1,15 say '-------------------'
    2 2,15 say 'Updated History'
    2 3,15 say '-------------------'
    3 5,1 say '1. NAME : '
    3 8,1 say '2. NUMBER : '
    3 9,1 say '3. D.O.B. : '
    310,1 say '4. DOCTOR : '
    3 7,15 say Name
    3 8,15 say Number
    3 9,15 say Birth
    310,15 say Dr

USE Diagnosis_file
GOTO TOP
LOCATE ALL FOR (P_Number = Number)

IF EOF()
    315,1 say '** The Number does not exist in Diagnosis(History) file ***'
ENDIF

    311,1 say '5. VISIT DATE: '
    312,1 say '6. DIAG CODE: '
    313,1 say '7. DIAGNOSIS: '
    314,1 say '8. CLIN NOTE: '

DO WHILE .NOT. EOF()

    311,15 say Vst_Date
    312,15 say D_Code
    313,15 say Diagnosis
    314,15 say C_Notes
    316,10 say '** Hit "Return" for Next Visit Date: **' GET ret
    READ

CONTINUE
ENDDO

    319,1 say '--------------------------------------------------------'
    221,15 say 'Hit "Return" to continue.' GET ret
    READ

USE
return
STORE 0 TO Month, Year
STORE 5 TO count
STORE 1 TO num
STORE 0 TO Dia_Cnt
Cur_Diag = ''
ret = ''
clear
@ 8,15 say '------------------------'
@ 9,15 say 'Monthly Service Report'
@10,15 say '------------------------'
@12,15 say 'Enter the Month (99) : ' 
@12,39 GET Month PICTURE '99'
@14,15 say 'Enter the Year (9999) : ' 
@14,39 GET Year PICTURE '9999'
@16,15 say '*** PRINTER MUST BE TURNED ON *** '
READ

USE Diagnosis_file

clear

SET DEVICE TO PRINT

@ 1,15 say '*** MONTHLY SERVICE REPORT *** '
@ 1,50 say STR(Month,2,0)
@ 1,52 say '/'
@ 1,53 say STR(Year,4,0)
@ 3, 8 say 'DIAGNOSIS NUMBER '
@ 4, 8 say '-------------------------------------'

SORT TO ST ASCENDING ON Diagnosis FOR ((MONTH(Vst_Date) = Month) .AND.; (YEAR(Vst_Date) = Year))

USE ST
GOTO TOP
LOCATE ALL
Cur_Diag = Diagnosis

DO WHILE .NOT. EOF()

IF (Diagnosis <> Cur_Diag)
  @count, 0 say num
  @count,12 say Cur_Diag
  @count,50 say Dia_Cnt

STORE count + 1 TO count
STORE num + 1 TO num
STORE 0 TO Dia_Cnt
Cur_Diag = Diagnosis

ENDIF

STORE Dia_Cnt + 1 TO Dia_Cnt
CONTINUE
ENDDO

@count, 0 say num
@count,12 say Cur_Diag
@count,50 say Dia_Cnt
@count+1,8 say '-------------------------------------------------------'
@count+2,8 say ',

SET DEVICE TO SCREEN

USE
erase ST.dbf

return
Appendix M.

User Manual
This system is executed with dBASE III on DOS Version 3. In order to open this system, the command

```
do system
```

should be issued. This command calls SCREEN 1. The SCREEN 1 is the main menu for selecting the option which is

1. Appointment
2. Billing
3. Patient History
4. maintain History.

To exit this system, selection "E" is selected.

1. Appointment

Appointment deals with find, cancel, modify, schedule, and print appointment scheduling as shown on SCREEN 2.

- F=> Find Appointment
- C=> Cancel Appointment
- M=> Modify Appointment
- S=> Schedule Appointment
- P=> Print Appointment
- R=> Return to Main menu.
The option "F" calls for SCREEN 4 which accepts the patient's name. The exact name is input by the user to find the information which is already input for appointment. The information found is displayed in the form of SCREEN 3.

The option "C" calls SCREEN 4 which accepts the patient's name. The exact name should be input to find the information which is already input for appointment. The information found is displayed in the form of SCREEN 5. In order to cancel the appointment, the answer "Y" must be input for the question on the screen. This action deletes the appointment information from the system.

The option "M" calls SCREEN 4 which accepts the patient's name. The exact name should be input to find the information which is already input for appointment. The information found is displayed on the screen with a corresponding number. The cursor will be moved to the items of the number which will be modified. To finish modification, "R" must be input. This action will move the screen to main menu.

The option "S" accepts date from screen and displays all appointments of the date which is already input into the system. An answer "Y" must be input for the question "Available? (Y/N):" to call a screen which accepts the appointment information in the form of SCREEN 6, and any character except "Y" returns the screen to main menu.
The option "P" prints all the patient's appointment information, of a certain date, which is accepted from the screen in the order of the appointment time. However, before he or she can call information to the printer, by entering "P", she or he must first open the printer. This option calls a screen which accepts the appointment date in a form of Month/Date/Year, each of which is a 2 digit integer. Fig 5.1 is an example of a printed appointment schedule, the layout of this option.

The option "R" returns the screen to main menu.

2. Billing

The option "2" of main menu screen calls the Billing menu in the form of SCREEN 7 which consists of 9 options.

1=> Set up
2=> Enter Charge
3=> Enter Payment
4=> Print Bill
5=> Inquiry account status of a patient
6=> Print Insurance form
7=> An update of accounts.( Aging Charges ).
8=> Report of Monthly Account
9=> Return to main menu.
The option "1" calls SCREEN 4 which accepts a patient's name. The System displays the patient number, address, phone, date of birth, and doctor of the patient. The answer "Y" to the question "Is this patient Information Correct?(Y/N):" on the screen, returns the screen to main menu. Any input except "Y" for this question makes the system search for the next name like that entered until the correct patient's data appears on the screen. The purpose of this option is to confirm the patient number, since future entries will be maintained by only this number.

The option "2" displays SCREEN 9. This option is called to enter the patient's service charges. A template appears to enter such data as patient name, date, service, and amount of service charges. In case of common service, the input of the service code will find the fee of the service, otherwise, the cursor will move to entry of charge. After entry, the phrase "More Charge to input for this Patient?(Y/N):" on the screen. If there is more charge to input for the patient, the cursor returns to the date. After which, a new date of screen and a second service charge is entered per screen direction, and so on until all charges are inside the system. Any other input except "Y" will display a question "More Patients?(Y/N):". If there are any more patients to input charge for, the answer "Y" must be selected. This will repeat the process described above. Any other input except "Y" will return the screen to main menu.
The option "3" displays SCREEN 10. This option is selected to enter a patient's payment(s). When the patient number, date, payment type, and the amount of payment are input, the answer "Y" must be input for the question "More Payment to input for this patient?(Y/N):" if there is more payment to input for the patient. The payment type, "H", "A", or "R" should be selected depending on the payment type; "H" is for check, "A" is for cash, and "R" is for credit card. Any other input except "Y" will display the question "More patients?(Y/N):". If there are any more patients to input payment for, the answer "Y" should be selected. This will repeat the process described above. Any other input except "Y" will return the screen to main menu.

The option "4" is for bill. A set of Month, Date, Year is accepted from the screen, then the answer "Y" should be input for the question "Is this correct Bill Date?(Y/N):" which will appear on the screen. The system will print each patient's charge and payment of the month accepted from screen, accumulative balance so far and display the aged charge on the bottom of the bill. Printer must be turned on before this option is selected. Fig. 5.2 is an example of bill, layout of this option.

The option "5" is for bill inquiry from patients. This option calls SCREEN 4 which accepts a patient's name. The system displays the patient's information on the screen. Character "Y" must be input if the information matches the patient in question. Any other input except "Y" displays the next same patient name. The output is displayed on the screen.
in the form of SCREEN 11. Any key can be hit in order to return to main menu.

The option "6" is for insurance form printing. A patient number, date of service, place of service, procedure code, and laboratory charge outside the office will be input for printing a insurance form. Fig.5.6 is an example of printed insurance form.

The option "7" is for aging charges for all patients. The option "7" calls a screen which accepts current month, date, and year. Current month and date are 2 digits entry and the current year is a 4 digits. To print the Aging Report, "Y" should be input for the question "Is This Correct Date?(Y/N): " on the screen. Before printing, printer must be turned on. Fig.5.3 is an example of Monthly Aging Report, the layout of this option.

The option "8" is for Monthly Accounting Report which prints status of all open accounts, the sum of total charges, total payments, and the current balance of all patients of the accepted month. The option "8" calls a screen which accepts current month, and current year. Current month is a 2 digit entry, and the current year is 4 digits. Before printing, printer must be turned on. Fig.5.4 is an example of the Monthly Accounting Report lay out of this option.

The option "R" returns the screen to main menu.
3. Patient History

The option "3" of main menu screen calls the Diagnosis menu in the form of SCREEN 15 which consists of 3 options.

I => Input Patient Information
D => Search Patient History
R => Return to Main menu.

The option "I" calls SCREEN 8 which accepts all information of a patient. The answer "Y" should be input to enter the information into system.

The option "D" accepts a patient number, then the system displays the patient's information in the form of SCREEN 12. In order to find the history of the next visit date of the patient, "Return" key should be hit. This option displays all history of a patient.

The option "R" returns the screen to main menu.

4. Maintaining History

The option "4" of main menu screen calls the Maintain History menu in the form of SCREEN 13.
U→ Update History
P→ Print Updated History
S→ Report Service Type
R→ Return to Main menu.

The option "U" calls SCREEN 14 which accepts patient number, visit date, diagnosis code, diagnosis, and clinical notes. The answer "Y" should be input to add the new diagnosis result into the system.

The option "P" calls SCREEN 4 which accepts a patient's name. This option displays the patient number, social security number, address, date of birth, and doctor of the patient name. If the information is that which is being looked for, then the answer "Y" must be input to get the complete history of the patient. When "Y" is input, the system finds the history of the patient, and displays this in the form of SCREEN 14.

The option "S" is for Monthly Service Report. This option calls a screen which accepts the month and year. The month is a 2 digit number, and the year is a 4 digit number. Before printing, printer must be turned on. Fig. 5.5 is an example of the Monthly Service Report layout of this option.

The option "R" returns the screen to main menu.
### *** APPOINTMENT ***

**DATE:** 04/21/87

<table>
<thead>
<tr>
<th>No</th>
<th>Patient Name</th>
<th>Time</th>
<th>Doctor</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>kkkkk</td>
<td>8:00</td>
<td>jjj</td>
<td>ppp</td>
</tr>
<tr>
<td>2</td>
<td>Yong-Gi Kim</td>
<td>8:45</td>
<td>Park</td>
<td>aaa</td>
</tr>
<tr>
<td>3</td>
<td>cccccc</td>
<td>9:30</td>
<td>dddd</td>
<td>dddddd</td>
</tr>
<tr>
<td>4</td>
<td>test2</td>
<td>11:30</td>
<td>ddf</td>
<td>ffffff</td>
</tr>
<tr>
<td>5</td>
<td>Han</td>
<td>12:00</td>
<td>fff</td>
<td>dd</td>
</tr>
<tr>
<td>6</td>
<td>Lee</td>
<td>12:15</td>
<td>hhh</td>
<td>yyy</td>
</tr>
<tr>
<td>7</td>
<td>Timothy Park</td>
<td>12:30</td>
<td>fff</td>
<td>sss</td>
</tr>
<tr>
<td>8</td>
<td>hhhh</td>
<td>12:45</td>
<td>gg</td>
<td>gg</td>
</tr>
<tr>
<td>9</td>
<td>ffffffff</td>
<td>1:00</td>
<td>jjkjk</td>
<td>jhkjkj</td>
</tr>
<tr>
<td>10</td>
<td>hhhhhhhhhh</td>
<td>2:30</td>
<td>sssss</td>
<td>ssssss</td>
</tr>
<tr>
<td>11</td>
<td>a</td>
<td>3:30</td>
<td>wwww</td>
<td>wwww</td>
</tr>
<tr>
<td>12</td>
<td>Lim</td>
<td>3:45</td>
<td>ff</td>
<td>qqq</td>
</tr>
<tr>
<td>13</td>
<td>bbbbbbb</td>
<td>4:15</td>
<td>aaaaa</td>
<td>aaaaaa</td>
</tr>
<tr>
<td>14</td>
<td>test</td>
<td>4:30</td>
<td>wwww</td>
<td>wwww</td>
</tr>
<tr>
<td>15</td>
<td>test3</td>
<td>5:13</td>
<td>sd</td>
<td>sdf</td>
</tr>
<tr>
<td>16</td>
<td>wuyqiwoye</td>
<td>6:45</td>
<td>sdds</td>
<td>dfdsds</td>
</tr>
</tbody>
</table>

---

**Fig. 3.1** Printed Appointment Sheet
<table>
<thead>
<tr>
<th>Date</th>
<th>Amount</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>05/30/87</td>
<td>123.00</td>
<td>ddd</td>
</tr>
<tr>
<td>05/23/87</td>
<td>333.00</td>
<td>ccc</td>
</tr>
<tr>
<td>05/11/87</td>
<td>333.00</td>
<td>c</td>
</tr>
</tbody>
</table>

**PAYMENT**

<table>
<thead>
<tr>
<th>Date</th>
<th>Amount</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>05/11/87</td>
<td>111.00</td>
<td>R</td>
</tr>
<tr>
<td>05/21/87</td>
<td>100.00</td>
<td>R</td>
</tr>
</tbody>
</table>

**ACUMMULATIVE BALANCE**

<table>
<thead>
<tr>
<th></th>
<th>&gt;30days</th>
<th>&gt;60days</th>
<th>&gt;90days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>789.00</td>
<td>1653.48</td>
<td>1807.22</td>
</tr>
</tbody>
</table>
### Monthly Aged Accounting

**5/1987**

| Pat.
| Number | Curr  | >30days | >60days | 90days |
|--------|-------|--------|---------|---------|--------|
| 1      | -1190.88 | 0.00   | 0.00    | 0.00    |        |
| 2      | 1320.48  | 129.22 | 0.00    | 0.00    |        |
| 3      | 0.00     | 2664.44| 0.00    | 0.00    |        |
| 4      | 0.00     | 87.00  | 0.00    | 0.00    |        |
| 5      | 0.00     | 44.55  | 0.00    | 0.00    |        |
| 7      | -172.35  | 0.00   | 0.00    | 0.00    |        |

<table>
<thead>
<tr>
<th>TOT Curr</th>
<th>TOT_30</th>
<th>TOT_60</th>
<th>TOT_90</th>
</tr>
</thead>
<tbody>
<tr>
<td>-42.75</td>
<td>2925.21</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Total Charge: 2882.46

---

**Fig.5.3** Monthly Aged Accounting
### Monthly Accounting Report

**5/1987**

<table>
<thead>
<tr>
<th>Patient Number</th>
<th>Charge</th>
<th>Payment</th>
<th>Balance</th>
<th>Charge Count</th>
<th>Payment Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>300.00</td>
<td>1411.00</td>
<td>-1111.00</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>187.00</td>
<td>100.00</td>
<td>87.00</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>200.00</td>
<td>-200.00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>120.23</td>
<td>0.00</td>
<td>120.23</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0.00</td>
<td>500.00</td>
<td>-500.00</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>27.24</td>
<td>12.00</td>
<td>15.24</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>17.00</td>
<td>0.00</td>
<td>17.00</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Total:** 651.47  2223.00  -1571.53  9  8

---

Fig. 5.4 Monthly Accounting Report
### MONTHLY SERVICE REPORT

<table>
<thead>
<tr>
<th>DIAGNOSIS</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 a</td>
<td>2</td>
</tr>
<tr>
<td>2 d</td>
<td>2</td>
</tr>
<tr>
<td>3 s</td>
<td>1</td>
</tr>
<tr>
<td>4 ssssss</td>
<td>1</td>
</tr>
<tr>
<td>5 v</td>
<td>3</td>
</tr>
<tr>
<td>6 w</td>
<td>1</td>
</tr>
<tr>
<td>7 xx</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 5.5 Monthly Service Report
HEALTH INSURANCE CLAIM FORM

*** PATIENT AND INSURED INFORMATION ***
1. PATIENT NAME: Yong-Gi Kim
2. DATE OF BIRTH: 09/03/54
3. INSURED NAME: Yong-Gi Kim
4. PATIENT ADDRESS: 1000 Yreka ct. Missoula, MT
5. PATIENT SEX: M
6. INSURED ADDRESS: Yong-Gi Kim, 1000 Yreka Court, MSLA, MT

*** PHYSICIAN OR SUPPLIER INFORMATION ***
7. LAB. CHARGE OUTSIDE OFFICE: 23.00
8. PHYSICIAN NAME: Park
9. SERVICE DATE: 05/15/87
10. SERVICE PLACE: H
11. PROCEDURE CODE: 1
12. SERVICE DESC.: dddd
13. DIAGNOSIS CODE: d
14. SERVICE: aaaaaa
15. CHARGE: 100.00

Fig. 5.b Insurance Form