To the Graduate Council:

I am submitting herewith a thesis written by Ann Pearson Rule entitled "Computer Implementation of a Medical History." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Sara R. Jordan

Sara R. Jordan, Major Professor

We have read this thesis and recommend its acceptance:

Michael G. Thomason

Accepted for the Council:

Stilton A. Smith

Vice Chancellor
Graduate Studies and Research

ℓ

COMPUTER IMPLEMENTATION OF A MEDICAL HISTORY

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee

Ann Pearson Rule

August 1974

ACKNOWLEDGMENT

The author is indebted to her advisor, Dr. Sara R. Jordan, for her help and encouragement in the development of this thesis. Appreciation is also expressed to Dr. Arthur L. Kretchmar for offering many helpful suggestions concerning the programming for this project. Also, the author wishes to express her appreciation to Dr. Michael G. Thomason for his services on the Master's Degree committee.

1165555

ABSTRACT

This thesis gives a general overview of information processing in medicine. The development of medical information systems by the modular approach and the total approach is discussed. The various features of MUMPS, the high-level programming language which was used in the software development for this thesis, are described.

A survey of previous automated and non-automated medical histories is given, followed by a more detailed discussion of a driver program, which was used to generate the automated history for this thesis.

Finally, a software package is discussed, including not only the basic interview programs, but also a response correction program, a data base integrity check program, and a data search program.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION	1
	Computers in Medicine	1
	Modular Approach Versus Total Approach	2
II.	MUMPS	7
III.	AUTOMATED MEDICAL HISTORIES	15
	Background	15
	Driver for Automated Medical History	22
	Implementation of BDC Questionnaire	26
IV.	CONCLUSIONS AND RECOMMENDATIONS	61
LIST OF REFERENCES		64
APPENDIXES		66
A.	QUESTIONS PERTAINING TO THE MOTHER OF THE INFANT	67
B.	LISTING OF COMPUTER PROGRAMS	72
VITA		85

LIST OF FIGURES

FIGURE		PAGE
1.	Core Memory Allocation of the MUMPS System	9
2.	Example of MUMPS Coding	10
3.	A Simplified Tree-Structured Patient File	13
4.	Free Text Answer Type	24
5.	Yes/No Answer Type	24
6.	Single Choice from List Type	25
7.	Operation Sequence in ASK Mode	27
8.	BDA - Driver	30
9.	BDB - New Patient Data	31
10.	ICK - Input Patient ID	32
11.	↑BDC Global Array	34
12.	HIN - History Interview	35
13.	BDS - Patient Data Summary	38
14.	HSM - Summary; Integrity Check	39
15.	IDB - Integrity of Data Base	43
16.	ICC - Integrity Case by Case	44
17.	IEF - Integrity Entire File	46
18.	CPF - Correct Patient File	48
19.	LST - List Patients	52
20.	MDP - Move and Delete Patient	54
21.	DTS - Data Search Setup	56
22.	SEA - Data Search	58

CHAPTER I

INTRODUCTION

I. COMPUTERS IN MEDICINE

Compared with other advances in patient care, little progress has been made in information processing, which is an important area of medical practice. Data processing in many fields has been changed drastically by the computer during the last few decades. However, computerized information processing is still relatively young in medical care, especially in the hospital. The majority of the information used by hospitals is still hand-written and delivered to different areas in the hospital by multiple slips of paper. Retrival of information usually takes the form of hand-shuffling through the medical record, which is usually a disorganized collection of doctors' and nurses' reports, test results, and other records. Recently a study of hospital costs showed that over 20 percent of the hospital budget is spent on information processing (1, p. 1321).

The idea that computer technology would improve the operations of a typical hospital is one that few people would disagree with. However, early interest in using computers in medicine was troubled with the usual overenthusiasm and unrealistic expectations. The computer was expected to provide an accurate collection and retrieval of all clinical information, do automatic diagnosis, collect and analyze

a variety of physiologic signals, perform and interpret laboratory tests, and in general, perform many wonderful services for patients. However, the application of computer technology to medicine suffered numerous failures, which were certainly not unique to medicine. Due to the conflicting objectives which are inherent in the medical environment, the idea of a computer-based medical information system is one that is not easily agreed upon. Other problems concern the definition of the task to be done and the development and implementation of the computer system.

A medical information system means different things to different members of the hospital staff. Clearly, the physician, the nurse, the administrator, the research investigator are all looking for something different from such a system. For example, the administrator would want to use the system for patient billing and management decisions, and the research investigator would be interested in data concerning diseases, treatment, resources utilized, and the outcome of therapy.

There are two different approaches used in the development of medical information systems. These two very different philosophies come from the divergence of priorities between the professional staff and management (2, p. 244). These are simply two groups with different backgrounds and different expectations.

II. MODULAR APPROACH VERSUS TOTAL APPROACH

The total systems approach takes into account all information

processing tasks. All aspects are subjected to a systematic and detailed study. The total data flow is considered, and all the hospital's functions are considered at one time. A powerful computer system and a powerful data management programming system would be needed in order to develop a system using the total approach. The modular approach, as the name suggests, is interested not in the whole hospital picture, but in only one part of the picture at a time. The modular approach works in the following way. First, explicit objectives, procedures, and standards are defined for some part of the hospital, like a laboratory or the pharmacy. A computer system suitable for this part is gracefully phased in. Other small systems are developed in other parts of the hospital, and eventually several smaller modular systems will be linked into one total system.

There are several problems to be considered when using the total system approach. First, medical information processing is very diverse in nature and is not very well defined. There are many different people and hospital units involved, and it is very difficult to come to a general agreement about how the data is to be handled. However, organizational agreement and cooperation are much easier to obtain at the department level. A second problem is that computer technology and programming languages are weak when considered in the total approach. The equipment requirements of the modular approach are more easily satisfied with available computer technology than that of the total systems approach. The minicomputer is the most likely choice for modular systems. The third problem to be considered here is the one of installing a total system. The hospital, unlike an assembly line,

cannot be shut down until a large computer system is installed and working. Again the modular approach has an answer. It is easier and much more realistic to phase in small systems rather than some total system that may disrupt the entire hospital.

There are other advantages of the modular system approach. Initial start-up costs are smaller. It is simpler to determine cost/benefit analysis, since each computer is directly linked to one operational unit. The use of small computers makes it easier to duplicate computer equipment. The hospital does not need to invest in highly specialized personnel to develop an all-embracing computer system. Also, once one department has demonstrated success with its computer system, it will be much easier to convince others of its usefulness. A very important advantage of the modular approach is its relative flexibility. In any computer system, as the system is developed, the user is better able to understand the way things should have been done and will naturally want to make changes. Also, the user will discover new tasks for the system to perform. The nature of medical care changes rapidly, making the need for a flexible information system very pertinent. Finally, once modular systems have been developed and are in use, other medical care facilities may choose those functions that will be useful for their own needs.

There are three areas of weaknesses concerning the modular system approach. The use of several small computer systems leads to redundancy in hardware and extra interface equipment, since a single department might need access to several different systems. Also, each system needs its own operating system, whereas a large computer has only one

operating system and usually a more powerful instruction set. Another problem area concerning the modular system approach is that it tends to cause a fragmentation of management. In the total approach, the plan is for overall system management, but with the modular approach, some departments may tend to be unaware of others' needs and may give more priority to their own needs. The final problem is that the modular approach does not help the development of a computer-based integrated medical record. The modular approach places an emphasis on data processing activities of individual hospital departments. Most of the data that are collected and used by a single department are of interest to other departments and would contribute to an overall medical record. However, a scheme which uses several separate computers makes it very difficult for different modules to exchange data.

Two general requirements are needed to allow a modular system to work effectively. First, the programming language must permit different modules to function within the same system. Secondly, the file structure must permit the form of data storage to be independent of the data source.

Such a modular system has been developed. The Laboratory of Computer Science at the Massachusetts General Hospital has been working on such a medical information system for over a decade. In 1966 when it was decided to use the modular system approach rather than the total system approach, a new high-level programming language and computer system was developed. The language, MUMPS, was successful in eliminating the problem concerning file structure and the problem

concerning different modules performing in one software system. The language and the computer system are still in use at the Massachusetts General Hospital and also are widely used throughout the country, including at the University of Tennessee Hospital. The programming work for this thesis was done at the University Hospital using the MUMPS language. The primary work was the creation of an automated medical history for the Birth Defects Clinic at the Hospital. One of several questionnaires used by the clinic was chosen to be automated. The questionnaire, "Questions Pertaining to the Mother of the Infant," was automated using a "driver" program, which is discussed in Chapter III. Once the basic questionnaire was implemented, a package of programs was developed to perform such tasks as correcting previous responses to the questionnaire, performing an integrity check on the stored responses, and doing a data search on the entire Birth Defects Clinic patient file. These tasks and others are discussed in Chapter III. But before getting into the discussion of the implementation of the software and the details of automated medical histories, the MUMPS language will be discussed in the next chapter.

CHAPTER II

MUMPS

MUMPS (Massachusetts General Hospital Utility Multi-Programming System) is a time-sharing system designed to be used on a medium scale computer. It is currently implemented on a PDP-15 (Digital Equipment Corporation) computer at the University of Tennessee Hospital. The computer has 16,000 eighteen-bit words and is interfaced to a fixed head disk, DEC tape, and a variety of remote terminal devices, such as teletypes, CRT terminals, and DEC writer.

All application programs are written in MUMPS, a near-English, user-oriented, interpretive programming language (6, p. 474). MUMPS is patterned after JOSS, another high-level, interpretive language developed in 1964 at the RAND Corporation. MUMPS has also been influenced by TELCOMP, STRINGCOMP, and FILECOMP. A programmer is permitted by MUMPS to write a program, debug it, edit it, run it, and modify it during an interactive session at a console. The interpreter is re-entrant and is part of the executive system. All active users are assigned partitions of core memory. Before a program can be used, an available partition must be found. The desired program is brought into that partition from disk. As long as the program is active, it remains in its partition. As a result of this partitioning, the time-sharing monitor can change from user to user without swapping a program in from a disk or drum. The user's program may also call or

overlay another program, causing the present program in the partition to be replaced. Core and disk storage allocations are shown in Figure 1.

MUMPS programs are divided into parts which are indicated by an integer part number. Each part is further divided into steps, which are identified by a step number consisting of a decimal fraction appended to the part number. A step may consist of a single instruction or a series of commands executed one after another. Instructions may be stated in a long mnemonic form or a compact form where only the first letter of the command is used. A statement preceded by a step number is considered to be an indirect statement and is stored to be executed later as part of a program. A statement without a step number is a direct statement and is executed immediately after it is entered from a terminal.

Figure 2 shows some examples of MUMPS statements. The lefthand side shows the statements using the long mnemonic form and the righthand side shows the statements using only the first letter of the commands. The first two statements are direct statements which merely set a variable A equal to 23 and then print out the value of that variable. Remembering that programs are divided into numbered parts, the third statement is another direct statement which causes part 2 to be printed out. Part 2 consists of the same commands to print out the contents of A except that the statements are indirect rather than direct. The final direct statement causes part 2 to be executed, again printing out the value of the variable A.

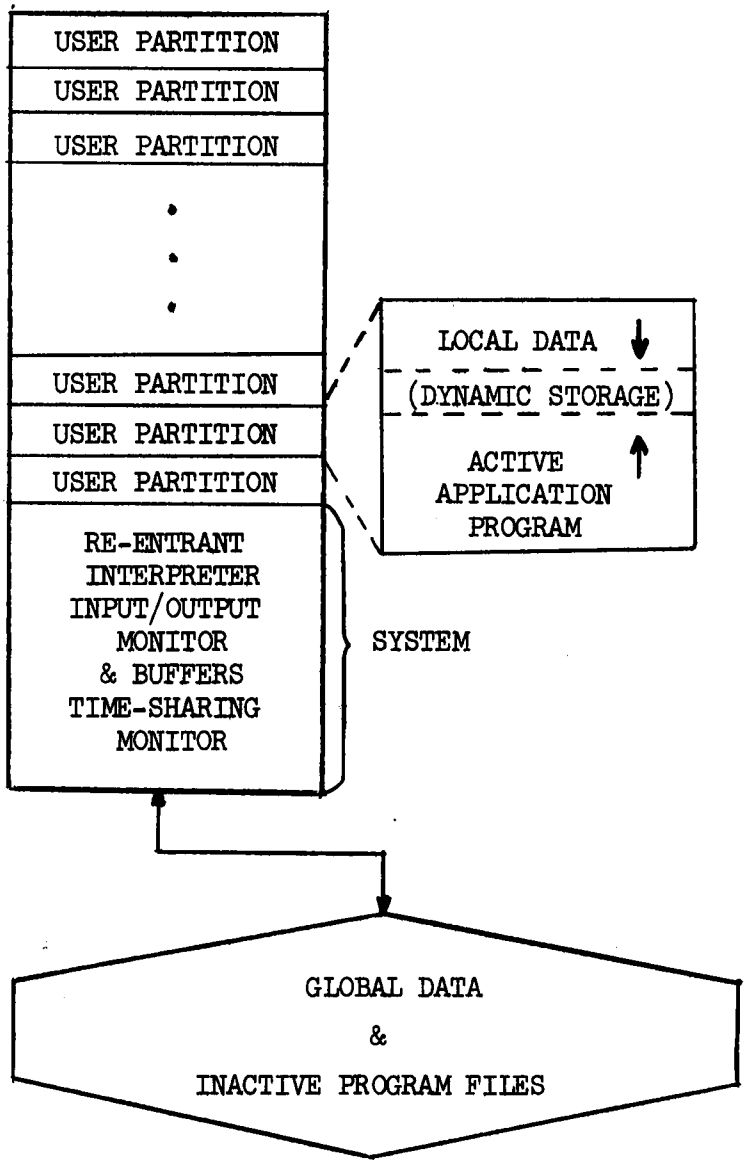


Figure 1. Core memory allocation of the MUMPS system.

```
>SET A=23                    >S A=23
>TYPE A                      >TYPE A
23                           23
>WRITE 2                     >W 2

2.20 SET A=23                2.20 S A=23
2.30 TYPE A                  2.30 T A

>DO 2                        >D 2
23                           23
```

Figure 2. Example of MUMPS coding.

Much of the application of information processing in medicine involves the capture, analysis, storage, and retrieval of text string information. The MUMPS language has been designed to meet this need. In addition to algebraic and boolean processing, MUMPS has a number of high-level operators, which make string extraction, location of sub-strings, comparisons, and checking of syntax and form of information relatively easy. For example, the EXTRACT function will extract all characters from a string that are between two given numerical values, inclusive. The following MUMPS command will set the variable C equal to the string value "ABC".

```
SET C=$EXTRACT("ABCDEFG",1,3)
```

Another feature of the MUMPS language which is useful is its input/output scheme. MUMPS programs can be written independently without any particular device in mind. Then at execution, any device can be used by assigning a device number to a special system variable which indicates the device to be used. Formatting and control of

position on a page are done by using special format characters and variables. All of this means, for example, that the same program can be used to generate a report on the CRT scope and to also print out the report on a printer.

MUMPS allows application programs to create and use their own local data which is stored in the program's partition and also to use global data, which is shared by other programs in the system and is stored on disk. Local data functions as scratch or transient data. Only elements for which data are actually defined in a local array are allocated space. A symbolic local variable may contain either a numerical value or a variable length string value. When it has a string value, only the space that is actually required is allocated. The scheme used for local data is also used for global data. Data stored as global data on the disk are referenced symbolically. The file is similar to that of a local datum except that the name is preceded by the character up-arrow. Fields in a data file are treated as elements of an array and are referenced by means of subscripts. Subfields in a data file are referenced by appending additional subscripts. So the globals are just a set of arrays stored on disk which function as a common data base for all the users.

Since any user may use an unprotected global, there are means to prevent two or more users from simultaneously trying to change the same globals. A global may be referenced at any time to do anything except alter it. When a user wants to change a global, an OPEN command must be executed. This command prevents any other program from altering

data in the global specified as the argument of the OPEN command. This action is terminated whenever a CLOSE command is executed or the program ends. Also since different programs may modify the content and structure of a global, MUMPS provides a set of functions which allow the programmer to determine the type and structure of a global array. In general, these functions permit the programmer to locate nodes or elements where information is stored, to determine if a particular node is defined, and to determine what type of data is stored in a particular node.

The basic structure of global data is an N-dimensional subscripted, dynamic, tree-structured array. Any node in a global array may be a numeric value, a string value or a pointer to a lower level of the tree. Data can be stored at any level of the tree, and there are no limitations on the size of the tree. The structure of the tree is not fixed but dynamic since nodes can be added or deleted at will.

Figure 3 shows a global tree-structure that could be used for a simplified patient file. The first level of the array identifies a particular patient. The second level represents a series of information. If any extra information is to be stored for a particular item, it would be on the third level or lower. In Figure 3 for example, at the third level under X-rays, we find more detailed information concerning x-rays occurring on different dates.

Storage and retrieval of data from a global array is just like that of local variables. For example, if we were working with the

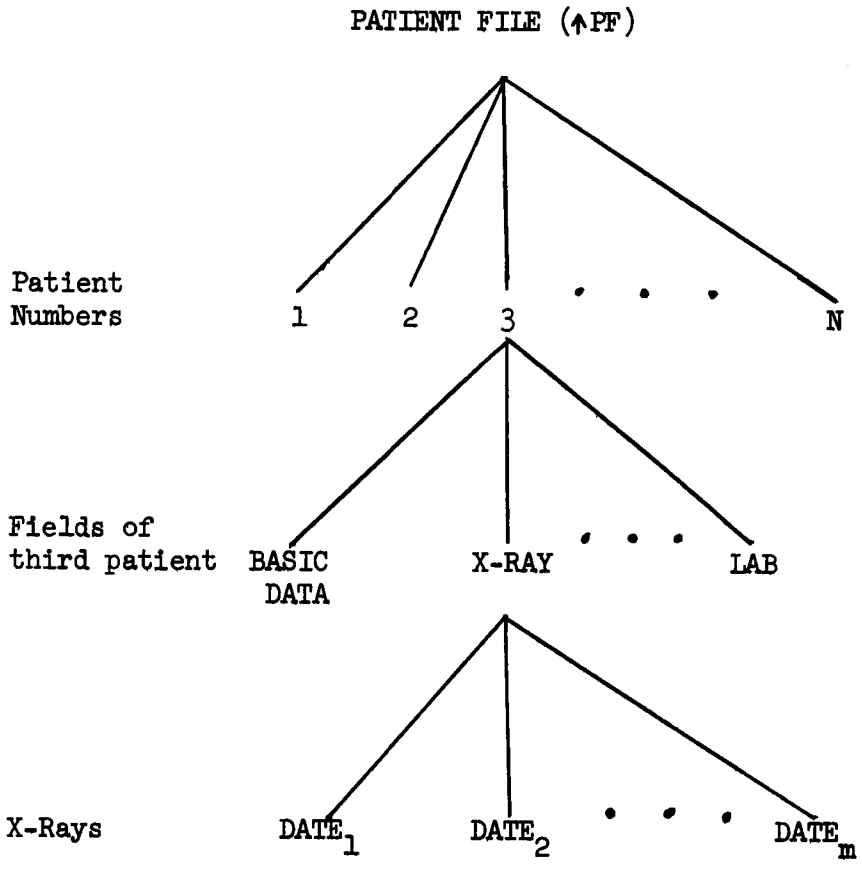


Figure 3. A simplified tree-structured patient file.

global shown in Figure 3 and wanted to set the first x-ray data for patient number 3 equal to the string value "5/6/74", we could use the following statement:

```
SET ↑PF(3,2,1)="5/6/74"
```

Retrieval is similar. Suppose we wanted to print out the same date node. The following statement would do it:

```
TYPE ↑PF(3,2,1)
```

To summarize the power of the MUMPS language, the major advantages are the ease of programming, the ease of debugging, the compact code, the elaborate string processing capability, the input/output flexibility and device interchangability, and perhaps most importantly, the powerful filing system on secondary storage. It is the only system that supports such extensive file manipulation, string handling, and input/output flexibility on a medium scale computer (5, p. 305).

CHAPTER III

AUTOMATED MEDICAL HISTORIES

I. BACKGROUND

Data gathered from the medical history forms the primary document for review of clinical information for patient care, clinical investigation and auditing of medical-care process. Despite increasing demands in this area, there has been relatively little research done. The traditional method of taking a medical history consumes a large amount of the physician's time, and usually the history is inadequate due to a time limitation. Recording a history longhand is not only a long laborious process, but also results in non-standardized and sometimes illegible histories. Furthermore, information obtained from a history is arranged chronologically rather than according to diagnostic or therapeutic problems.

Medical researchers have not been totally oblivious to the need for improved medical histories. There has been work done in the field, both with and without the aid of computers. The recent work usually does concern itself with automated histories for the following principal reasons (1, p. 1324). The first reason is to speed up the history-taking process and increase the efficiency in the delivery of patient care. As mentioned above, history taking is time consuming and usually produces incomplete and illegible records. Automated techniques could

not only decrease the time needed and produce typed reports, but could also give the physician a useful tool for further study and development. The second advantage of using a computer in taking a medical history involves its use as a screening device to detect disease. Another reason is that it would be possible with the use of computers to standardize procedures, terminology, and coded information.

The first attempts at standardized histories were self-administered patient questionnaires, usually consisting of about 200 questions. A number of self-administered questionnaires have been developed, many by physicians. The first major self-administered questionnaire was the Cornell Medical Index (CMI) (4, p. 530). The CMI was constructed around 1949 to obtain important facts about a patient's medical history for comprehensive diagnostic and prognostic appraisals. The CMI was a form on which patients indicated their symptoms and other significant medical information before being interviewed. Then when the interview occurred, the physician had information on which to base tentative diagnostic and prognostic appraisals. The CMI was a four-page sheet which collected data by asking questions. It consisted of 195 questions which were chosen to correspond to those usually asked in a comprehensive medical interview. Questions were asked using informal language, avoiding technical terms. The CMI was administered simply by handing the form to the patient and asking that it be completed. The CMI was used at the New York Hospital Out-Patient Department and it was determined that the data collected by the CMI was as accurate as that obtained in oral interviews. Although the

CMI was not automated, it was still useful since it did save the physician's time; it gave the physician a preliminary survey of the patient's medical problems; and its systematic form was easier to review.

The Cornell Medical Index-Health Questionnaire and the questionnaire portion of the Multiphasic Health Checkup of the Permanente Medical Group were two of the earlier self-administered histories to use a digital computer. The histories were taken in a form which permitted computer processing. Both questionnaires consisted of questions to which a patient answered "Yes" or "No." Responses to the CMI were keypunched onto cards for later computer processing. The Permanente group used prepunched cards, each of which had an individual question on it. The patient simply sorted the cards into a "yes" stack and a "no" stack. Later a print-out was gotten of all the questions to which the patient responded positively.

The Cornell and Permanente system did save the physician's time, yet the set of questions was fixed. For example, if a patient indicated that he had no heart trouble, he would still be asked further questions concerning the heart. In an oral interview with a physician, however, questions which are not pertinent are not asked. In other words, the questions asked depend on the patient's earlier responses. This method allows a history to be taken in great detail. One of the first computer-based self-administered histories which had branching techniques was a system developed in 1966 by Warner Slack and his associates at the University of Wisconsin Hospital (9, p. 194). A

small, high-speed digital computer, the LINC (Laboratory INstrument Computer) was used. The LINC was developed and constructed in 1962 at the Massachusetts Institute of Technology. The rest of the system included a typewriter keyboard, two magnetic-tape units, and a cathode-ray oscilloscope. This computer-based questionnaire was designed to imitate the physician as closely as possible. When a patient responded negatively, further questions concerning the topic of the original question were omitted. The patient could also choose as a possible response, "Don't Know" or "Don't Understand". The choice of questions presented varied depending on the patient's responses. During the actual history taking, the questions along with the list of the four possible responses appeared on the cathode-ray screen. The patient answered the questions by pressing one of four numbered keys. The patient was allowed to correct an erroneous answer or change his mind. When the patient was satisfied, he pressed a bar which entered the response into the computer. When each history had been completed, all print-out statements associated with the patient's responses were read into the computer from magnetic tape and printed by the teletype for a hard copy. As an initial study to test this system, the program was constructed to obtain data concerning symptoms of allergy. In the first study done of 50 patients, it was concluded that the computer print-out was superior to the past, hand-written histories but less detailed when dealing with the present illness of asthma. In addition, patient reaction to the interview was favorable.

The next system to be described is a visual-display technique in which the physician can enter the history and physical findings into a computer-based medical record (8, p. 571). This system was developed at the Mayo Clinic by a group headed by J. G. Mayne in the middle sixties. The system uses an IBM 7040, a graphic display terminal with a light-pen, keyboard, and printer. The display terminal projects images which are referred to as matrices since the information is arrayed in horizontal rows and vertical columns. In operation, the display screen initially displays general categories of information. A master guide which includes items like personal history, family history, present illness, and physical examination findings begins the interview. As the physician makes a selection, succeeding displays or matrices narrow the scope from general to specific information. The matrices were designed to represent the hierarchy which is familiar to physicians. At all times, the physician chooses his entry by passive recognition of data shown on the screen rather than by active recall. The physician may also get a typewritten copy of any matrix. In initial testing and study, encouraging acceptance from physicians was found. In general, this technique has resulted in clinical records which are uniformly formatted, complete and legible, and easily retrieved.

In 1968 Irving F. Kanner started using a pencil and paper questionnaire which had been developed by Mayne before the computerized questionnaire (7, p. 317). Kanner actually used this questionnaire for a while before he purchased a programmed typewriter, which had 100 feet

of magnetic tape memory. Kanner programmed onto the tape all the possible positive answers to the current questionnaire and also devised a way to search the tape. Now instead of hand typing a summary, a secretary simply notes a positive response and directs the tape to the correct position, and the typewriter automatically types out the response. This Programmed Medical History (PMH), as Kanner called it, consisted of 48 pages, having a maximum of 935 questions. A branching technique was used, with detailed questions skipped if the response is "No". The PMH was a self-administered history usually taking 40 minutes to complete. Kanner used the PMH similarly to the way the Cornell Medical Index was used. The PMH was filled out before the physician actually saw the patient. Kanner felt that the present illness of a patient was always brought out by the PMH and that the PMH was more adequate than the traditional methods. Also the PMH print-out was much easier to read than the usual handwriting. The PMH saved between 20 to 25 minutes in the physical examination. Even though not computerized, Kanner argues that the PMH would supply an easily and quickly read history that would be useful in patient care.

The computerized history developed at the Massachusetts General Hospital in 1968 was patterned after a typical oral history given by a physician (11, p. 39). It incorporates the normal branching and also includes additional questions which are more detailed to compensate for the loss of non-verbal input. The history consists of a dialogue between the patient and a teletypewriter terminal. The teletype prints out each question and a list of possible responses. The

patient needs only to type in the number corresponding to the appropriate response. Each question set is tailored to each patient in that subsequent questions depend on the patient's previous response. The interview usually takes around 45 minutes to complete. At its conclusion, a printed summary of the data can be printed out.

This system uses a PDP (Digital Equipment Corporation) computer, with teletypewriters, two DEC tape transports, and a fixed head disk. It also uses the MUMPS language and a special questionnaire driver which will be discussed in the next section of this chapter. All the question text and branching information are stored on the fixed head disk. This system was first used in the Out Patient Department of the Massachusetts General Hospital. Patients' attitudes ranged from tolerant to enthusiastic, and the quality of the collected data was excellent. In general, physicians accepted this system as a helpful adjunct to patient care.

The final automated medical history to be discussed is one developed at the University of Illinois Hospital in the early seventies (3, p. 99). The system's hardware consists of a 32K IBM 1800. The software consists of an interpreter written in FORTRAN which interprets SMILE (Simple Medical Interviewing Language), a composer-editor for writing and correcting programs written in SMILE, and utility programs for listing programs and saving patient data on tape. At present, SMILE has nine instructions which use two data sets stored on disk. The first set of data is a list of responses made by a patient. The second set is the text set which makes up the questions. The system

has been used at the University of Illinois Hospital, with positive results. The language used was easily learned by physicians and other users. Also, the system allowed the development of new history-taking programs without the need for continued aid from a computer specialist.

II. DRIVER FOR AUTOMATED MEDICAL HISTORY

Part of the work for this thesis concerned creating an automated medical history (AMH) from a questionnaire used by the Birth Defects Clinic at the University of Tennessee Hospital. This section describes the programming system which was used to create the AMH.

There are two methods presently used to implement an AMH (10, p. 91). The first method handles the question text and branching rules directly in the code of the program. Whenever the AMH needs to be changed, the actual program code must be changed. The second method treats the question text and branching information as data. The program which uses this method is called a driver. The term driver means a system written for the execution of a repetitive task which is data independent and which may be used by several users simultaneously (10, p. 91).

A questionnaire may consist of five different question types: free text answer, numerical answer, yes/no answer, single choice from list, and multiple choice from list.

Each question and its associated data is referred to as a frame and has a unique frame number. Free text and numerical answer frames consist of question text (Q), transfer (T), and format (F) information.

A free text frame is shown in Figure 4. The question text contains the question which is actually asked. It may consist of more than one line. The transfer is the frame number of the next question to be asked during the AMH. In the case of free text answers and numerical answers, there is only one transfer. The formats contain the text, including line-feed controls (!), which will be printed out during the summary. Like the question text, there may be more than one line of formats.

The yes/no, single choice and multiple choice questions also consist of question text, transfers, and formats plus responses (R) and prose (P). These three question types differ from the first two discussed above in that they have more than one transfer. During the execution of a question frame, the transfer which is taken corresponds to the particular response chosen. The responses are the possible choices which may be picked during an interview. The prose contains the text to be printed during the summary. Unlike the formats which are always printed, only one line of the prose will be printed, depending on the response to the question. Yes/no type and single choice type questions are shown in Figure 5 and 6, respectively. Yes/no type questions function just like the single choice type except that two of the choices are always "Yes" and "No". Multiple choice types also function like single choice types except more than one response may be chosen.

The driver centers around one program which implements the question types described above. The execution of each question frame shares this common routine. The program is completely independent of the data; that is, the same routine will handle any question frame.

```
TYPE:1

Q1:EXCERPT FROM OBSTETRICIAN'S REPORT
Q2:INITIALS OF MOTHER:

T1:2

F1:!!!!          QUESTIONS PERTAINING TO MOTHER OF INFANT
F2:!!EXCERPT FROM OBSTETRICIAN'S REPORT:!! 1.  INITIALS OF
MOTHER:
```

Figure 4. Free text answer type.

```
TYPE:3

Q1:DID YOU CONCEIVE DESPITE SUCH PRECAUTION:

R1:YES
R2:NO
R3:DON'T KNOW
R4:DON'T UNDERSTAND

T1:25
T2:25
T3:25
T4:25

F1:!1Ø.  CONCEIVED DESPITE PRECAUTION:

P1:YES
P2:NO
P3:
P4:NOT AVAILABLE
```

Figure 5. Yes/no answer type.

```
TYPE:4

Q1:FEATURES OF THIS PREGNANCY:
Q2:EMESIS

R1:FREQUENT
R2:OCCASIONAL
R3:NEVER
R4:NO DATA
R5:DATA NOT AVAILABLE

T1:5Ø
T2:5Ø
T3:6
T4:6
T5:6

F1:! 6.  FEATURES OF THIS PREGNANCY:!      A.  EMESIS:

P1:FREQUENT
P2:OCCASIONAL
P3:NEVER
P4:
P5:NOT AVAILABLE
```

Figure 6. Single choice from list type.

Also there is an interactive editor routine which allows the frames to be defined, deleted, or altered. When defining a frame, the editor, depending on the question type, asks for each part that is required. The editor also allows the user to view any frame which is defined.

This driver program is written in MUMPS. With the time-sharing used by the MUMPS system, the driver may administer different automated histories or the same history to different subjects at the same time. The driver has been used at the Massachusetts General Hospital in the administration of a general review of systems history and in a special chief complaint history.

The driver has two basic operation modes (10, p. 96). The driver is in the ASK mode during an on-line interview. It is in the PROSE mode where it is producing a summary of the interview. Figure 7 shows the general flow of the driver in ASK mode for a single question.

The driver program described provides a powerful and flexible method of creating an automated history. The driver technology is not only applicable to AMH interviews, but to any type of questionnaire or interview.

III. IMPLEMENTATION OF BDC QUESTIONNAIRE

General Implementation

The Birth Defects Clinic at the University of Tennessee Hospital uses several different questionnaire forms to gather information concerning its patients. The one that was chosen to be computerized for

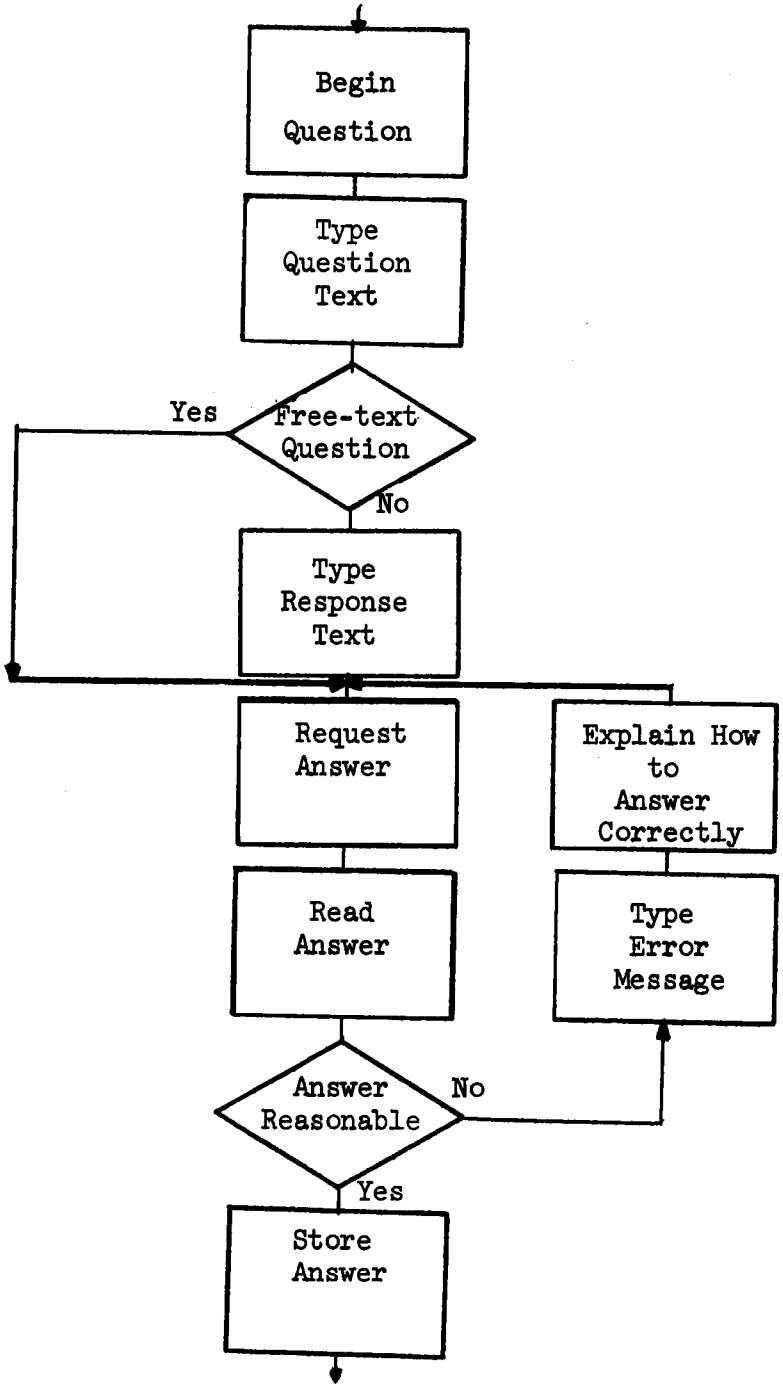


Figure 7. Operation sequence in ASK mode.

this project was a four-page questionnaire entitled "Questions Pertaining to the Mother of the Infant" as shown in Appendix A.

The first step was to use the editor, mentioned in the last section, to set up the questionnaire. The actual questionnaire was studied and each possible question was classified into one of four types: free-text answer, yes/no answer, single choice from list, or multiple choice from list. The proper flow of the questionnaire was determined; that is, for each question frame, the proper transfer or transfers were determined. The prose and formats were also done at this time. The summary obtained from the selected formats was made to look as much as possible like a completed questionnaire.

It was important to the Birth Defects Clinic to be able to determine which questions had not been answered. For this reason, all yes/no questions, single choice questions, and multiple choice questions included a response, "NO DATA" which corresponded to the question not being answered. A "NO DATA" response to a free-text question was indicated by hitting the "ENTER" key without typing a single character as an answer. Also the Clinic was interested in being able to indicate that certain data was missing or could not be found. For this reason, a response of "DATA NOT AVAILABLE" was added to the list of possible responses. This response could be indicated in a free-text question simply by typing in the response, "DATA NOT AVAILABLE."

It was decided that a sample of 36 patients would be used to demonstrate this project. The sample was selected at random by the Clinic. Even though the questionnaire frames were set up, several

existing routines needed modifications before an actual history could be taken of these 36 forms. All routines which were modified or written for this project are shown in Appendix B. They are all written in MUMPS, using the short or abbreviated form.

Main Driver, Interview, and Summary

The first routine to be discussed is the general driver for this package of programs. This routine, BDA, whose flowchart is shown in Figure 8, simply allows the user to select what task is to be done. It identifies the Birth Defects Clinic software, and then asks the user which routine or task out of six available tasks is desired. If the user does not know, he inputs the character ? and the names of the available routines are printed out. Each name begins with a unique letter; so if the user already knows the routine desired, all that is needed is to type the first letter of this routine. Then automatically, execution of that routine is started and the chosen task begun.

Four pieces of information, initials, a case-number, a K-number, and sex, are used to identify a patient. Each new patient to come to the clinic is given a unique case number. These numbers have been given out in ascending numerical order beginning with the number one. Also each patient who is diagnosed as having a Downs Syndrome is given a K-number, which is assigned just as the case-numbers are.

There are two routines, BDB and ICK (whose flowcharts are shown in Figures 9 and 10 respectively) which input the patient's

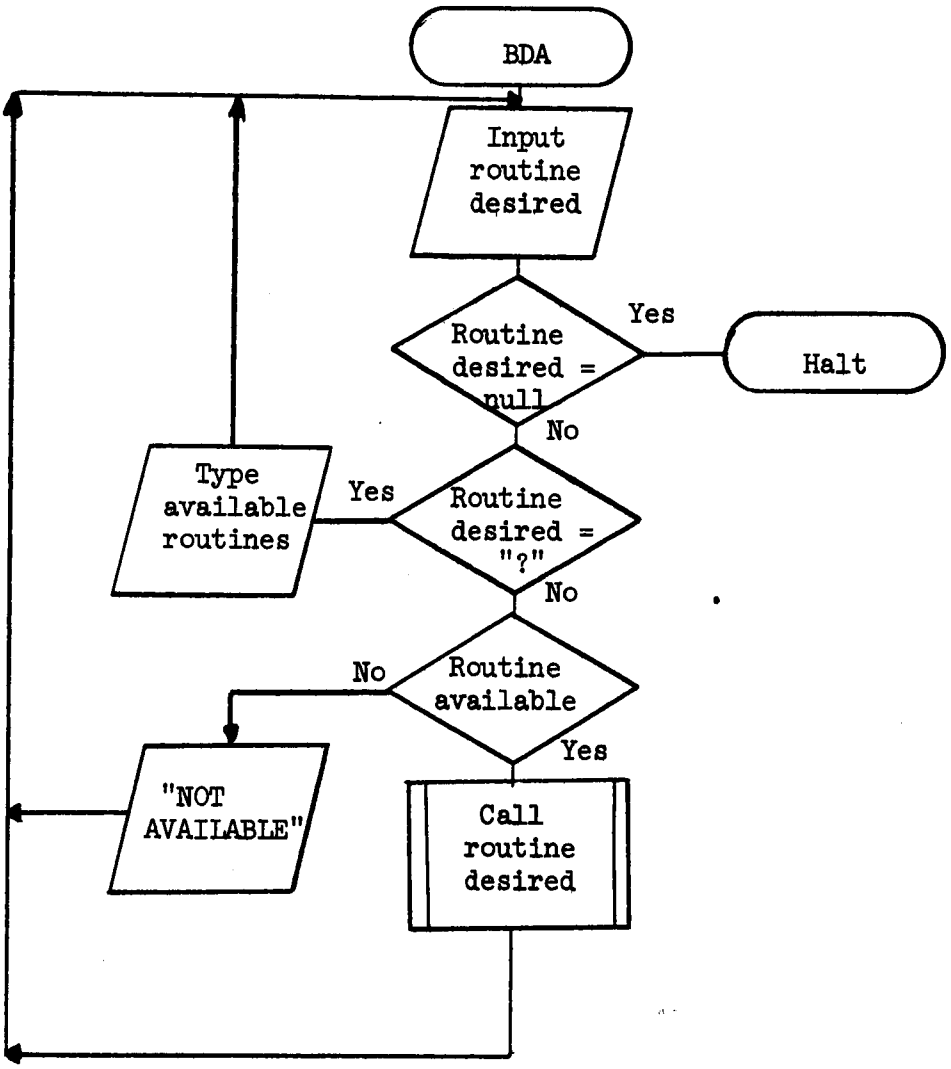


Figure 8. BDA - Driver.

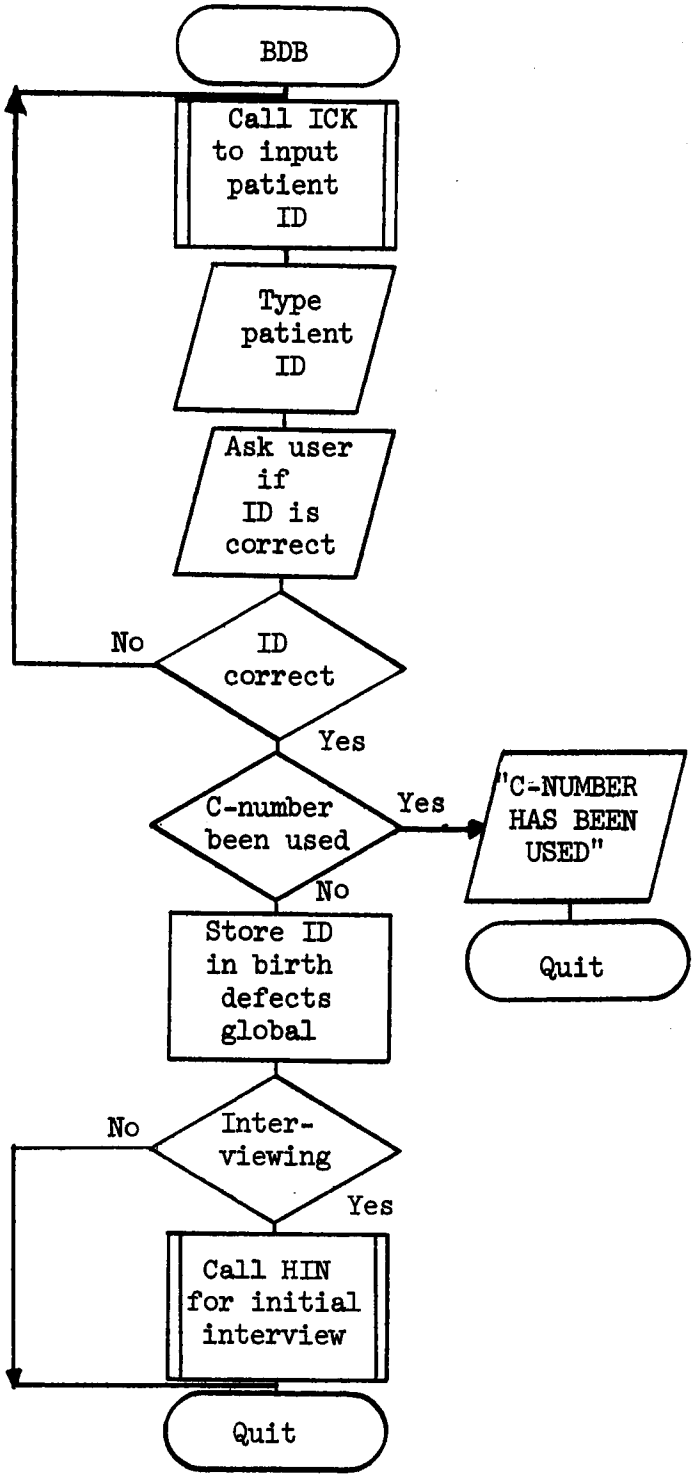


Figure 9. BDB - New patient data.

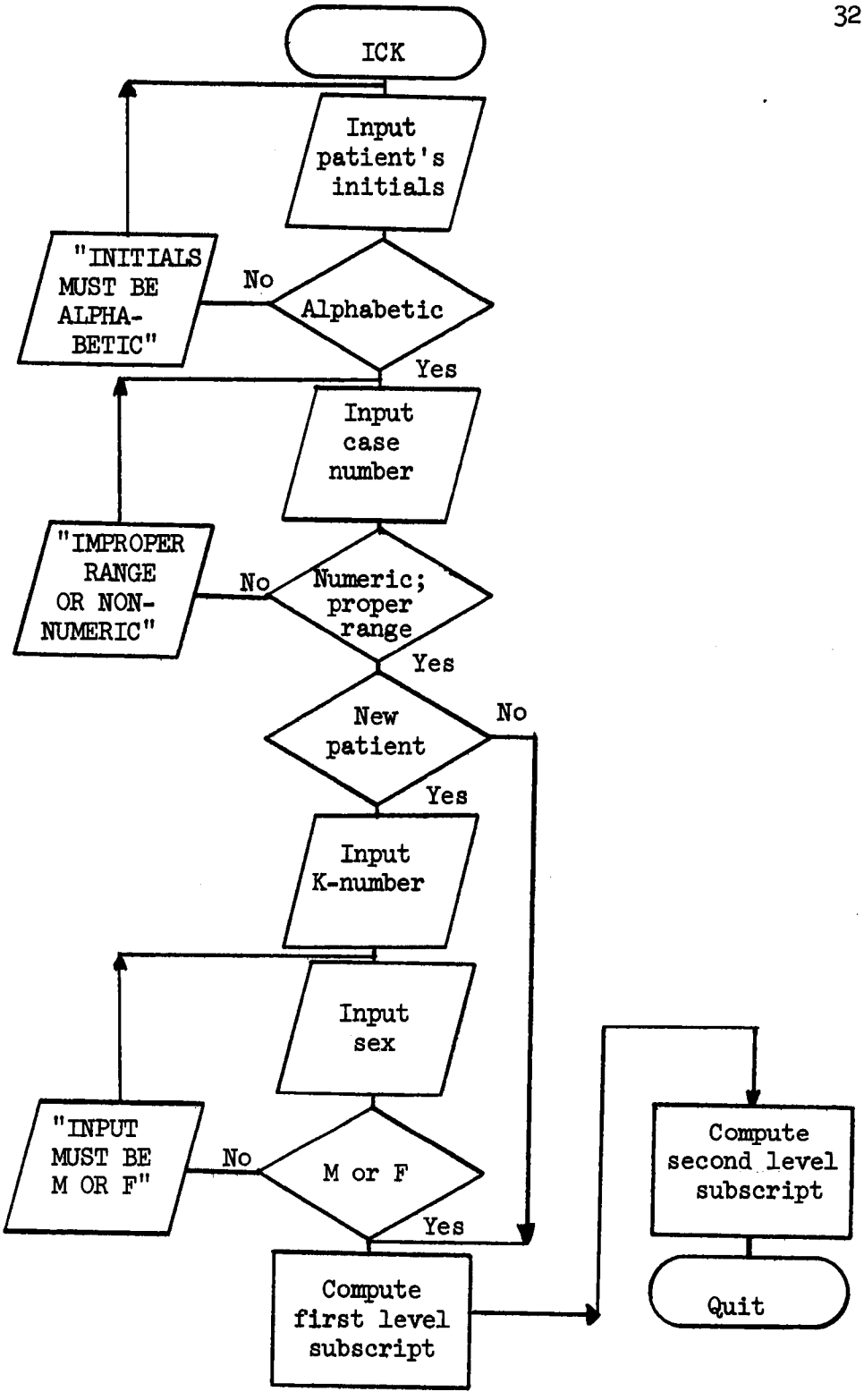


Figure 10. ICK - Input patient ID.

identification before a history can be taken. BDB calls ICK which inputs the four identifying items.

At this point it is important to understand the structure of the global array, ↑BDC, where the questionnaire answers are stored. This structure is shown in Figure 11. The first level consists of pointers which are obtained from a hash code on the patient's initials. This first level subscript is computed in ICK by summing two or three numbers, depending on whether the initials consist of two or three letters. The second level of the global also consists of pointers, which are obtained by dividing the case-number by 100. The third level contains the initials, case-number, K-number, and sex of the patient. And finally, the answers to the questionnaire are stored at the fourth level.

After BDB has entered the identifying items and computed the first two subscripts for the patient, the identifying items are typed out, and the user is asked to verify them. It is important that the initials and case-number be correct, since they determine the position of the data in the global. If the user sees an error, he may correct it, and new subscripts will be computed. This loop may be repeated as many times as needed to get the correct identifications. After the user has indicated that the identifications are correct, they are stored in the global and the actual history or interview is begun. The flowchart of the routine which does the history is shown in Figure 12.

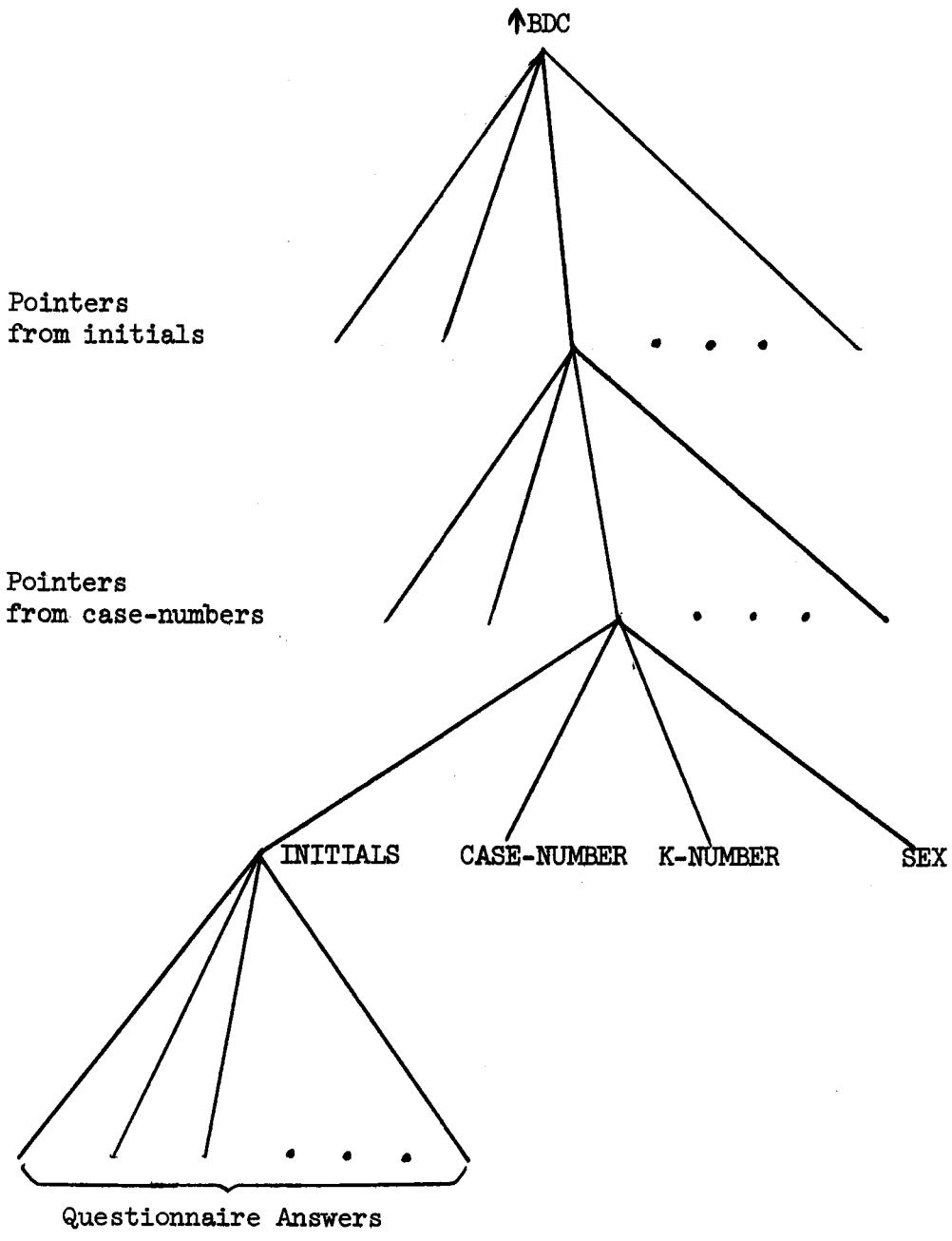


Figure 11. ↑BDC global array.

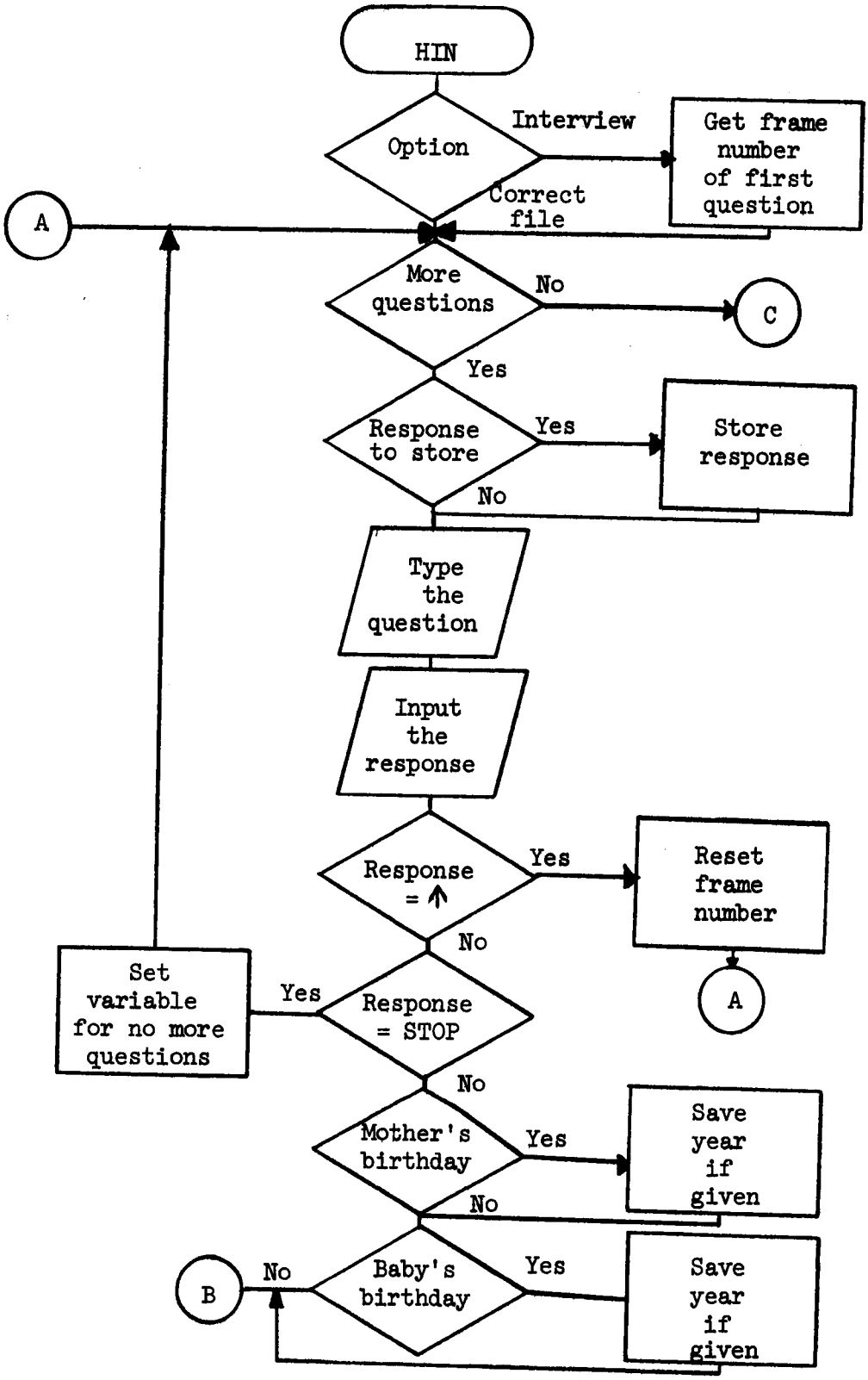


Figure 12. HIN - History interview.

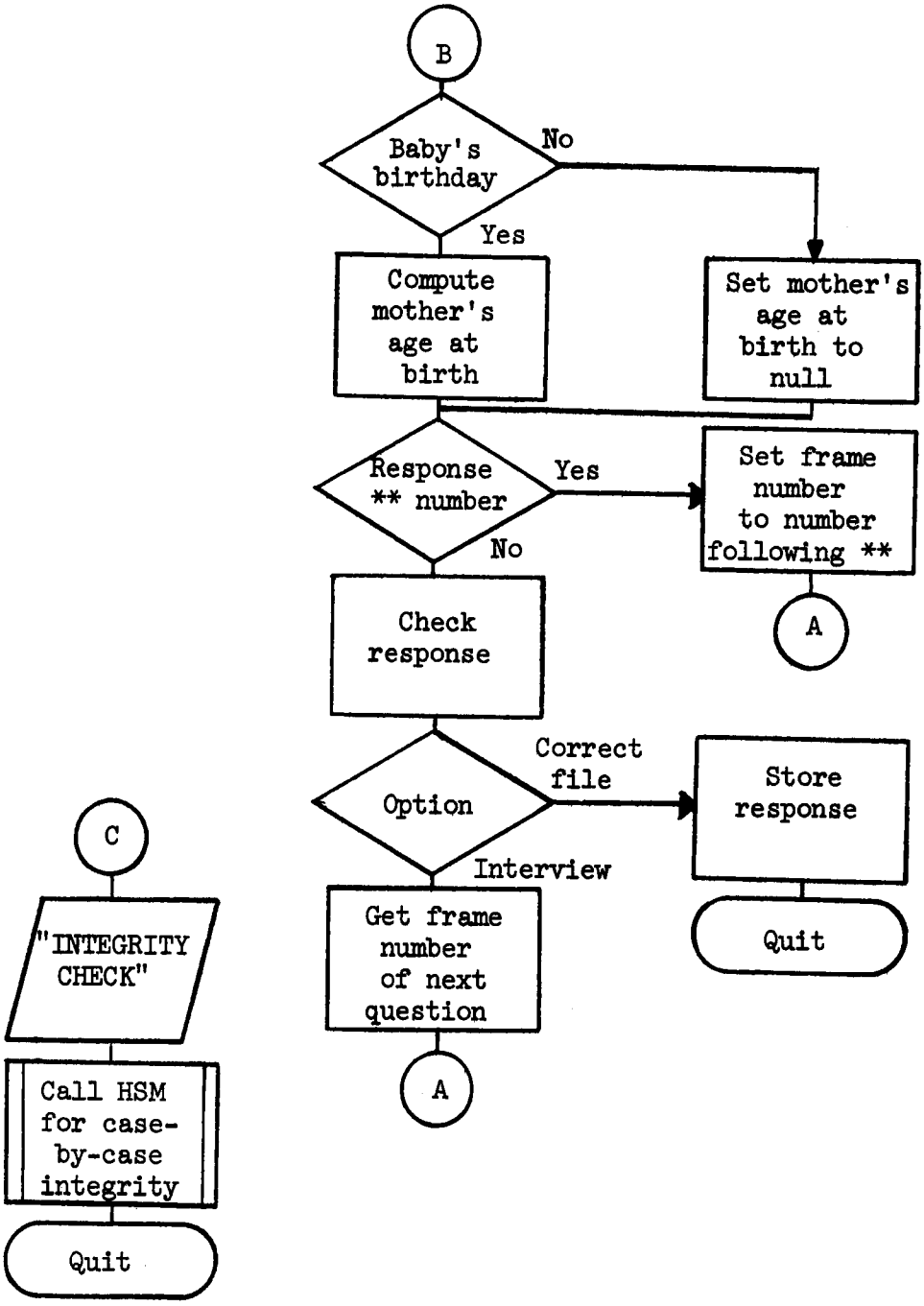


Figure 12. (continued)

The Birth Defects Clinic was especially interested in one data item, the age of the mother at the time of the child's birth. It was decided to make this computation, using the mother's birthday and child's birthday, in HIN. If both dates are available, the mother's age is stored at the fourth level of the global at node 275. If either or both dates are missing, this node is not defined.

With this modification to the interview routine and the two routines used to input the patient's identifications, the automated history was done for the 36 chosen questionnaires. The time to do one history using a CRT terminal is approximately 10 to 15 minutes, depending mainly on whether the free-text questions have long responses.

Although the routine, HSM, which produces a summary, was modified for the integrity routine, it needed no major modifications for the summary outputs. However, a routine, BDS was written to initiate the summary. The flowcharts for BDS and HSM are shown in Figure 13 and 14, respectively. BDS calls ICK which inputs only the initials and case-number of the patient whose summary is desired. It also computes the first and second subscripts so the questionnaire data may be found. After setting switches to assure that the summary option of HSM is taken, a call to HSM is made. The only modification to HSM concerning the summary was to type out an error message if there was no data stored in the ↑BDC global for the patient whose initials and case-number were typed in at the beginning of the summary.

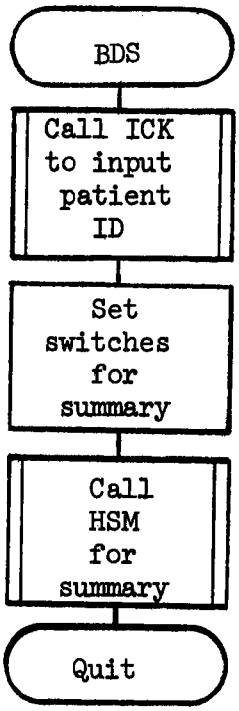


Figure 13. BDS - Patient data summary.

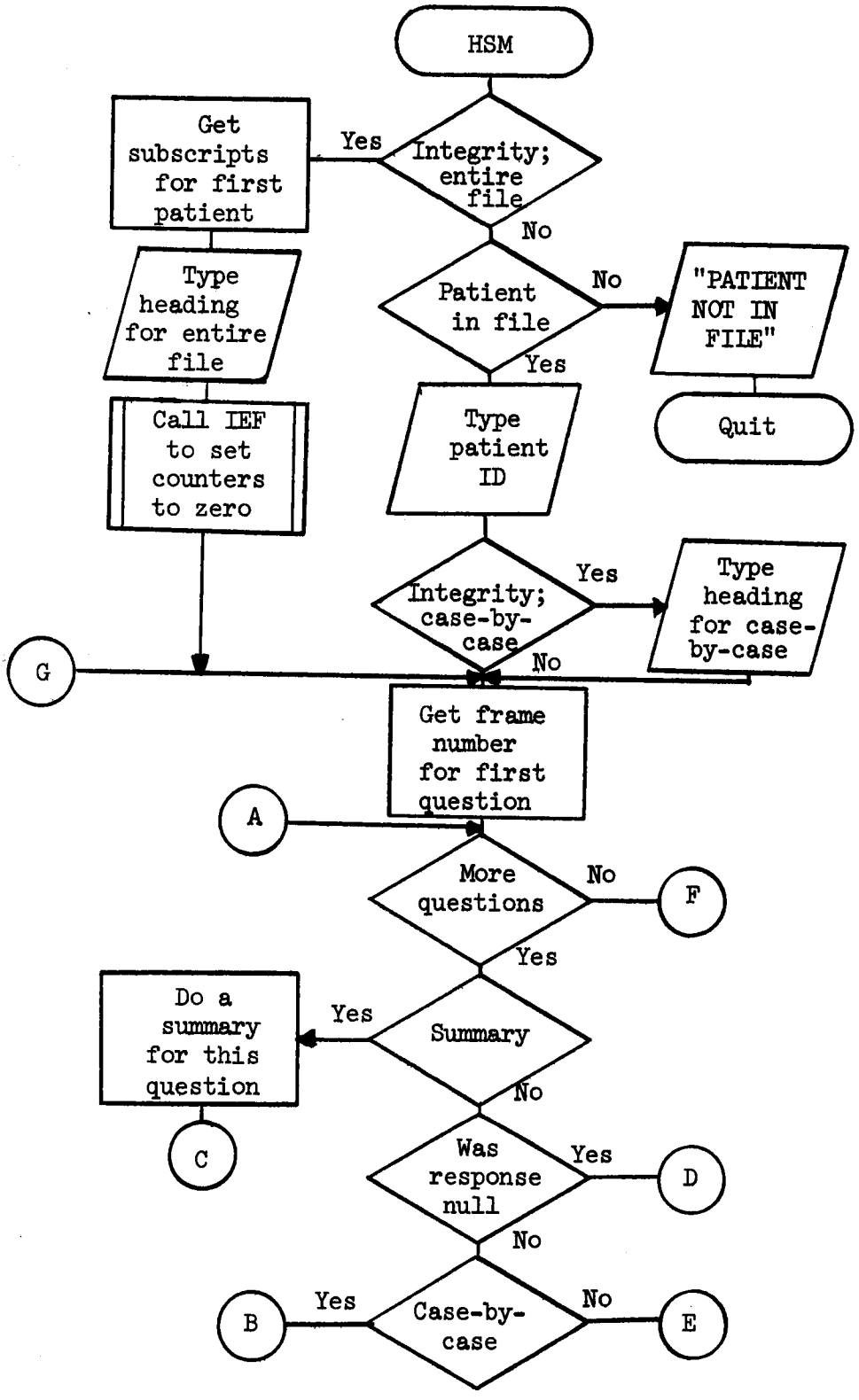


Figure 14. HSM - Summary; integrity check.

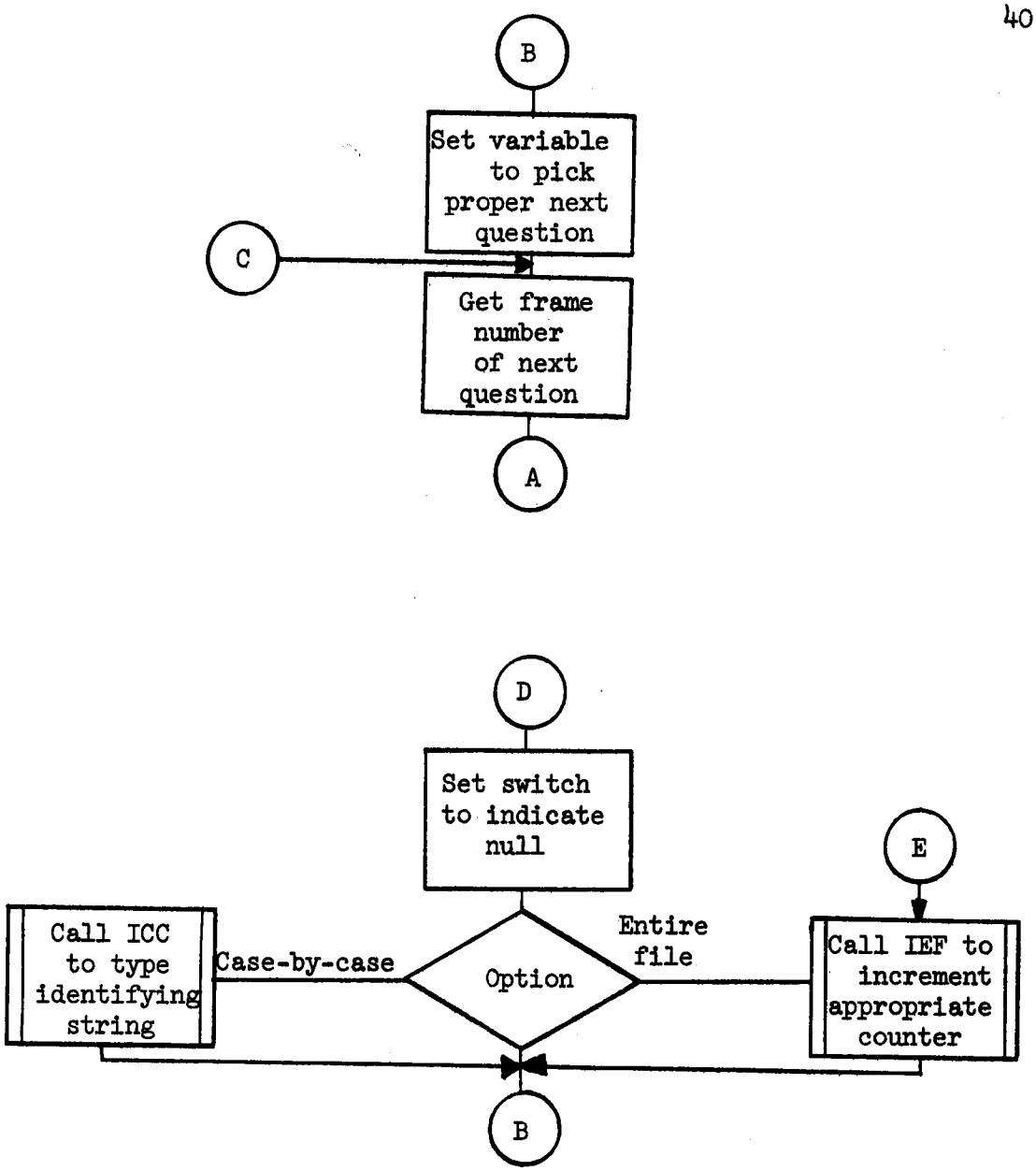


Figure 14. (continued)

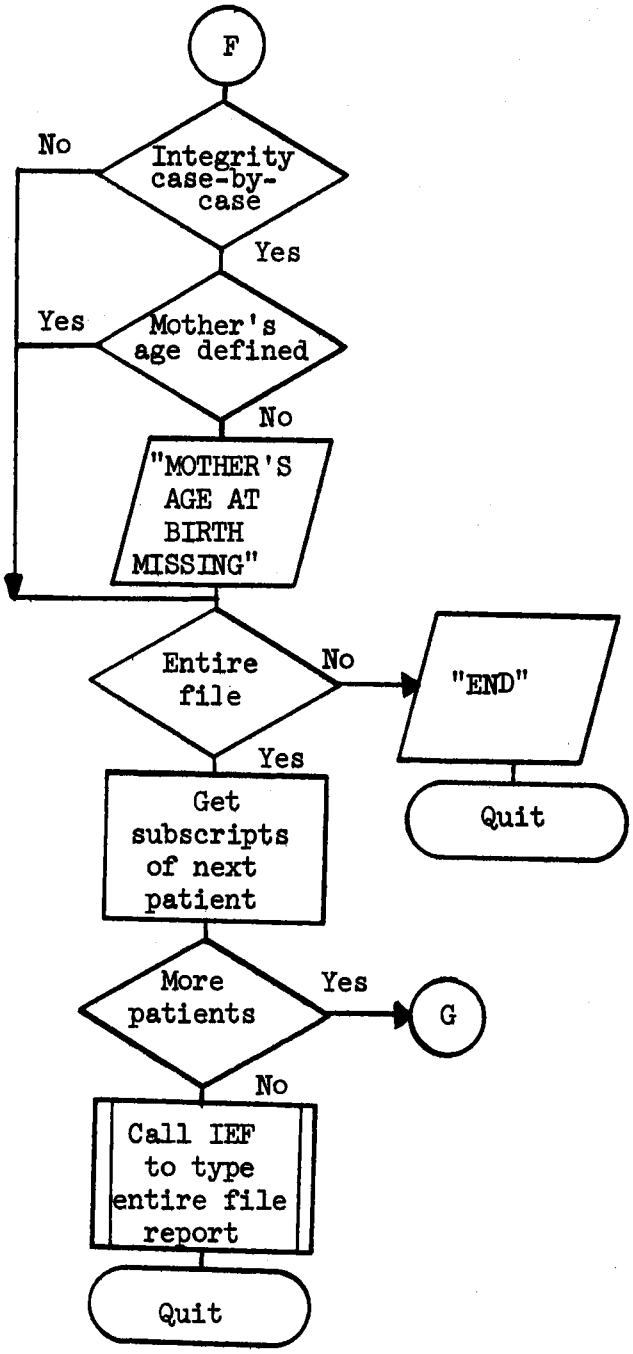


Figure 14. (continued)

Data Base Integrity Check

The data base integrity check procedure searches for questions which have been answered with a "NO DATA" response or a null response. The user may choose one of two options. The case by case integrity check prints out for an individual patient the questionnaire page number and question number for each question having a null response. The entire file integrity check prints out for every question the number of times it was asked, the number of times it was answered with a null response, and the percent of null responses. IDB, whose flowchart is shown in Figure 15, allows the user to select one of the two options. If the user chooses case by case, then the routine ICK, which has already been discussed, is called to input the initials and case-number of the patient for which the integrity check is to be done.

HSM which also does the summary, performs the majority of the work for both options of the integrity check. Considering the case by case option first, HSM first checks that the patient's identification is valid; that is, it performs a check to see if data for that patient is actually stored in the global. If the test fails, an error message is typed out. Otherwise HSM checks each answer for a null response. Whenever a null response is detected, a call is made to the routine, ICC, whose flowchart is shown in Figure 16. The frame number of the question having a null response is passed to ICC. ICC uses this number to find the string which identifies the question on the actual questionnaire. After all questions are checked, a check is made for

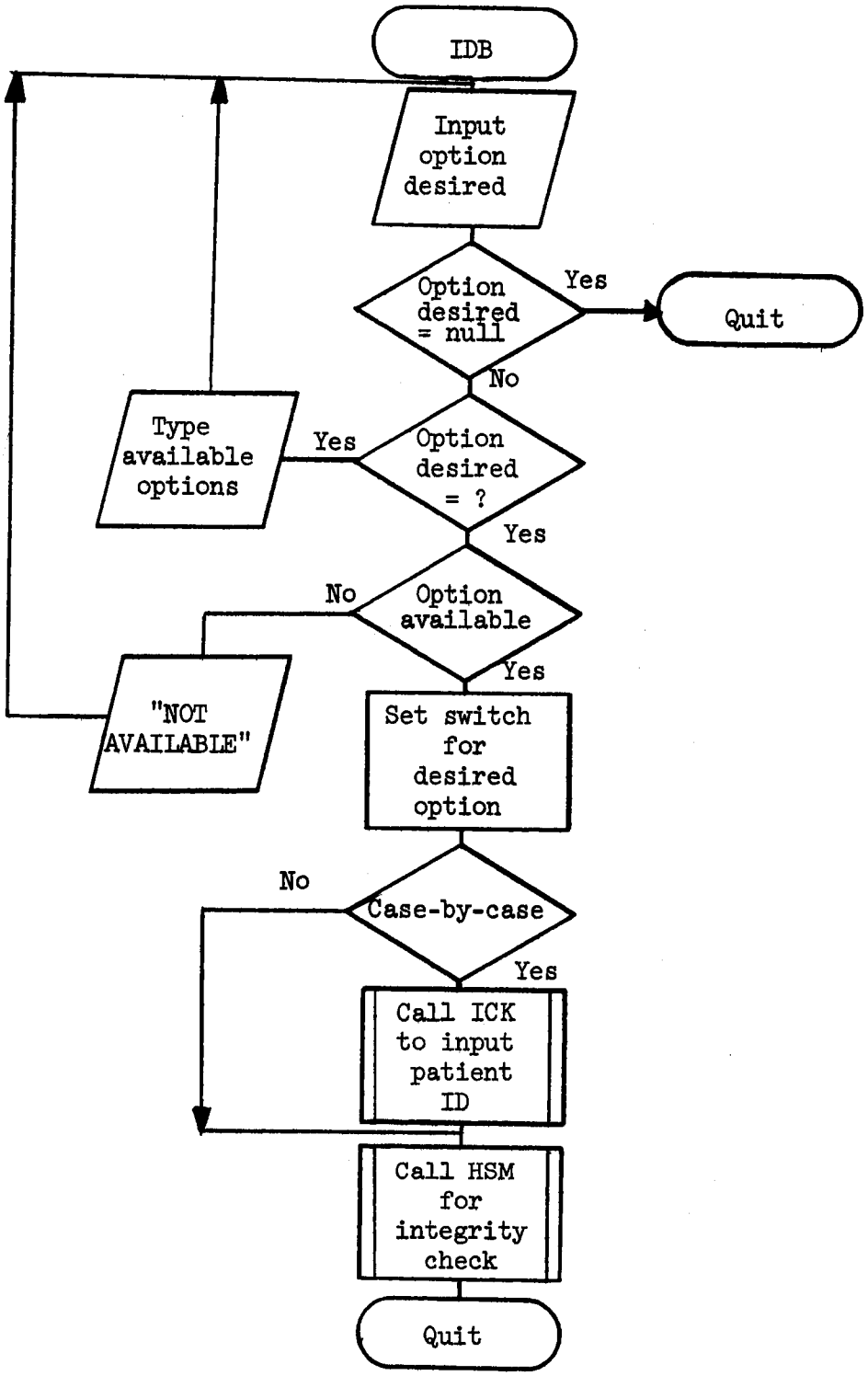


Figure 15. IDB - Integrity of data base.

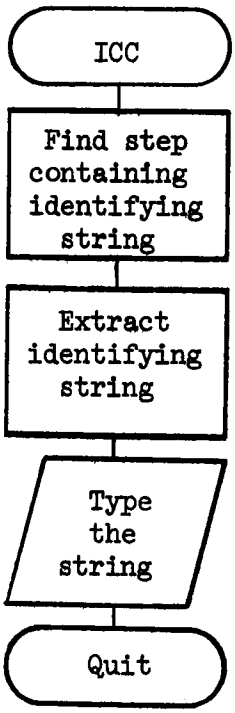


Figure 16. ICC - Integrity case by case.

the mother's age at delivery. If this age is missing, a message is printed out.

For the entire file integrity check, HSM makes calls to IEF, whose flowchart is shown in Figure 17. IEF actually performs three different functions depending on a program switch setting. In order to do the integrity, two accumulators, one to count the times the question was asked and another to count its null responses, are needed for each question of the questionnaire. HSM's first call to IEF cleared all these accumulators, which are stored in the global, ↑DQ, which contains the questionnaire data. Then for every patient in the birth defects global, each question is tested for a null response and a program switch is set. Then a call is made to IEF which increments the appropriate accumulators. Finally, after every question in the global has been checked, the last call is made to IEF. This time the percent of null responses is computed, and the question's identifying string, the number of times the question was asked, the number of times it was answered with a null response, and the percent of null responses are printed out for each question of the questionnaire.

Correction of Patient File

Once a history has been taken for a patient, some means is needed to correct erroneous input or answer questions whose answers had previously been null. This can be done using the correct patient file routine, CPF, whose flowchart is shown in Figure 18. Again, ICK is used to input the initials and case-number of the patient whose

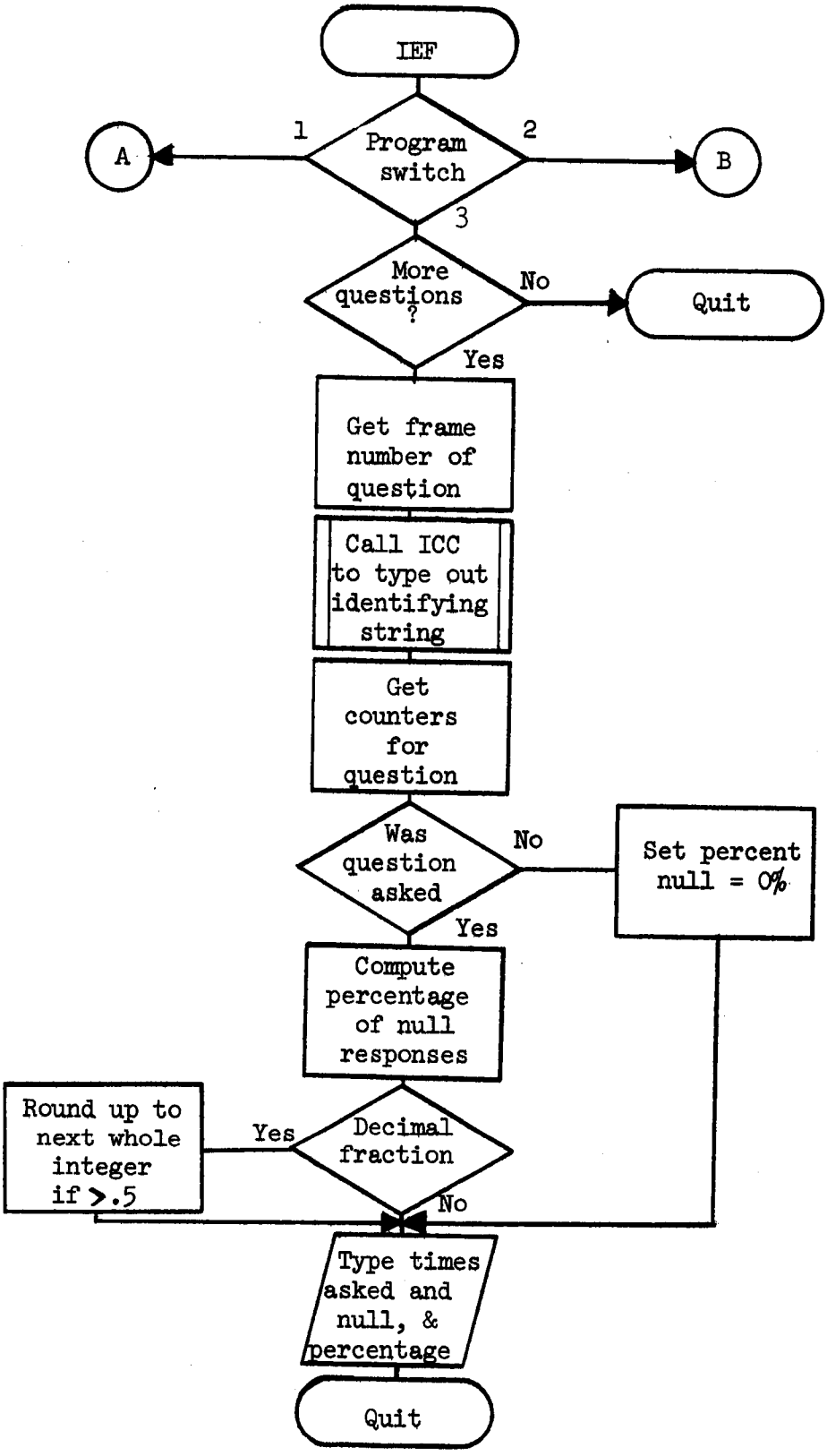


Figure 17. IEF - Integrity entire file.

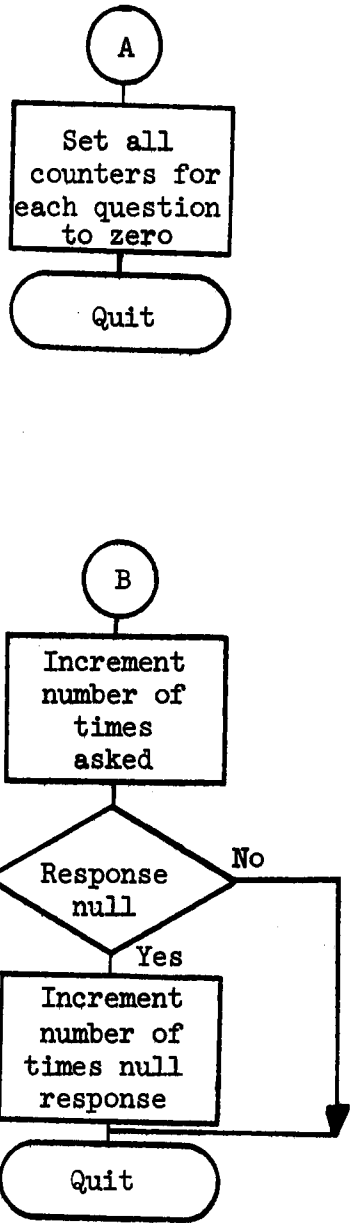


Figure 17. (continued)

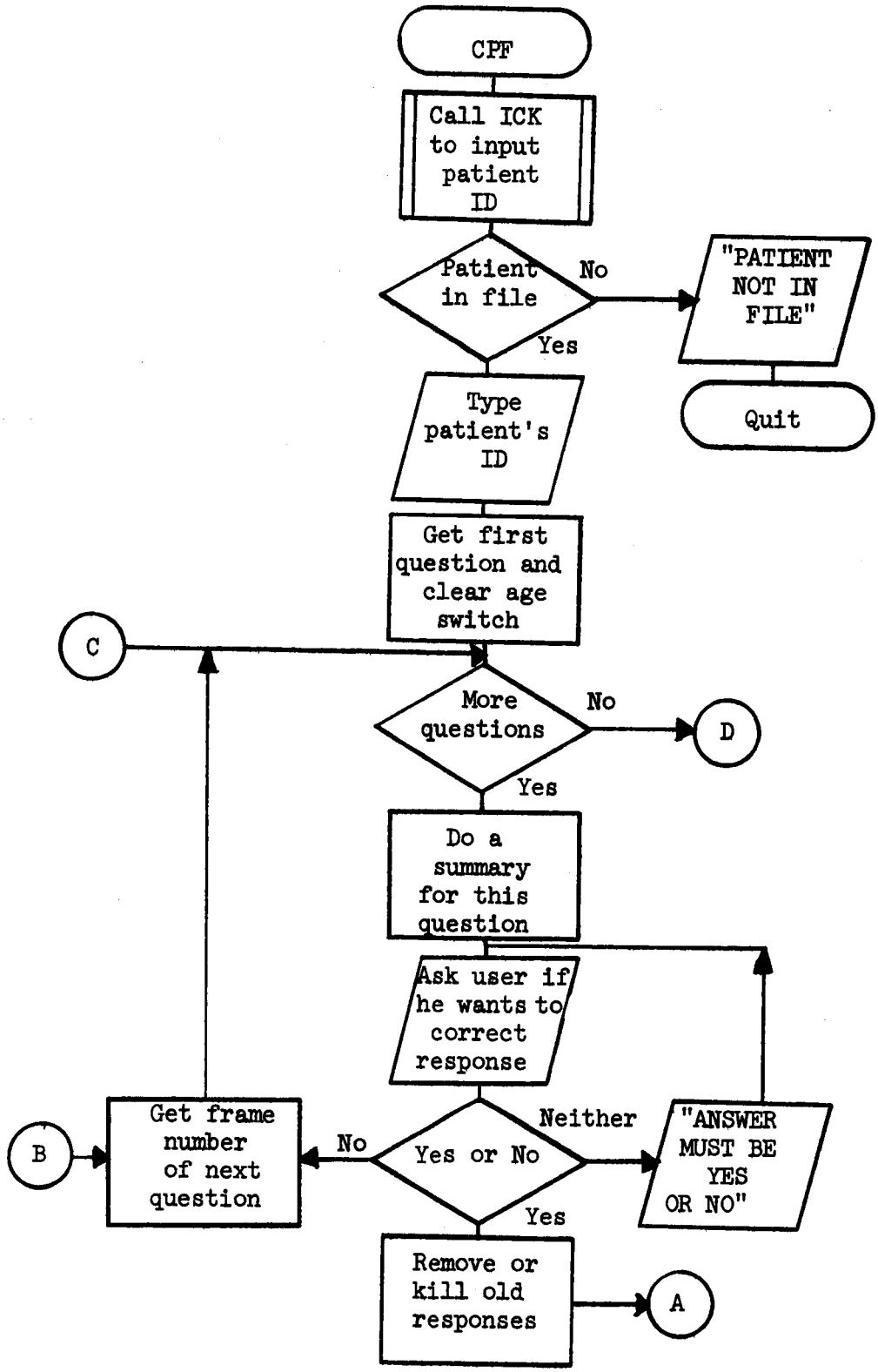


Figure 18. CPF - Correct patient file.

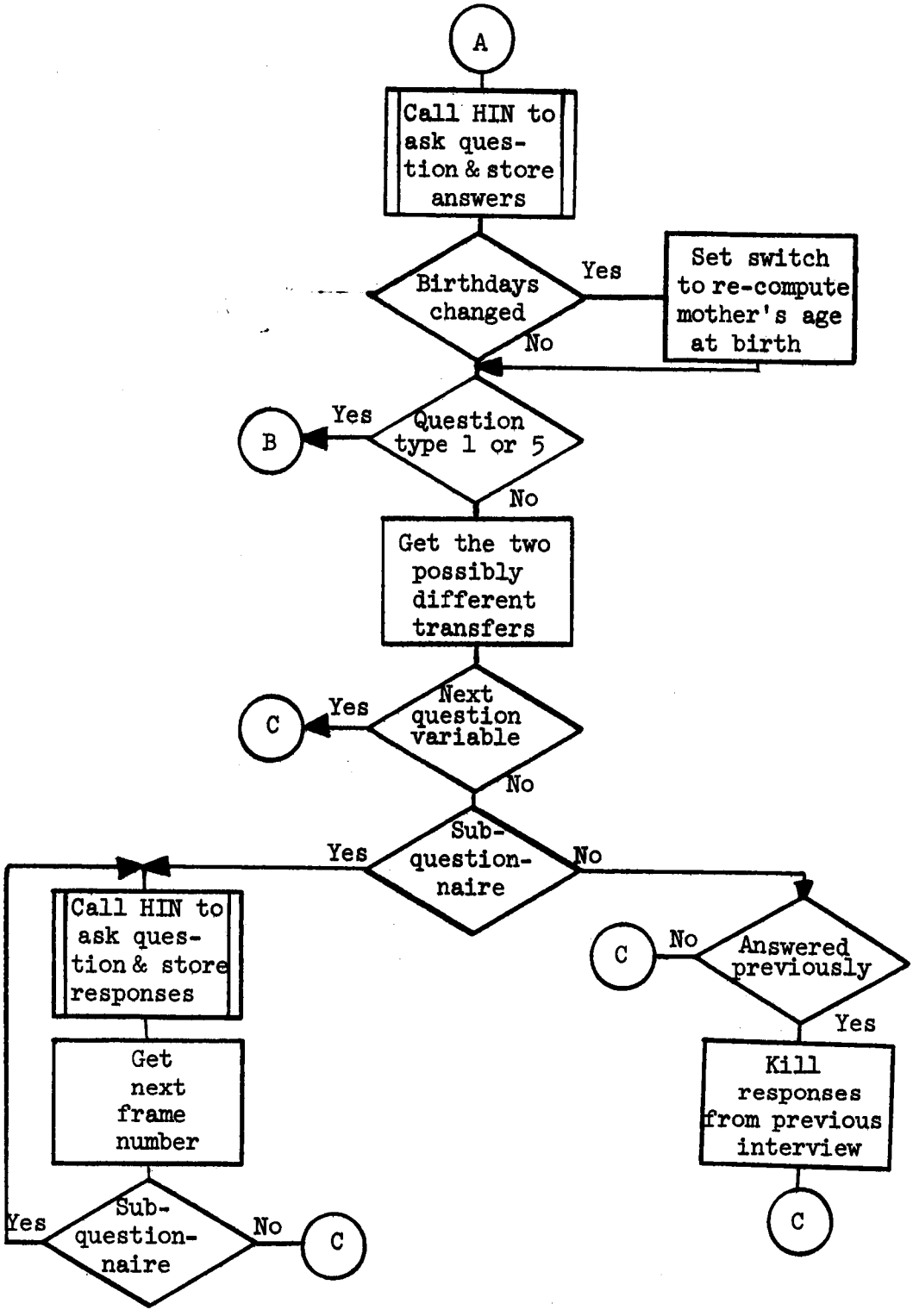


Figure 18. (continued)

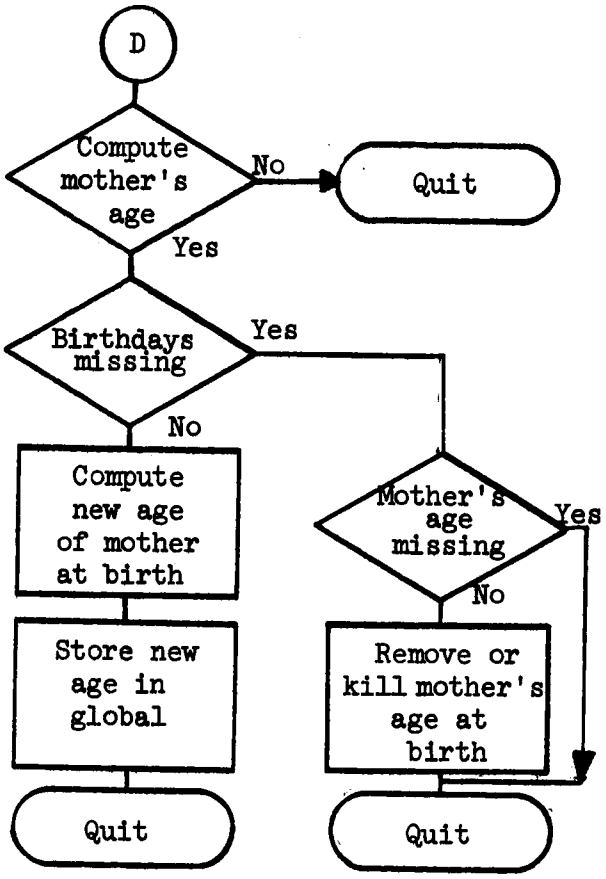


Figure 18. (continued)

data the user wishes to correct. If this patient is not in the global, a message is printed out. Otherwise, a summary for each question is done one at a time. After each summary, the user is asked if he would like to correct this response. At this point the user may type in "YES," "NO," or "STOP". If he types "NO", the program continues and types out the summary for the next question and again the user is asked if he would like to correct the response. If the user types in "STOP", the correcting program terminates. If the user types in a "YES", a call is made to the interviewing program, HIN, to re-ask the question. There are two things to be considered in changing a response to a question having multiple transfers. If the new response leads to a set of questions in order to get more detailed information, these questions must now be asked. Also if the new response does not lead to a more detailed set of questions and the previous response did, then the responses to these questions must be destroyed. Also if either the mother's birthday or the child's birthday or both were changed, then a new age of mother at delivery must be computed. Or if one or both of the dates were deleted, the computed age must also be deleted.

Listing Patients and Moving and Deleting Patients

There are two other options that were found to be needed. The first option gives a listing of the initials, case-number and K-number of every patient in the birth defects global. This task is done by the routine LST, whose flowchart is shown in Figure 19. The second needed option is moving or deletion of a patient. Under this task,

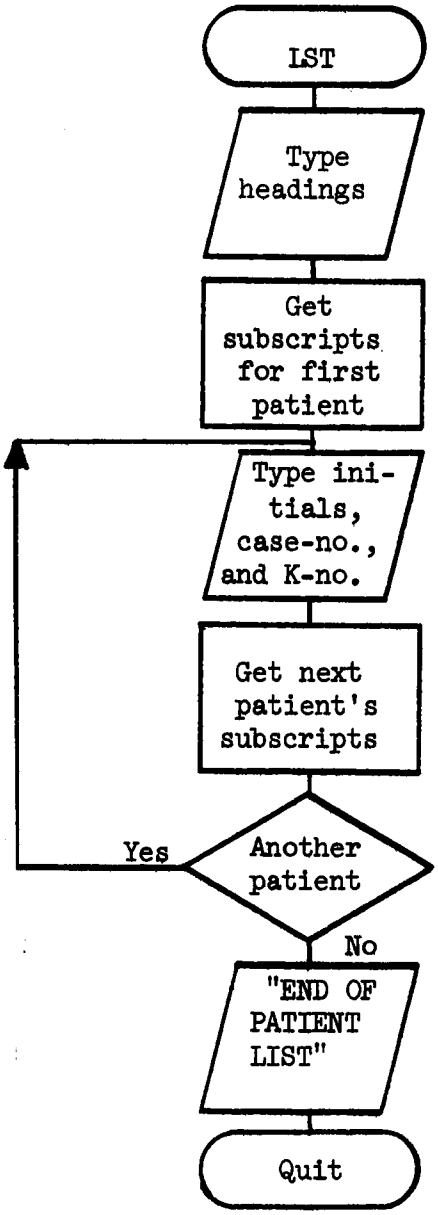


Figure 19. LST - List patients.

there are two options. One, the data for an individual patient may be deleted from the birth defects global. The second option is to move the data using a new set of initials and case-number, that is, the first two subscripts of the data are changed. This option is useful when a mistake has been made in a patient's identification. The routine that performs this task is MDP, and its flowchart is shown in Figure 20.

Data Search

The data search routines allow the user to find all patients in the file whose responses to up to a maximum of five questions match the responses which the user types in. For example, the user may want a list of all patients whose fathers smoke or a list of all patients whose fathers smoke and whose mothers smoke. Whenever the user is searching on morethan one item, the logical operator used is the Boolean AND.

The data search option uses two routines. The setup routine DTS, whose flowchart is shown in Figure 21, allows the user to search for one to five items. The user inputs the frame number of the automated questionnaire which corresponds to an actual questionnaire question. A hardcopy key which forms the association of frame number to questionnaire question number is used. After the user has types in a frame number and the number is checked for validity, the question corresponding to that frame number is asked. The response which the user chooses forms the specification for this particular item. For

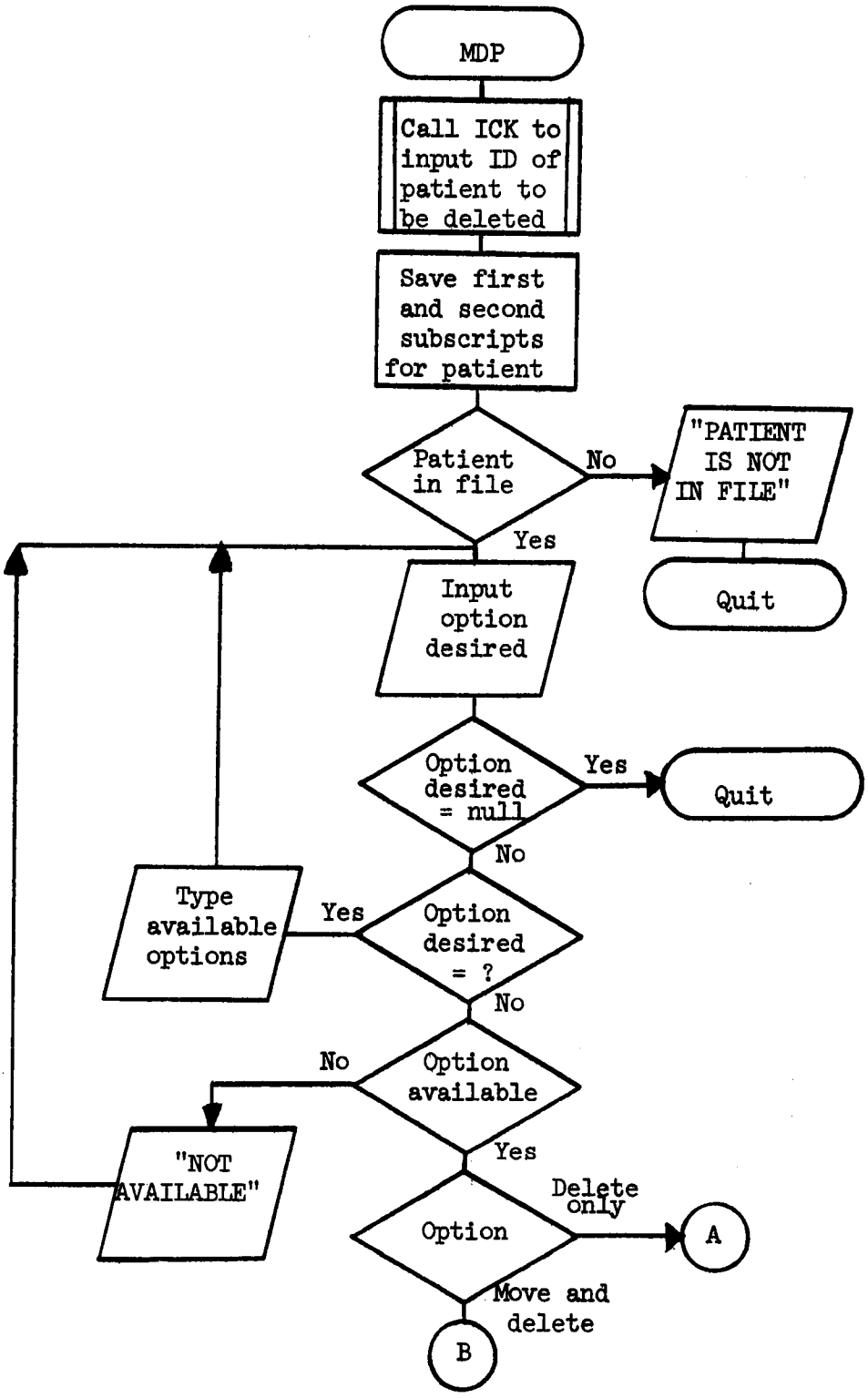


Figure 20. MDP - Move and delete patient.

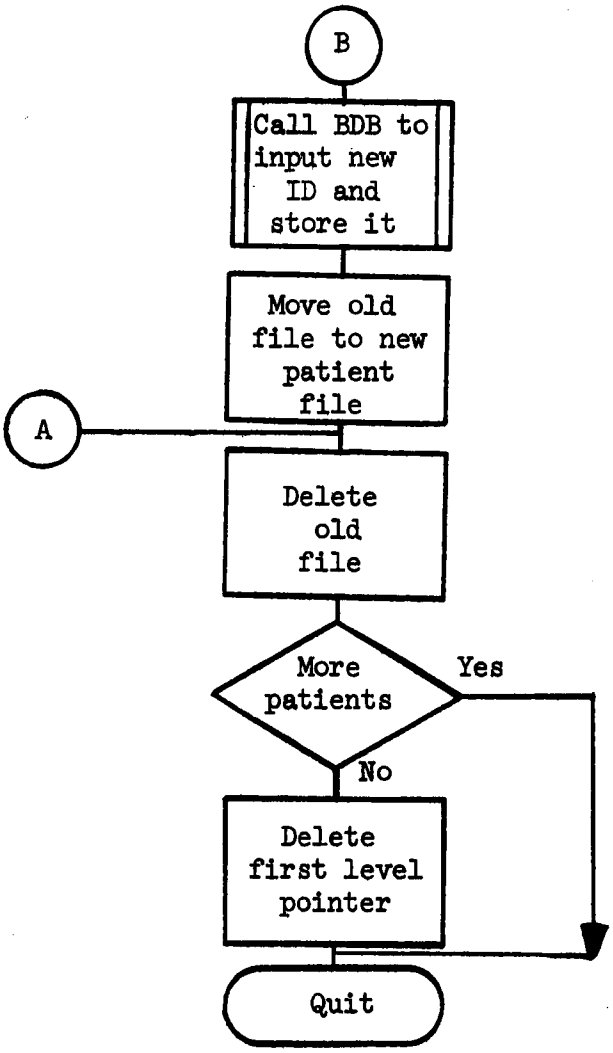


Figure 20. (continued)

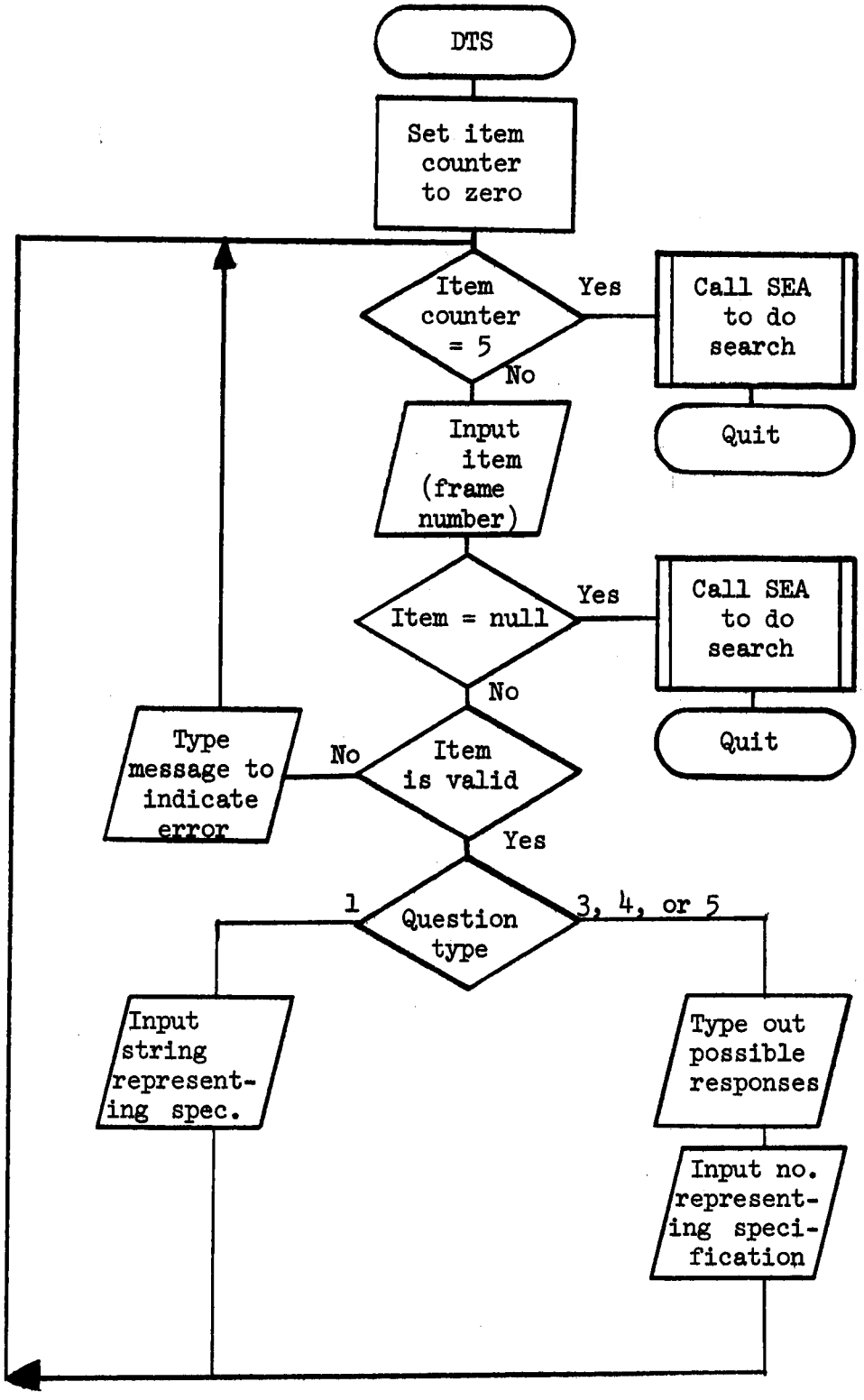


Figure 21. DTS - Data search setup.

example, if the user types in the frame number associated with the frame shown in Figure 5, the question in Q1 would be printed out, along with the three responses. Then if the user chose the first response, the initials, case-number, and K-number of all patients who answered YES to that question would be printed out. The user's responses are stored in a local array with the frame numbers functioning as subscripts. The basic loop of inputting a frame number followed by a response to the associated question may be repeated five times.

After the user has specified five items or has input a null item, a routine which does the actual data search is called. The flowchart of this routine, SEA, is shown in Figure 22. Throughout the search, counts of the total number of patients in the file, the number of hits, and the number of patients who have not answered at least one of the questions are computed. It is possible to search for a null answer, in which case the last counter mentioned would not be incremented. For every patient in the file, SEA looks at the responses to the questions which correspond with the item or items the user typed in during the setup part. If each response matches the user's specification, the patient's identification is printed out. In other words, a hit has been made. On the otherhand, whenever a response does not match the user's specification, the search on that patient is terminated, and the next patient's responses are checked. This is continued until all patients have been checked. At this point, the three counts, total population, total hits, and total null responses are printed out.

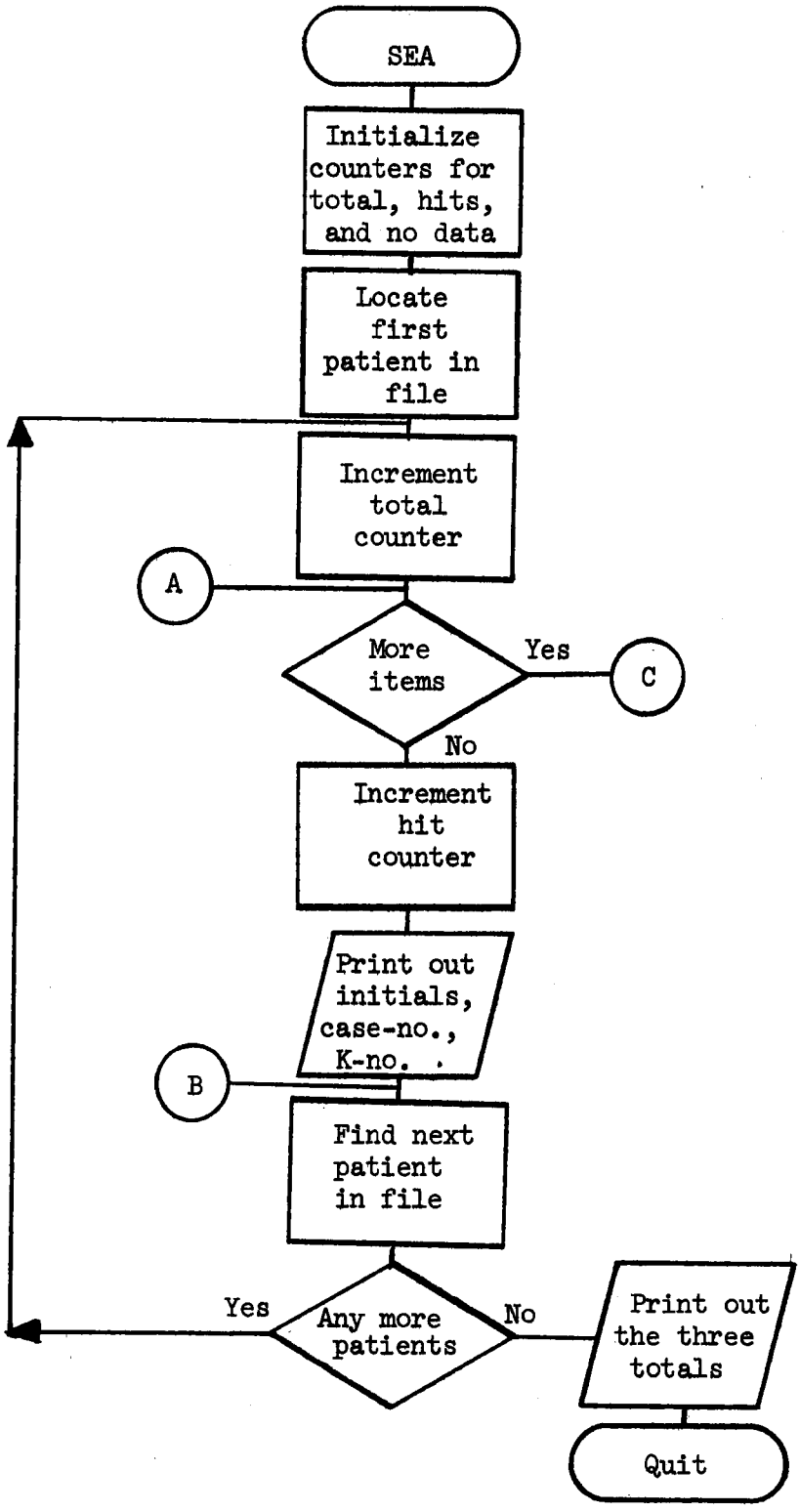


Figure 22. SEA - Data search.

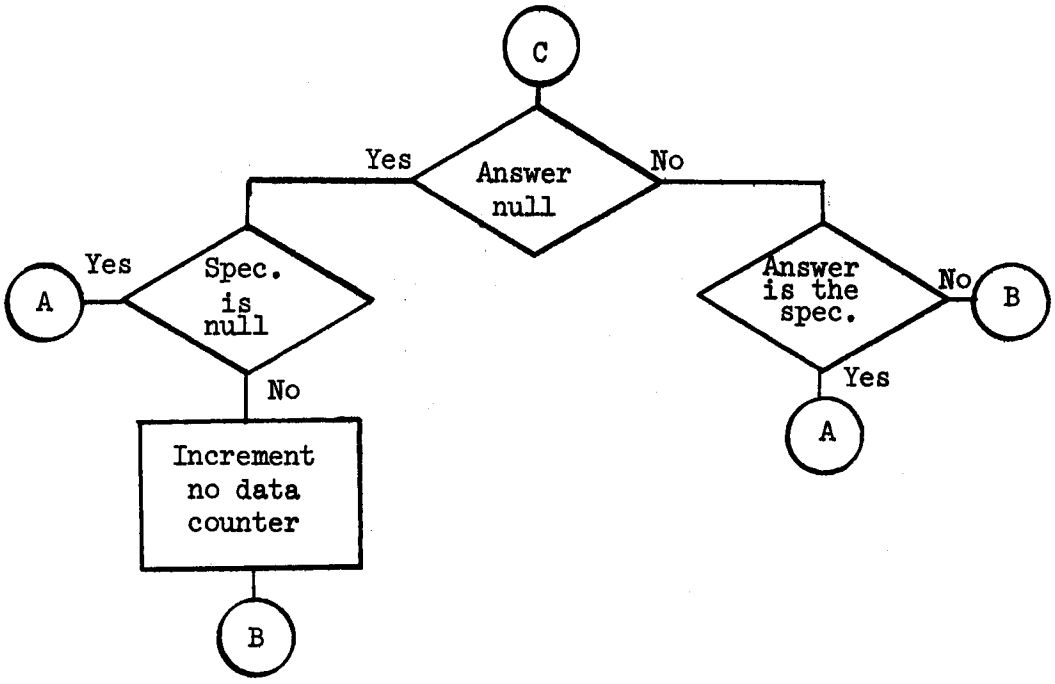


Figure 22. (continued)

Essentially, any answer to any question from the questionnaire may be searched for. Not only may the data search option be used to find simple correlations, but also to perform a data base integrity check. For example, from a data search, the user could find all patients who gave a null answer to some particular question.

CHAPTER IV

CONCLUSIONS AND RECOMMENDATIONS

Prior to the completion of this project, to obtain any information about a patient seen at the Birth Defects Clinic, one would have to go to a set of four large file cabinets and find that person's file. The file consists of several questionnaires, physician's notes, and other records which give little appearance of organization and are also sometimes illegible. Often too, a patient's file is not to be found in a cabinet but in use by one of several possible people. Now the data from the questionnaire, "Questions Pertaining to the Mother of the Infant", may be stored on a disk through the use of the software package developed for this thesis. For former patients, the information can be typed in using the original hardcopy of the questionnaire. For new patients, the information may be stored directly on disk, eliminating the four-page questionnaire.

Once the data for this questionnaire is stored for a patient, the Birth Defects Clinic has several options available to it. If a hardcopy of the questionnaire is needed, it may be obtained using the summary option. The same summary is available for rapid inspection, using the display screen. Also if any information is found to be incomplete or incorrect, that information may be corrected without going back through an interview.

Using the integrity check option, the clinic can find out which questions were not answered for any patients. If the interest of the clinic is complete data, an integrity check could be done whenever a patient is in for a visit, and the missing data could be obtained. The clinic can also analyze the questionnaire using the second option of integrity check, which prints out for each question the number of times the question was asked, the number of times answered with a null response, and the percentage of null responses. The clinic could possibly discover that some questions are irrelevant and need not even be included in the questionnaire.

The data search option is perhaps the most powerful. As discussed in Chapter III, the user can search all patients' files for specific responses to any questions up to a maximum of five questions. Doing this task without the use of the computer would be a very distasteful and time-consuming job. The clinic now has a tool to find possible trends and simple correlations concerning information about its patients.

From the programmer's and user's viewpoint, the software could have functioned more elegantly if each question in the questionnaire had had a unique question number. For example, there is a question number six on both page two and page three. Also there are two questions numbered one on page one. Even worse, there are questions like number six a on page one consisting of more than one question. This ambiguity caused problems whenever a question number needed to be printed out for inspection or typed in by the user. For both the

integrity check and the data search keys which associate actual questionnaire questions with frame numbers or identifying strings must be used. This extra work could have been eliminated had a better numbering system originally been used. Since each question is automated in the form of a question frame and each frame has a unique frame number, the most desirable numbering system would be one where each question has a unique number.

LIST OF REFERENCES

LIST OF REFERENCES

1. Barnett, G. Octo. "Computers in Patient Care," New England Journal of Medicine, 279:1321-1327, 1968.

2. Barnett, G. Octo. "The Modular Hospital Information System," Computers and Biomedical Research, 4:243-266, 1974.

3. Beers, Ray F., Christine Bernecker, Gerald L. Gottlieb, and Max Samter. "An Approach to Automation of Medical Interviews," Computers and Biomedical Research, 5:99-107, 1972.

4. Brodman, Keever, Albert J. Erdmann, Irving Lorge, Harold G. Wolff, and Todd H. Broadbent. "The Cornell Medical Index," Journal of American Medical Association, 140:530-534, 1949.

5. Greenes, R. A., A. N. Pappalardo, C. W. Marble, and G. O. Barnett. "A System for Clinical Data Management," Proceedings of 1969 Fall Joint Computer Conference, AFIPS, 35:297-305, 1969.

6. Greenes, R. A., A. N. Pappalardo, C. W. Barble, and G. O. Barnett. "Design and Implementation of a Clinical Data Management System," Computers and Biomedical Research, 2:469-485, 1969.

7. Kanner, Irving F. "Programmed Medical History-Taking With or Without Computer," Journal of American Medical Association, 207:317-321, 1969.

8. Kieby, Joseph M., John L. Juergens, Bradner L. Hisey, and Paul E. Williams. "A Computer-Based Medical Record," Journal of American Medical Association, 205:99-104, 1968.

9. Slack, Warner V., G. Phillip Hicks, Charles E. Reed, and Lawrence J. Van Cura. "A Computer-Based Medical-History System," New England Journal of Medicine, 274:194-198, 1966.

10. Swedlow, David B., G. Octo Barnett, Jerome H. Grossman, and Daniel E. Souder. "A Simple Programming System ("Driver") for the Creation and Execution of an Automated Medical History," Computer and Biomedical Research, 5:90-98, 1972.

11. Swedlow, David B., J. Grossman, G. O. Barnett, and M. McGuire. "The Collection of Medical History Data Using a Real-Time Computer System," Proceedings of Annual Conference on Engineering and Biology, 10:393, 1968.

APPENDIXES

APPENDIX A

QUESTIONS PERTAINING TO THE MOTHER OF THE INFANT

EXCERPT FROM OBSTETRICIAN'S REPORT:

1. Initials of mother _____

2. Date of birth

Day	Month	Year

3. Family status

Single	Married	Widowed	Divorced

4. Previous pregnancies and confinements (no.) [ ]

5. Noteworthy details of pregnancies and deliveries, such as any malformations, illness or unusual features of children:

_____

_____

6. Features of this pregnancy

a. Emesis (Frequent - Occasional - Never)
(From - To)

+	o	n

b. Uterine bleeding (indicate month(s))

Months

c. Other disorders associated with pregnancy: _____

_____

QUESTIONS PERTAINING TO DELIVERY AND INFANT:

1. Delivery (Date: _____ Place: _____)

2. If surgical, give details: _____

_____

Unusual features (placenta, fetal membranes, umbilical cord, amniotic fluid)

3. Infant:

   a. Signs of maturity [Yes | No]

   b. Length and birth weight: _____

   c. Postnatal behavior: _____

   d. Unusual features (malformations): _____

QUESTIONS RELATED TO DELIVERY:

1. Miscarriages (number) [ ]

2. Gynecological operations, including curettages: _____

   _____

3. During the first three months of pregnancy, did you suffer from vomiting? [Yes | No]

4. During the first three months of pregnancy, did you take anything to relieve pain? [Regularly | Occasionally | Never]

   a. Which preparation(s)? _____

   b. How much a day? _____

   c. For how long? From _____ To _____

5. During the first three months of pregnancy, did you take drugs to help you sleep or calm you down? [Regularly | Occasionally | Never]

   a. Which preparation(s)? _____

   b. How much a day? _____

   c. For how long? From _____ To _____

6. During the first three months of pregnancy, did you take or were given any hormone preparations? [Yes | No]

   a. Which preparation(s)? _____

   b. How much a day? _____

c. For how long? From _____ To _____

7. During this time, did you take drugs to relieve vomiting? Yes | No

a. Which preparation(s)? _____

b. How much a day? _____

c. For how long? From _____ To _____

8. During this time, did you take any other drugs? Yes | No

a. Which preparation(s)? _____

b. How much? _____

c. For how long? From _____ To _____

9. Did you use contraceptives (oral or local)? Yes | No

a. What did you use? _____

10. Did you conceive despite such precaution? Yes | No

OTHER QUESTIONS:

1. Mother's occupation:

a. Formerly: _____

b. Now: _____

2. Date of marriage:

Day	Month	Year

3. Are the parents blood relations? Yes | No

a. In what degree? _____

4. X-ray or radium irradiation (occupational or diagnostical)? Yes | No

a. Before pregnancy? Yes | No

b. During pregnancy? Yes | No

5. Before pregnancy, did you take certain drugs regularly or at least frequently? [Yes] [No]

   a. Which preparation(s)? _____

6. Poisoning, chronic exposure to toxic agents (excl. nicotine and alcohol)? [Yes] [No]

   a. Detail: _____

7. Nicotine? [Yes] [No]

   a. At what age did you start smoking? _____

   b. How many cigarettes per day? _____

   c. When in the pregnancy did you stop? _____

8. Alcohol? [Regularly] [Occasionally] [Never]

   a. If regularly, what type of drink? _____

9. Infectious diseases during early pregnancy (Rubella, mumps, polio, influenze, hepatitis, etc.)? [Yes] [No]

   a. Which disease(s)? _____

10. Other diseases or accidents during or just before pregnancy? [Yes] [No]

    a. Which? _____

Diabetes: If so, stabilized during early pregnancy? [Yes] [No]

   a. Coma, frequency of: _____

Hypoglycemia? [Yes] [No]

   a. Shock, frequency of: _____

Allergies (hypersensitivities)? [Yes] [No]

   a. Details: _____

QUESTIONS PERTAINING TO THE FATHER OF THE INFANT:

1. Date of birth

Day	Month	Year

2. Occupation: _____

3. Nicotine? [Yes] [No]

   a. At what age did he start smoking? _____

   b. Cigarettes, cigars, pipe, chewing tobacco? [ct] [c] [p] [ct]

   c. How many a day? _____

4. Alcohol? [Regularly] [Occasionally] [Never]

   a. If regularly: heavy or moderate? [H] [M]

   b. What type of drink? _____

5. X-ray and radium irradiation? [Yes] [No]

   a. Occupational, diagnostical, therapeutical? [O] [D] [Th]

   b. Details: _____

6. Does he regularly take certain drugs? [Yes] [No]

   a. Which preparations? _____

QUESTIONS FOR BOTH PARENTS:

In the families of both parents, are constitutional anomalies known to have occurred (malformations, feeblemindedness, etc.)? [Yes] [No]

   a. What type (please give detailed information)? _____

_____

Remarks of questioning physician:

Pedigree:

APPENDIX B

LISTING OF COMPUTER PROGRAMS

```
ø.ø1 BDC:BDA - DRIVER FOR BIRTH DEFECTS CLINIC

1.1ø K  S %E=-.ø1 T !#,?15,"BIRTH DEFECTS CLINIC",!!
1.2ø R "ROUTINE DESIRED : ",Z I Z="" H
1.25 S I=1ØØ I Z="?" D 2 T !!! K  G 1.2
1.3ø S I=$N(I) I I=ø T !Z," IS NOT AN AVAILABLE ROUTINE.",!!
G 1.2
1.35 S S=$S(I) I '$E(S,1,$L(Z))=Z G 1.3
1.4ø D "1.45 C ".$P(S/2) T $E(S,$L(Z)+1,$F(S,"/")-2),!!
1.45 C XXX G 1.1

2.1ø T !!"AVAILABLE ROUTINES:",!!
2.2ø S I=$N(I) I I=ø Q
2.3ø S S=$S(I) T $P(S/1)
2.4ø T ! G 2.2

99.65 D ↑K(99)
99.8ø E 99.1

1ø1.1ø NEW PATIENT/BDB

1ø2.1ø CORRECT PATIENT FILE/CPF

1ø3.1ø PATIENT DATA SUMMARY/BDS

1ø4.1ø DATA SEASCH/DTS

1ø6.1ø INTEGRITY OF DATA BASE/IDB

1ø7.1ø LIST PATIENTS/LST

1ø8.1ø RELOCATE AND DELETE PATIENT/MDP

ø.ø1 BDC:BDB - NEW PATIENT DATA

1.ø1 S OA=1ø.ø2 C ICK
1.1ø T !!"INITIALS: ",U(1),!,"C-NUMBER: ",U(2)
1.12 T !"K-NUMBER: ",U(3),!,"SEX: ",U(4)
1.14 R !!"**IS THE ABOVE THE CORRECT INFORMATION?**  ",RES
1.16 I ("RES="YES")&('RES="NO") T !"**YES OR NO**" G 1.14
1.18 I RES="NO" G 1.ø1
```

```
1.2Ø K RES O ↑BDC
1.7Ø I $D(↑BDC(NAM,M))=Ø S ↑BDC(NAM,M,1)=U(1) G 1.85
1.75 T !!,U(2)," HAS ALREADY BEEN USED AS C-NUMBER" Q
1.85 S ↑(P+.1)=U(2),↑(P+.2)=U(3),↑(P+.3)=U(4),SX=$B(U(4)="M",1,1,2)
C
1.9Ø S H=5,CF=Ø I $D(ON)=Ø C HIN
1.95 Q

99.65 D ↑K(99)
99.8Ø E 99.1

Ø.Ø1 BDC:ICK - INPUT PATIENT ID
Ø.Ø3 " : "

1.Ø2 K I,Z,S S F=1,NAM=Ø,P=1
1.4Ø S V=$S(OA) F I=1:1 U $P(V,I)="" D 2 S U(I)=A K A
1.6Ø K I,J,K,V S X=$X(U(1),1,3) F I=1,2,3 D 4 S NAM=NAM+%X
1.65 S M=$M(U(2)/"1ØØ"),M=$V(M) Q

2.Ø5 F J=1:1 U $N(2.15)>2.3 E $N(2.15)
2.1Ø D "2.2 T !$P(V,I),".$S(.Ø3)." R A"
2.13 I '$N(1Ø.Ø2+(I-1)/1Ø)=1Ø+I/1Ø G 2.2
2.15 D "2.3 I ".$S(1Ø+I/1Ø)." T $S(1Ø.Ø1+I/1Ø) G 2.2"
2.2Ø T !$P(V,I)-" : " R A

4.1Ø S %X=$X(X,I),%X=($F("ABCDEFGHIJKLMNOPQRSTUVWXYZ",%X)-1)/1ØØ
4.2Ø I I<3 S %X=%X*26 I I<2 S %X=%X*26

1Ø.Ø1 INITIALS,C#
1Ø.Ø2 INITIALS,C#,K#,SEX
1Ø.1Ø 'A:2A&'A:3A
1Ø.11 ??? 2 OR 3 ALPHABETIC CHARACTERS PLEASE
1Ø.2Ø 'A:N!(A:5N&A]"32767")
1Ø.21 ??? NUMERALS ONLY PLEASE - NOT GREATER THAN 32767

99.65 D ↑K(99)
99.8Ø E 99.1



Ø.Ø1 BDC:HIN - HISTORY INTERVIEW

1.1Ø K V S %F=63 I CF=1 G 2
1.2Ø S Q=$H(↑DQ(H,Ø)) G 2

2.Ø1 T !!! I $D(FRM)>Ø T #
2.Ø2 I $D(V(.1))=Ø S V(.1)=Ø
2.1Ø I $G(↑DQ(H,Q))=Ø T "END OF INTERVIEW"!! D 4 G 6
```

```
2.2Ø S Y=↑(Q) T Q,") " D 4
2.21 F K=1:.Ø1 U $G(↑DQ(H,Q,K))=Ø S N=↑(K) T N I '$E(N,$L(N))="
" T " "
2.3Ø I Y>2 T !! F K=1:1:9 T "   ",K,". " D 3 T ! I $G(↑(6.Ø1+K/1ØØ))=Ø
Q
2.4Ø S N=1 R !!!"> ",V(Q),"  " I V(Q)="↑" I V(.1)>Ø S Q=V(.1)
K V G 2
2.41 I V(Q)="STOP" S Q=Ø K V G 2
2.42 I Q=2 S MBD=$E(V(Q),$L(V(Q))-1,$L(V(Q))) I 'MBD:1NN S MBD=""
2.43 I Q=8 S BBD=$E(V(Q),$L(V(Q))-1,$L(V(Q))) I 'BBD:1NN S BBD=""
2.44 I Q=8&CF=Ø I 'MBD=""&'BBD="" S MBD=$M(BBD-MBD) K BBD G
2.5
2.45 I Q=8&CF=Ø S MBD="Ø"
2.5Ø I 'V(Q):"**"N G 1Ø+Y
2.51 S J=$E(V(Q),3,99),J=$V(J) I J>Ø&J<3Ø1 I $G(↑DQ(H,J))>Ø
S Q=J K V G 2
2.52 T J,"??" S J=↑DQ(H,Q,4) G 2.4
2.6Ø I CF=1 D 2.Ø2 D 4 Q
2.65 S Q=↑(4),Q=$P(Q,N) I Q["/" S Q=$P(Q/SX)
2.8Ø S Q=$V(Q) I Q>=Ø G 2
2.9Ø D 4 S H=Ø-Q G 1.1

3.1Ø I Y=3 T $S(1ØØ+K/1Ø) Q
3.2Ø I $G(↑(2+K/1ØØ))>Ø T ↑(2+K/1ØØ) Q
3.3Ø T "NONE OF THE ABOVE"

4.1Ø S J=$H(V(.1)),V(.1)=J I J=Ø Q
4.2Ø F J=J:.Ø1 U $D(V(J))=Ø O ↑BDC S ↑BDC(NAM,M,P,J)=V(J) D
5

5.1Ø I 'J=8!CF=1 G 5.3
5.2Ø I 'MBD="Ø" S ↑(275)=MBD K MBD
5.3Ø S RES=V(J) K V(J) C

6.1Ø T "INTEGRITY CHECK FOR DATA JUST ENTERED",!
6.2Ø S CBC=1,SMR=Ø C HSM Q

11.Ø5 I CF=1 S QO=Q
11.1Ø I 'V(Q)="" S Q=Q+.Ø1 R !"> ",V(Q) G $L
11.2Ø I '$I(Q)=Q K V(Q)
11.25 I CF=1 S Q=QO
11.3Ø G 2.6

12.1Ø I V(Q):1NN G 2.6
12.2Ø T "**ANSWER WITH A NUMBER, PLEASE**" G 2.4

13.1Ø I V(Q):1N S N=$V(V(Q)) I N>Ø&N<=K T "(",$S(1ØØ+N/1Ø),")"
G 2.6
13.2Ø T "**ANSWER WITH " F J=1:1:K-2 T J,", "
13.3Ø T K-1," OR ",K,"**" G 2.4
```

```
14.1Ø I V(Q):1N S N=$V(V(Q)) I N>Ø&N<=K T "(",↑(2+N/1ØØ),")"
G 2.6
14.2Ø G 13.2

15.Ø1 S Z=V(Q),V(Q)="",J=Ø I 'Z:R G 15.3
15.1Ø S J=J+1,S=$E(Z,J),N=$V(S) I J $L(Z) G 15.4
15.15 G $B(N>Ø&N<=K,15.2,S:P,15.1,1,15.3)
15.2Ø S V(Q)=V(Q).S G 15.1
15.3Ø T "**CHOOSE ANSWERS FROM 1 TO ",K,"**" G 2.4
15.4Ø S N=1,S=$C(K) I $L(V(Q))>1 I V(Q)[S T "INCONSISTENT ANSWER!"
G2.4
15.5Ø G $B($L(V(Q)),2.6,1,15.3)

99.65 D ↑K(99)
99.8Ø E 99.1

1ØØ.1Ø YES
1ØØ.2Ø NO
1ØØ.3Ø NO DATA
1ØØ.4Ø DATA NOT AVAILABLE
1ØØ.5Ø RATHER NOT ANSWER



Ø.Ø1 BDC:CPF - CORRECT PATIENT FILE

1.Ø3 S OA=1Ø.Ø1,H=5,CF=1 C ICK
1.Ø5 I $G(↑BDC(NAM,M))=Ø G 7
1.Ø9 T !!!!,"INITIALS: ",↑BDC(NAM,M,1),!,"C#: ",↑(1.1Ø)
1.1Ø T !,"K#: ",↑(1.2Ø),!,"SEX: ",↑(1.3Ø)
1.11 S SX=↑(1.3Ø),SX=$B(SX="M",1,1,2)
1.2Ø S %E=-.Ø1,%F=61,Q=$H(↑BDC(NAM,M,P,Ø)),MOM=Ø G 2

2.1Ø K V I Q=274 G 9
2.2Ø F K=Q:.Ø1 U $G(↑BDC(NAM,M,P,K))=Ø S V(K)=↑(K)
2.3Ø S Y=↑DQ(H,Q),J=1,S=Ø,N=$G(↑(Q,5)) F Z=5:.Ø1 U $G(↑(Z))=Ø
D 3
2.4Ø I S=Ø D 1Ø+Y T " "
2.6Ø R !"**DO YOU WANT TO CHANGE THIS RESPONSE?** ",RES
2.61 I RES="STOP" G 9
2.62 I ('RES="YES")&('RES="NO") T !"**YES OR NO**" G 2.6Ø
2.64 I RES="NO" D 8 G 2
2.65 O ↑BDC F K=Q:.Ø1 U $D(↑BDC(NAM,M,P,K))=Ø K ↑(K)
2.67 C  C HIN S J=1 I Q=2!Q=8 S MOM=1
2.68 I ↑DQ(H,Q)=1!↑(Q)=5 D 8 G 2
2.7Ø D 8.1Ø,8.3Ø S QU=$P(QU,3),QU=$V(QU)
2.72 I Q=QU G 2
2.74 I (QO=5!QO-19!QO=2Ø)&(RES="1"!RES="2") G 5
```

```
2.76 I RES="1" G 5
2.78 I $D(↑BDC(NAM,M,P,Q))=Ø S Q=QU G 2
2.8Ø O ↑BDC K ↑BDC(NAM,M,P,Q) C
2.82 D 5.2Ø I Q=QU G 2
2.84 G 2.8Ø

3.1Ø S W=↑(Z) F N=4.1:.1:4.3 D 4.9
3.11 S N=Ø,D=" "
3.2Ø S N=N+1 I N>$L(W) S D=$B(D=" "!D="!",Ø,1,1) T $E(" ",1,D)
Q
3.3Ø S D=$E(W,N) I D="↑" D 1Ø+Y S S=1 G 3.2
3.4Ø I D="!" T ! G 3.2
3.5Ø T D G 3.2

4.1Ø HE/SHE
4.2Ø HIM/HER
4.3Ø HIS/HER
4.9Ø S D=$S(N) I W[D S W=$E(W,1,$F(W,D)-1-$L(D)).$P(D/SX).$E(W,
$F(W,D),99)

5.1Ø C HIN
5.2Ø S Q=↑DQ(H,Q,4),Q=$P(Q,1),Q=$V(Q)
5.3Ø I Q=QU G 2
5.4Ø G 5.1Ø

7.Ø1 T !!!!,"DATA FOR PATIENT NOT FOUND",!!!! Q

8.1Ø S QO=Q,Q=↑DQ(H,Q,4),QU=Q,Q=$P(Q,J)
8.2Ø I Q'["/" S Q=$P(Q/SX)
8.3Ø S Q=$V(Q)

9.1Ø I MOM=Ø Q
9.15 I $D(↑BDC(NAM,M,P,2))=Ø!$D(↑(8))=Ø G 9.7Ø
9.2Ø S MM=↑(2),BB=↑(8)
9.3Ø S MBD=$E(MM,$L(MM)-1,$L(MM)) I 'MBD:1NN S MBD=""
9.4Ø S BBD=$E(BB,$L(BB)-1,$L(BB)) I 'BBD:1NN S BBD=""
9.5Ø I 'MBD=""&'BBD="" S MBD=$M(BBD-MBD) K BBD G 9.9Ø
9.7Ø I $D(↑(275))=Ø Q
9.8Ø O ↑BDC K ↑(275) Q
9.9Ø O ↑BDC S ↑BDC(NAM,M,P,275)=MBD K MBD Q

11.1Ø F K=Q:.Ø1 U $D(V(K))=Ø T V(K)," "

12.1Ø T V(Q)

13.1Ø S J=$V(V(Q)) D 2Ø

14.1Ø G 13
```

```
15.1Ø S L=$L(V(Q)),K=Ø
15.2Ø S K=K+1,J=$E(V(Q),K),J=$V(J) D 2Ø I K=L S J=1 Q
15.3Ø S J=$B(L=1,3,K=L-1,2,1,1) T $P(", / AND "/J) G 15.2

2Ø.1Ø I '↑(6+J/1ØØ)="↑"!$G(↑(2+J/1ØØ))=Ø T ↑(6+J/1ØØ) Q
2Ø.2Ø T ↑(2+J/1ØØ)

99.65 D ↑K(99)
99.8Ø E 99.1



Ø.Ø1 BDC:BDS - PATIENT DATA SUMMARY

1.Ø2 S OA=1Ø.Ø1 C ICK
1.9Ø S H=5,SMR=1,CBC=1 C HSM

99.65 D ↑K(99)
99.8Ø E 99.1



Ø.Ø1 BDC:HSM - SUMMARY;INTEGRITY CHECK

1.Ø3 I SMR=Ø&CBC=Ø G 8
1.Ø5 O ↑BDC I $G(↑BDC(NAM,M))=Ø G 7
1.Ø9 T !!!!,"INITIALS: ",↑BDC(NAM,M,1),!,"C#: ",↑(1.1Ø)
1.1Ø T !,"K#: ",↑(1.2Ø),!,"SEX: ",↑(1.3Ø)
1.11 S SX=↑(1.3Ø),SX=$B(SX="M",1,1,2)
1.14 I CBC=1&SMR=Ø T !!!,"QUESTIONS HAVING NO DATA OR NULL RESPONSE",!!
1.2Ø S %E=-.Ø1,%F=61,Q=$H(↑BDC(NAM,M,P,Ø))
1.22 G 2

2.1Ø K V I $G(↑BDC(NAM,M,P,Q))=Ø G 9
2.2Ø F K=Q:.Ø1 U $G(↑(K))=Ø S V(K)=↑(K)
2.21 O ↑DQ S Y=↑DQ(H,Q),NUL=Ø,XX=↑(Q+.1) I 'SMR=Ø G 2.3Ø
2.22 I (Q=75&V(Q)="3")!(Q=3&V(Q)="5")!(Q=77&V(Q)="4") S NUL=1
G 2.91
2.23 I (Y=1&V(Q)="")!(Y=3&V(Q)="3") S NUL=1 G 2.91
2.24 I (Y=4&V(Q)="4")!(Y=5&V(Q)="5") S NUL=1 G 2.91
2.25 I CBC=Ø G 2.93
2.26 S J=$B(Y=3,$V(V(Q)),Y=4,$V(V(Q)),1,1) G 2.6Ø
2.3Ø S J=1,S=Ø,N=$G(↑(Q,5)) F Z=5:.Ø1 U $G(↑(Z))=Ø D 3
2.4Ø I S=Ø D 1Ø+Y T " "
2.6Ø C  S Q=↑DQ(H,Q,4),Q=$P(Q,J) I Q["/" S Q=$P(Q/SX)
2.8Ø S Q=$V(Q) I Q>=Ø G 2
2.9Ø S H=Ø-Q G 1.2
```

```
2.91 G $B(CBC=1,2.92,1,2.93)
2.92 C ICC G 2.26
2.93 C IEF G 2.26

3.1Ø S W=↑(Z) F N=4.1:.1:4.3 D 4.9
3.11 S N=Ø,D=" "
3.2Ø S N=N+1 I N>$L(W) S D=$B(D=" "!D="!",Ø,1,1) T $E(" ",1,D)
Q
3.3Ø S D=$E(W,N) I D="↑" D 1Ø+Y S S=1 G 3.2
3.4Ø I D="!" T ! G 3.2
3.5Ø T D G 3.2

4.1Ø HE/SHE
4.2Ø HIM/HER
4.3Ø HIS/HER
4.9Ø S D=$S(N) I W[D S W=$E(W,1,$F(W,D)-1-$L(D)).$P(D/SX).$E(W,
$F(W,D),99)

7.Ø1 T !!!!,"DATA FOR PATIENT NOT FOUND",!!!! Q

8.1Ø O ↑BDC S NAM=$H(↑BDC(Ø)),M=$H(↑BDC(NAM,Ø)),P=1
8.3Ø T !!!"QUESTION",?2Ø, "ASKED  NULL RESPONSES  PER CENT NO
DATA"!
8.4Ø S FIN=1 C IEF S FIN=2 G 1.2Ø

9.Ø1 I CBC=1&SMR=Ø&$D(↑(275))=Ø T !!"AGE OF MOTHER AT BIRTH
MISSING"
9.2Ø I 'CBC=Ø T !!!"END"!! Q
9.3Ø S M=$H(↑BDC(NAM,M)) I 'M=Ø G 1.2Ø
9.4Ø S NAM=$H(↑BDC(NAM)) I 'NAM=Ø S M=Ø G 9.3Ø
9.5Ø S FIN=3 C IEF Q

11.1Ø F K=Q:.Ø1 U $D(V(K))=Ø T V(K)," "

12.1Ø T V(Q)

13.1Ø S J=$V(V(Q)) D 2Ø

14.1Ø G 13

15.1Ø S L=$L(V(Q)),K=Ø
15.2Ø S K=K+1,J=$E(V(Q),K),J=$V(J) D 2Ø I K=L S J=1 Q
15.3Ø S J=$B(L=1,3,K=L-1,2,1,1) T $P(", / AND "/J) G 15.2

2Ø.1Ø I '↑(6+J/1ØØ)="↑"!$G(↑(2+J/1ØØ))=Ø T ↑(6+J/1ØØ) Q
2Ø.2Ø T ↑(2+J/1ØØ)

99.65 D ↑K(99)
99.8Ø E 99.1
```

```
Ø.Ø1 BDC:ICC - INTEGRITY CASE BY CASE

1.2Ø S NN=1Ø F I=1Ø:1Ø:3ØØ S NN=NN=.Ø1 I Q<=I Q
1.39 S A=I-1Ø,A=Q-A,X=$S(NN) T !,$P(X;A) Q

1Ø.Ø1 P1,1-1;P1,2-1;P1,3-1;P114;P116A;P1,6B;P1,6C;P111-2;P121-2;P112-2
1Ø.Ø2 P1,3A;P113B;P123B;P1,3C;P1,3D;P2,1;P2,2;P2,3;P2,4;P2,5
1Ø.Ø3 P2,6;P2,7;P2,8;P3,9-1;P3,1A;P3,1B;P3,2;P3,3;P3,4;P3,5
1Ø.Ø4 P3,6;P3,7;P3,8;P3,9-2;P3,1Ø-2;P411ØA;P441ØA;P461ØA;P4,1;P124
1Ø.Ø5 P1,5;P4,4;P4,5;P4,6;P4,BOTH PARENTS;P4,REMARKS;P4,2;P4,3;;P126A
1Ø.Ø6 ;;;;;;;;;P3,5A
1Ø.Ø7 P3,6A;P3,7A;P3,7B;P3,7C;P3,8A;P3,9A-2;P3,1ØA-2;P421ØA;P431ØA;
P451ØA
1Ø.Ø8 P471ØA;P4,3A;P4,3B;P4,3C;P4,4A;P4,4B;P4,5A;P4,5B;P4,6B;P4,
PARENTS-A
1Ø.1Ø ;;;;;;;;;P122-2
1Ø.11 P132-2;P142-2;P152-2;;;;;;;
1Ø.19 ;;;;;;;;;P2,4A
1Ø.2Ø P2,4B;P2,4C;;;;;;;;P2,5A
1Ø.21 P2,5B;P2,5C;;;;;;;;P2,6A
1Ø.22 P2,6B;P2,6C;;;;;;;;P2,7A
1Ø.23 P2,7B;P2,7C;;;;;;;;P2,8A
1Ø.24 P2,8B;P2,8C;;;;;;;;P3,9A-1
1Ø.25 P3,1Ø-1;;;;;;;;;
1Ø.28 ;;;;;;;;;P3,3A
1Ø.29 ;;;;;;;;;P3,4A
1Ø.3Ø P3,4B;;;;;;;;;

99.65 D ↑K(99)
99.8Ø E 99.1



Ø.Ø1 BDC:IEF - INTEGRITY ENTIRE FILE

1.1Ø I FIN=1 D 3 Q
1.2Ø I FIN=2 D 2 Q
1.3Ø I FIN=3 S J=1ØØ D 4 Q

2.2Ø S ASK=$P(XX,1),NAN=$P(XX,2)
2.23 S ASK=$M(ASK+"1") I NUL=1 S NAN=$M(NAN+"1")
2.3Ø S XX=ASK.","_NAN,↑(Q+.1)=XX

3.1Ø O ↑DQ S Q=$H(↑DQ(H,Ø)),Q=Q+.1
3.2Ø S ↑(Q)="Ø,Ø",Q=$H(↑(Q)) I Q S Q=Q+.1 G $L
3.3Ø C
```

```
4.1Ø S J=$N(J) I J=Ø T !!"END OF INTEGRITY CHECK"! Q
4.2Ø S Y=$S(J) F IJ=1:1:15 S Q=$P(Y,IJ) D 5
4.3Ø G 4.1

5.1Ø I Q="" Q
5.2Ø S Q=$V(Q) C ICC S XX=↑DQ(H,Q+.1) D 2.2Ø I ASK="Ø" G 5.35
5.25 S PER=$M(NAN/ASK*"1ØØ") I PER:N"."N G 5.45
5.3Ø T ?22,ASK,?34,NAN,?51,PER Q
5.35 S PER="Ø" G 5.3Ø
5.45 S XX=$P(PER.2),PER=$P(PER.1)
5.5Ø I XX]"5" S PER=$M(PER+"1") G 5.3Ø
5.55 G 5.3Ø

99.65 D ↑K(99)
99.8Ø E 99.1

1Ø1.Ø1 1,2,3,4,4Ø,41,5,5Ø,6,7,8,9,1Ø,1ØØ,1Ø1

1Ø2.Ø1 1Ø2,1Ø3,11,12,13,14,15,16,17,18,19,19Ø,191,192,2Ø

1Ø3.Ø1 2ØØ,2Ø1,2Ø2,21,21Ø,211,212,22,22Ø,221,222,23,23Ø,231,232

1Ø4.Ø1 24,24Ø,241,25,26,27,28,28Ø,29,29Ø,291,3Ø,6Ø,31,61

1Ø5.Ø1 32,62,63,64,33,65,34,66,35,67,36,68,69,37,7Ø

1Ø6.Ø1 38,71,39,47,48,72,73,74,42,75,76,43,77,78,44

1Ø7.Ø1 79,45,8Ø,46,



Ø.Ø1 BDC:IDB - INTEGRITY OF DATA BASE

1.2Ø R "OPTION DESIRED : ",Z I Z="" Q
1.25 S I=1ØØ I Z="?" D 2 T !!! K  G 1.2
1.3Ø S I=$N(I) I I=Ø T !Z," IS NOT AN AVAILABLE OPTION.",!!
G 1.2
1.35 S S=$S(I) I '$E(S,1,$L(Z))=Z G 1.3
1.4Ø S CBC=$P(S/2),CBC=$V(CBC) T $E(S,$L(Z)+1,$F(S,"/")-2),!!
1.45 S OA=1Ø.Ø1,SMR=Ø,H=5 I CBC=1 C ICK
1.5Ø C HSM

2.1Ø T !!"AVAILABLE OPTIONS:",!!
2.2Ø S I=$N(I)!I I=Ø Q
2.3Ø S S=$S(I) T $P(S/1)
2.4Ø T ! G 2.2
```

```
99.65 D ↑K(99)
99.8Ø E 99.1

1Ø1.1Ø CASE BY CASE/1

1Ø2.1Ø ENTIRE FILE/Ø



Ø.Ø1 BDC:LST - LIST PATIENTS

1.1Ø T #!"INITIALS  C-NUMBER  K-NUMBER"
1.2Ø S NAM=$H(↑BDC(Ø)),M=$H(↑BDC(NAM,Ø)),P=1
1.3Ø G 2

2.1Ø T !,?3,↑BDC(NAM,M,P),?13,↑(P+.1),?23,↑(P+.2)
2.2Ø G 3

3.1Ø S P=$H(↑BDC(NAM,M,P+.3)) I 'P=Ø G 2
3.2Ø S P=1,M=$H(↑BDC(NAM,M)) I 'M=Ø G 2
3.3Ø S NAM=$H(↑BDC(NAM)) I 'NAM=Ø S M=Ø G 3.2Ø
3.4Ø T !!"END OF PATIENTS STORED IN BDC GLOBAL" Q

99.65 D ↑K(99)
99.8Ø E 99.1



Ø.Ø1 BDC:MDP - MOVE AND DELETE PATIENT

1.Ø1 T !!"INPUT ID OF PATIENT TO BE DELETED"
1.Ø2 S OA=1Ø.Ø1 C ICK
1.Ø6 S ON=NAM,OM=M
1.Ø8 I $D(↑BDC(NAM,M))=Ø T !!"PATIENT NOT IN FILE" Q
1.2Ø R !"OPTION DESIRED : ",Z I Z="" Q
1.25 S I=1ØØ I Z="?" D 2 T !!! G 1.2
1.3Ø S I=$N(I) I I=Ø T !Z," IS NOT AN AVAILABLE OPTION.",!!
G 1.2
1.35 S S=$S(I) I ' $E(S,1,$L(Z))=Z G 1.3
1.4Ø S PAR=$P(S/2),PAR=$V(PAR) T $E(S,$L(Z)+1,$F(S,"/")-2),!!
1.5Ø G PAR

2.1Ø T !!"AVAILABLE OPTIONS:",!!
2.2Ø S I=$N(I) I I=Ø Q
2.3Ø S S=$S(I) T $P(S/1)
2.4Ø T ! G 2.2
```

```
3.Ø5 T !!"INPUT ID OF NEW PATIENT"
3.1Ø D BDB
3.2Ø S Q=$H(↑BDC(ON,OM,P,Ø))
3.3Ø O ↑BDC S ↑BDC(NAM,M,P,Q)=↑BDC(ON,OM,P,Q) C
3.4Ø S Q=$H(↑BDC(ON,OM,P,Q)) I Q=Ø G 4
3.45 G 3.3Ø

4.1Ø O ↑BDC K ↑BDC(ON,OM)
4.15 S Q=$H(↑BDC(ON,Ø)) I 'Q=Ø K ON,OM G 4.3Ø
4.2Ø K ↑BDC(ON),ON,OM
4.3Ø T !!"END OF RELOCATE AND/OR DELETE"!! Q

99.65 D ↑K(99)
99.8Ø E 99.1

1Ø1.1Ø DELETE ONLY/4

1Ø2.1Ø MOVE AND DELETE/3



Ø.Ø1 BDC:DTS - DATA SEARCH SETUP

1.1Ø S IC=Ø,H=5,%F=63 T #! K V
1.2Ø I IC=5 C SEA Q
1.3Ø R !"TYPE FRAME NUMBER OF ITEM TO BE SEARCH : ",Q
1.32 I Q="" C SEA Q
1.34 I 'Q:N T !,"**MUST BE NUMBER**" G 1.3
1.35 S Q=$V(Q)
1.4Ø I $D(↑DQ(5,Q))=Ø T !!"**INVALID FRAME NUMBER**",! G 1.3
1.5Ø S IC=IC+1,Y=↑DQ(5,Q) G 2

2.1Ø I Y=1 T !"FREE-TEXT ANSWER, TYPE IN SEARCH SPEC.",!
2.15 I Y=3!Y=4!Y=5 T !"CHOOSE SEARCH SPEC. FROM FOLLOWING LIST",!
2.2Ø T Q,"> "
2.21 F K=1:.Ø1 U $G(↑DQ(H,Q,K))=Ø S N=↑(K) T N I '$E(N,$L(N))="
" T " "
2.3Ø I Y>2 T !! F K=1:1:9 T "   ",K,". " D 3 T ! I $G(↑(6.Ø1+K/1ØØ))=Ø
Q
2.4Ø S N=1 R !!!"> ",V(Q),"  " G 1Ø+Y
2.6Ø G 1.2Ø

3.1Ø I Y=3 T $S(1ØØ+K/1Ø) Q
3.2Ø I $G(↑(2+K/1ØØ)) Ø T ↑(2+K/1ØØ) Q
3.3Ø T "NONE OF THE ABOVE"

11.1Ø G 2.6
```

```
12.1Ø I V(Q):1NN G 2.6
12.2Ø T "**ANSWER WITH A NUMBER, PLEASE**" G 2.4

13.1Ø I V(Q):1N S N=$V(V(Q)) I N>Ø&N<=K T "(",$S(1ØØ+N/1Ø),")"
G 2.6
13.2Ø T "**ANSWER WITH " F J=1:1:K-2 T J,", "
13.3Ø T K-1," OR ",K,"**" G 2.4

14.1Ø I V(Q):1N S N=$V(V(Q)) I N>Ø&N<=K T "(",↑(2+N/1ØØ),")"
G 2.6
14.2Ø G 13.2

15.Ø1 S Z=V(Q),V(Q)="",J=Ø I 'Z:R G 15.3
15.1Ø S J=J+1,S=$E(Z,J),N=$V(S) I J>$L(Z) G 15.4
15.15 G $B(N>Ø&N<=K,15.2,S:P,15.1,1,15.3)
15.2Ø S V(Q)=V(Q).S G 15.1
15.3Ø T "**CHOOSE ANSWERS FROM 1 TO ",K,"**" G 2.4
15.4Ø S N=1,S=$C(K) I $L(V(Q))>1 I V(Q)[S T "INCONSISTENT ANSWER!"
G 2.4
15.5Ø G $B($L(V(Q)),2.6,1,15.3)

99.65 D ↑K(99)
99.8Ø E 99.1

1ØØ.1Ø YES
1ØØ.2Ø NO
1ØØ.3Ø NO DATA
1ØØ.4Ø DATA NOT AVAILABLE



Ø.Ø1 BDC:SEA - DATA SEARCH

1.1Ø S Q=.1,V(Q)=Ø I $H(V(Q))=Ø Q
1.2Ø S TOT=Ø,HIT=Ø,ND=Ø,NAM=$H(↑BDC(Ø)),M=$H(↑BDC(NAM,Ø)),P=1
1.3Ø T #!"INITIALS  C-NUMBER  K-NUMBER"
1.4Ø S TOT=TOT+1,Q=.1,NH=Ø
1.5Ø S Q=$H(V(Q)) I Q=Ø G 7
1.57 I $D(↑BDC(NAM,M,P,Q))=Ø G 2.9Ø
1.6Ø S Y=↑DQ(5,Q) G $B(Y=1,2,Y=3,3,Y=4,4,Y=5,5)

2.1Ø I ↑BDC(NAM,M,P,Q)="" G 2.8
2.2Ø S L=Q
2.21 I $D(↑BDC(NAM,M,P,L))=Ø G 8
2.22 I ↑(L)[V(Q) G 1.5Ø
2.23 S L=L+.Ø1 G 2.21
2.8Ø I V(Q)="" G 1.5Ø
2.9Ø S ND=ND+1 G 6
```

```
3.1Ø I ↟BDC(NAM,M,P,Q)="3" G 3.8
3.2Ø I ↑(Q)=V(Q) G 1.5Ø
3.3Ø G 8
3.8Ø I V(Q)="3" G 1.5Ø
3.9Ø S ND=ND+1 G 6

4.1Ø I Q=75 G 3
4.2Ø I Q=3 G 5
4.3Ø I ↟BDC(NAM,M,P,Q)="4" G 4.8
4.4Ø I ↑(Q)=V(Q) G 1.5Ø
4.5Ø G 8
4.8Ø I V(Q)="4" G 1.5Ø
4.9Ø S ND=ND+1 G 6

5.1Ø I Q=77 G 4
5.2Ø I ↟BDC(NAM,M,P,Q)="5" G 5.8
5.3Ø I ↑(Q)=V(Q) G 1.5Ø
5.4Ø G 8
5.8Ø I V(Q)="5" G 1.5Ø
5.9Ø S ND=ND+1 G 6

6.1Ø S M=$H(↑BDC(NAM,M)) I 'M=Ø G 1.4
6.2Ø S NAM=$H(↑BDC(NAM)) I 'NAM=Ø S M=Ø G 6.1Ø
6.3Ø T !!"TOTAL POPULATION : ",TOT
6.4Ø T !!"TOTAL MEETING ALL SPECS. : ",HIT
6.5Ø T !!"TOTAL HAVING NULL RESPONSES : ",ND
6.6Ø Q

7.Ø5 I NH=1 G 6
7.1Ø S HIT=HIT+1
7.2Ø T !,?3,↟BDC(NAM,M,P),?13,↟(P+.1),?23,↑(P+.2)
7.3Ø G 6

8.1Ø S NH=1 G 1.5Ø
```

VITA

Ann Pearson Rule was born in Knoxville, Tennessee on December 31, 1945. She attended elementary school in that city and graduated from Knoxville Central High School in May of 1964. She received a Bachelor of Arts degree in Mathematics from the University of Tennessee in June 1969. After working for Union Carbide at Oak Ridge, Tennessee for three years, she began work toward a Master's degree at the University of Tennessee in the fall of 1972. At the same time, she accepted a teaching assistantship and taught commercial-oriented programming courses at the undergraduate level. In August 1974 she received a Master of Science degree with a major in Computer Science.