A PROTOTYPE TO ILLUSTRATE INTERACTION WITH A PERSONNEL DATABASE

by

HAROON RASHID

B. S., Kansas State University, 1974

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1976

Approved by:

Major Professor

NEXT TO GOD, THY PARENTS

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

The report deals with the implementation of a prototype interactive database management system for the College of Arts and Sciences. The goal of this report is to provide a demonstration system using the interactive language APL, because of the low cost, ease of use and availability.

At present the University Data Processing Center provides the office of the dean with two copies of the annual budget report. One of these copies is kept in the dean's office, while the other one is separated and sent to the respective departments. There are a total of 24 departments in the College of Arts and Sciences. The listing of each department consists of the names of faculty, staff, their salaries, service record, allocated dollars, free dollars, allocated tenths, free tenths, social security number, etc. All changes in this huge list are made by hand and old information is scratched by pencil. The respective changes are made in the dean's copy as well as in the individual department's copy at the same time by two different people and this could lead to serious discrepancies. The errors may occur in both copies and can go undetected for a long period of time. They may be caught at the end of the fiscal year. Tracing of all such errors can be very time consuming and frustrating.

The main idea of this report is to design and implement a prototype interactive system which demonstrates the possibility of maintaining such a database using an interactive system. The system by no means solves all the problems of maintenance faced by the dean's office.

The system has been programmed, tested and tried by several people, program listings are available from the Department of Computer Science. A separate project [4] has been developed in the department which would on production basis more or less solve all the problems faced by the dean's

office. The system was developed using COBOL and IDMS. It is a conventional batch system rather than being interactive. This batch system manipulates and provides more detailed information as compared to the interactive system discussed in this report. Some of the additional information handled by the batch system as opposed to the interactive system is employee's present address, sex of an employee, retirement plan, accumulated service salary, FICA, etc.

CHAPTER II

READER'S GUIDE

The report has been organized essentially as described below. The sections below deal with the general information, recommended for study by the novice user and technical information to be read by computer science personnel and programmers in particular.

1. General Overview Discussion:

   This section gives an overview narrative of the system. The system has been described in very broad terms and should be readable by all novice users.

2. Principles of Interactive Systems - (concepts):

   Prime objectives of interactive systems have been discussed, which all good interactive systems should have. The concepts have been described in general, readable by all novice users.

3. System Evaluation:

   The system has been evaluated regarding its limitations concerning the programmer and the user in general. The user limitations have been discussed from the problem view point (functional limitations).

4. General User Information:

   The system has been described, in terms of the different commands which the user might use, in order to update or interrogate the information in the database. A general flow of the system has been shown, using proper system commands which are executed, to do the work. Workspace organization has also been discussed in general, along with the procedure to correct mistakes and information input procedure.

5. Appendix I - (Execution examples):

   Appendix I includes all the execution examples of the user commands. Each command has been executed a number of times to show the successful

and unsuccessful executions of a single command. Commands have been executed by using long and short names interchangeably.

System commands have been executed and their effect has been shown, at the very end of the Appendix.

1. GENERAL OVERVIEW DISCUSSION

The prototype demonstrates interactive data maintenance. Figures 1 and 2 show an overview of this system as well as the related conventional batch system.

This prototype is designed to meet the demand for fast and easy retrieval of information and for maintenance of transactions as compared to a batch system. The system of this type may be a little bit more expensive as far as running and actual usage is concerned. One of the factors which increases the running cost is the keying time, which is not a part of the conventional batch system.* (A batch system is one that collects user queries and maintenance transactions and runs them collectively.) For system users, this may be the potential to produce a long wait for results. In order to reduce wait time and inconvenience, the interactive prototype was designed. The interactive system eliminates the wait for a batch to be processed. A query is submitted directly to the system through an online terminal. Searching, deletions or maintenance can begin immediately. The user can display information, add an entry or delete an entry at any convenient time.

*Another factor is the higher cost for CPU time, because of the CPU rate, which is higher for high priority users, such as interactive users.

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

AN OVERVIEW OF THE SYSTEM USING APL

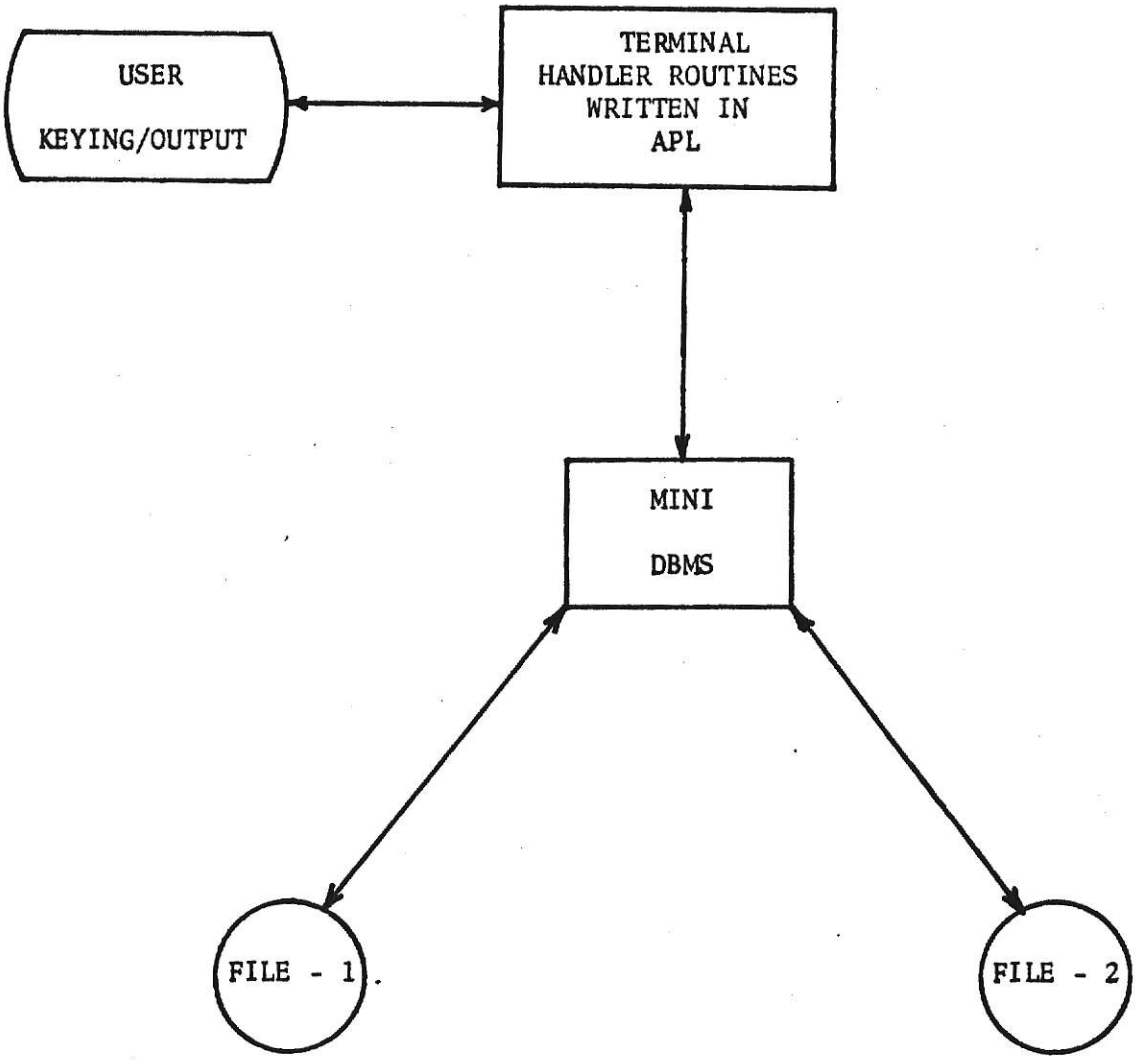


Figure 1

AN OVERVIEW OF THE CONVENTIONAL BATCH SYSTEM

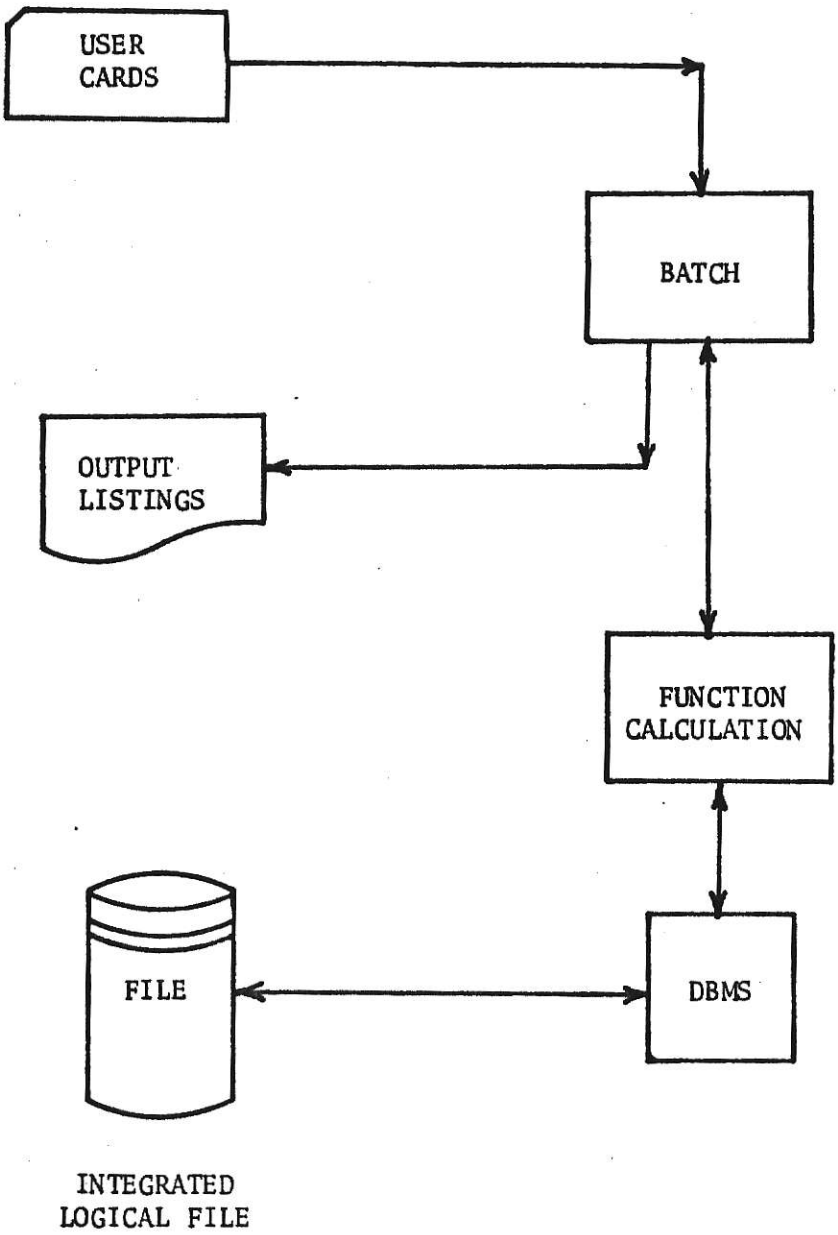


Figure 2

2. PRINCIPLES OF INTERACTIVE SYSTEMS

Some of the major principles for interactive systems are summarized below; these are the general principles which such systems should achieve (although not all interactive systems have these features).

i. No Catastrophic Errors:

The user in an interactive system should have no chances to bring the whole system down. Of course, this also is a general principle for design of good programs.

ii. Error Messages:

If an error is made, an understandable message should be output by the system, indicating exactly where the error occurred so that corrections may be made right at that time.

iii. DESCRIBE function:

The system should be provided with a general information function DESCRIBE. The function should provide vital information regarding different commands and their short names.

iv. Try Again:

If an error message is reported to the user, the system should allow the user to rectify the error and continue without restarting the whole computation.

v. Recovery:

If, by accident, some needed and useful information is deleted, then recovery should be made easier by printing all information before deletion or by requesting confirmation before deleting large quantities of data.

vi. Prompting:

The user when working with an interactive system should always be prompted by the system response.

vii. Order of Action:

The sequence of action of the user is not fixed; he can enter any commands as he wishes depending upon the task to be performed.

viii. Minimum Response Time:

The system should have a low response time. Otherwise, a periodic echo check of a standard waiting message should be made so as to keep the user prompted.

ix. Short Names:

Highly interactive systems should have relatively short names for the user commands, else the keying time can become very significant.

x. Cost & Usage Measures:

The user should have some measures regarding the amount of core being used, cost of execution, connect time, etc. Such information can be gathered by the use of I-beam functions, (provided by APL/360) [1], [2].

3. SYSTEM EVALUATION

The prototype has been evaluated in terms of shortcomings which do exist, due to serious limitations as imposed by this version of APL/360 at KSU. These limitations have been categorized and discussed separately. Some unique advantages which APL/360 offers as an interactive and array oriented language have also been discussed.

Limitations: There are 3 types of limitations, the functional limitations, external limitations, and internal limitations.

Functional Limitations: The prototype does not implement all the maintenance facilities, but rather points out the different advantages, which interactive systems of such types can offer. The system in particular does not implement extensive communication between the Employee record and Department record structure. The system also does not keep track of the source of dollars, detail about employees' retirement plan, FICA paid by the employee, type of fund, etc., in contrast to conventional batch system.

External Limitations:

i. APL/360 has no easy formatting facilities, FMT function is available but it is too slow and expensive; major formatting problems were solved by using blank character strings and integer numbers.

ii. Fractional and whole numbers cannot be mixed, therefore salaries and other dollar figures are in pennies.

iii. Size of the workspace is a very serious limitation, and is the only reason for using two separate workspaces to store the two structures and associated functions separately.

iv. Limited communication has been implemented between the two workspaces EMPREC and DEPREC, only to demonstrate the possibility of communication.

Internal Limitations:

i. APL is an array oriented language, therefore structures as shown in the Database Structures section are not used, rather a character matrix was manipulated along with a numeric matrix to obtain the structures as shown.

ii. String entities are not available, to get around this problem we have character vectors which are restructured, so as to look like strings.

iii. APL/360 is a highly functional language. Some programs are hard to read and understand.

Advantages: APL/360 has some unique advantages as a high level interactive language.

i. APL/360 being a functional language provides its users with assorted built-in functions, which greatly reduce the size of the programs, of course at the cost of readability of the programs.

ii. Extensive edit facilities are available to the user entering information from the terminal.

iii. Signing-on for APL/360 is very simple as shown in SIGN-ON and SIGN-OFF example in Appendix I.

iv. Development costs as compared to the batch system were drastically lower; for figures refer to Conclusions.

4. GENERAL USER INFORMATION

It is assumed that in a system as this, the user is not expected to know all the technical details of the system. His requests are simple and implemented through a small number of simple commands.

The system is composed of two files or individual workspaces; each workspace has its own groupings of records according to some predefined problem specification. All records in each of these workspaces are stored in alphabetic order. Workspaces have their own functions which manipulate the associated data. The name of each function is a simple command, which a user can use according to his needs. The names of all the commands are listed in the USER MANUAL (Appendix I) and in Tables 1 and 2 (pages 19-20).

Almost all user commands have two separate names for activating them. A long name usually relates to the process performed by the command, for example

PRINTALL	Prints all the entries in the structure
ADD	Adds new entries to the structure

Short names are just the shortened forms of long names: for the above two cases we have PRTALL and ADD. DESCRIBE function lists out all the command names and their short forms.

Use of Commands and Input of Data

In order to use any command, the user must type the long or short name of the command (as given by the DESCRIBE function). On typing the name, the system would respond back by putting out a message indicating to the user the type of information needed for further execution.

Any time numeric input is expected from the user, a symbol □: is output by the system; in case of character input the system merely waits

for the response. In case of simple inputs no headings are put out for user help. However, in case of more complex user responses, proper labels are output before waiting for the response. This helps the user in inputting the correct response. The user, after typing in the correct response, must hit the RETURN key, in order to indicate the system that input sequence has been completed and further execution of the command may be carried on. For detailed examples on use of commands and input of data refer to Appendix I.

How to Correct Typing Errors

System APL/360 provides the users with a very handy correction facility, to correct all mistakes made while keying. If at any point during input, before a RETURN key is hit, you recognize a wrong character(s), then backup with the help of BACK SPACE key on the keyboard to the point where you made the mistake. Now hit the ATTENTION key, wait for the attention character to print, then resume your normal keying from this point of correction.

General Workspace Information and Sequence Flow

There are two information structures namely Employee record and Department record. Both these information structures reside in different workspace areas. The nature of information stored in Employee record workspace (EMPREC) is indirectly related to the Department record workspace (DEPREC). Therefore each time we update the employee information, we have to keep track of all changes and at the conclusion of several employee updates must update the Department record entries, to be consistent with the updated Employee record information.

Each workspace has a maximum capacity of 32K bytes of storage, which is a system APL/360 implementation restriction of this version at KSU.

The two information structures collectively with their associated functions take up 44 K bytes (assuming each structure not having more than 12 entries). The above figures dictate that two separate workspaces must be used in order to fit everything in the restricted workspace and at the same time leave ample space to store intermediate results at execution time. (This problem would be eliminated by the installation of the new version of APL in the coming Fall).

Example SEQUENCE

The flowchart of Table 1 illustrates the general sequence flow which a novice user probably would follow in order to do certain updates or queries concerning employee or department records.

A stepwise explanation of the flowchart has been given below. The user signs-on to the system as shown in the SIGN-ON procedure in Appendix I using his APL account number. If the user is using the system for the first time, then it becomes necessary to go through the initialization procedure, which initializes the database. Before executing the INITIALIZE function all functions must be loaded into the active workspace, from the Employee record workspace (EMPREC). In case of old users, after loading the Employee record workspace ")LOAD EMPREC" the INITIALIZE function must not be executed.

Before performing any updates, department record information is copied into the active workspace by using the system command ")COPY DEPREC". Now we are ready to perform employee updates; a number of updates or queries can be performed; on completion of all updates and queries we execute the UPDATE function to make the department information consistent with the updated employee information.

After we are finished with all updates concerning employee records, we save the Employee record information by the use of a system command

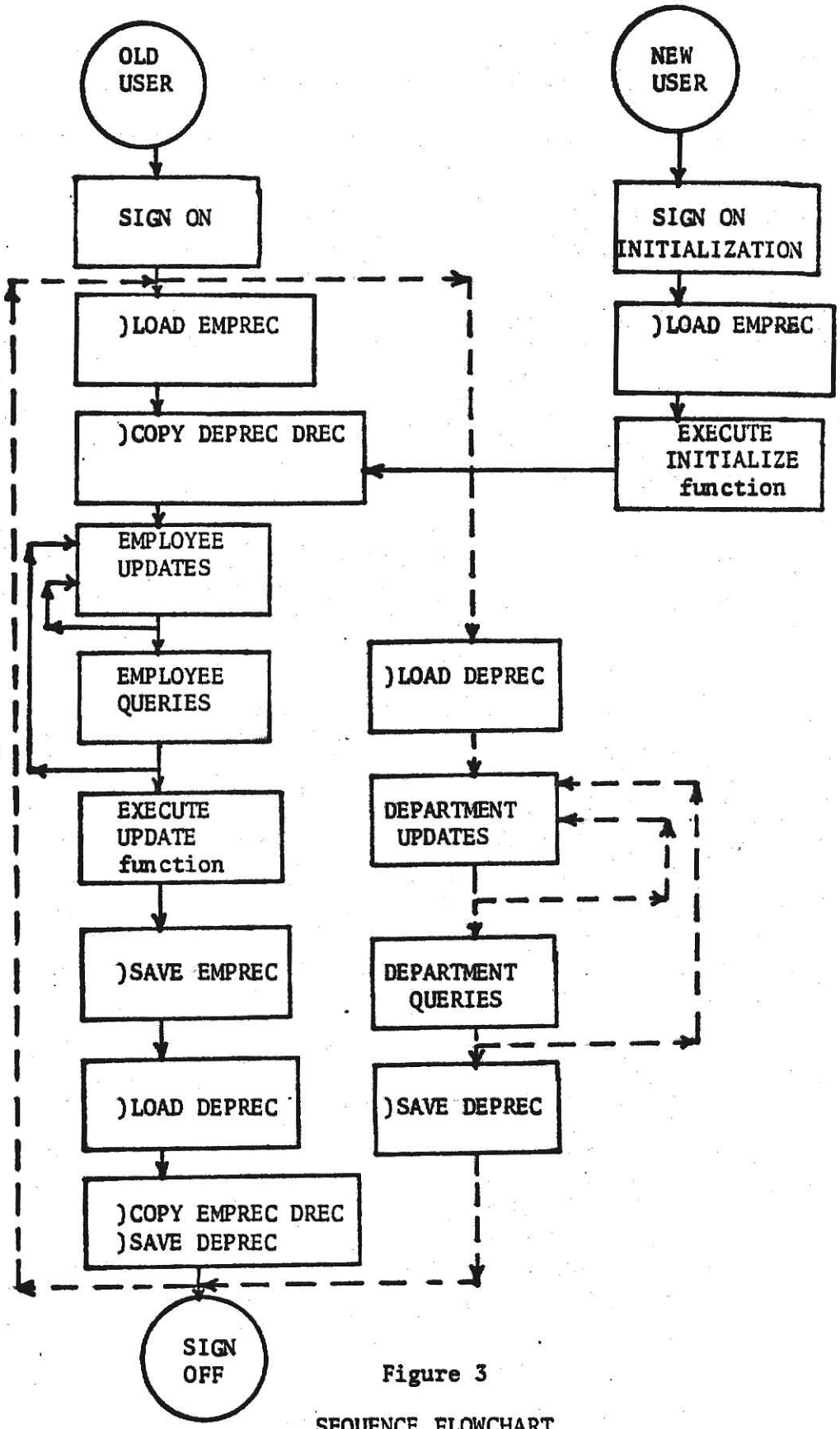


Figure 3

SEQUENCE FLOWCHART

")SAVE EMPREC."

In order to transfer the updated file (DREC) back to the Department record workspace we load the active workspace with the Department record workspace ")LOAD DEPREC." Now, to transfer the updated information into this workspace we execute a COPY command ")COPY EMPREC DREC." Note if this copy operation is not performed the Department record information would not be consistent with the current employee records, though an updated copy would exist in the EMPREC workspace.

At this point in time, we can sign-off as shown in Appendix I or we can follow the dotted path of flowchart Figure 3 and repeat the above cycle again or load the Department record workspace, perform updates and queries, save the workspace and sign-off, or reiterate through the whole process.

LIST OF EMPLOYEE RECORD FUNCTIONS

NO	FUNCTION	NAME
1	DESCRIBE	
2	PRINTALL	
3	TOTALS	QUERY
4	PRINTSEL	
5	PRINTLINE	
6	CODES	
7	DELETE	
8	INITIALIZE	
9	UPDATE	DATA BASE UPDATE
10	ADD	
11	ALTER	
12	INSERT	
13	CHANGE	
14	HEADING	INTERNAL
15	ARRANGE	
16	DATE	

Table 1

LIST OF DEPARTMENT RECORD FUNCTIONS

NO	FUNCTION	NAME
1	DESCRIBE	QUERY
2	DPRINTALL	
3	DTOTALS	
4	DPRINTSEL	
5	DPRINTLINE	
6	DCODES	
7	DCHANGE	DATA BASE UPDATE
8	DDELETE	
9	DINITIALIZE	
10	HEADING	INTERNAL
11	ARRANGE	

Table 2

CHAPTER III

DATABASE STRUCTURES

STRUCTURES:

The whole information has been divided into two main structures, namely

i. Employee Record Structure

ii. Department Record Structure.

Each of the above structures have been diagrammed to give a better understanding of the information structures.

A couple of entries have been shown in each of these structures to show how the coded information looks.

SYSTEM INFORMATION:

The data structures store all information concerning an employee or a department. Figures 4 and 5 show the physical appearance of the two respective data structures. Each field of the structure has been described on the following pages.

EMPLOYEE RECORD:

In actuality the information structure is composed of two matrices, one of them being the character matrix, which holds all the employee names, while the other is the integer number matrix holding all the numeric information.

EMPLOYEE RECORD STRUCTURE WITH SAMPLE ENTRIES

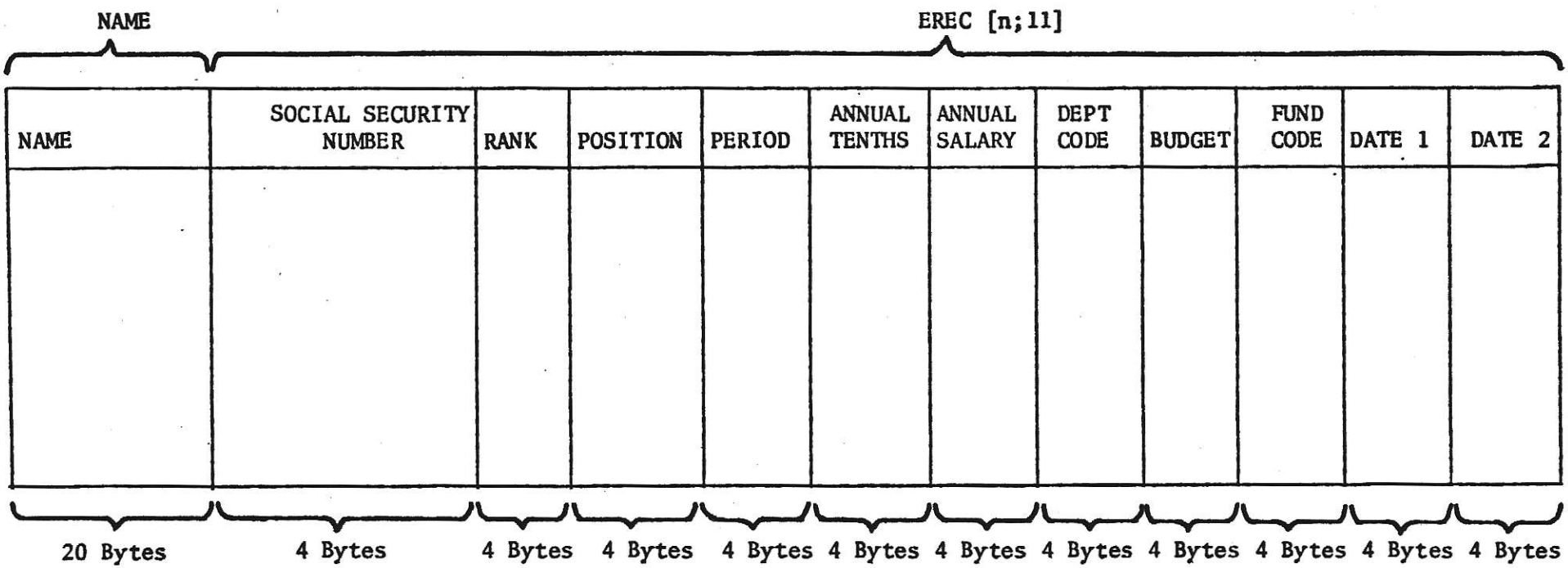
NAME	SOCIAL SECURITY NUMBER	RANK	POSITION	PERIOD	ANNUAL TENTHS	ANNUAL SALARY	DEPT CODE	BUDGET	FUND CODE	DATE 1	DATE 2
BURNS, CHERYL	515629778	4	1	9	6	400000	12	300000	43521	10175	90175
DOE, JOHN	515668888	6	1	12	9	1247567	18	1247567	43321	120175	120176
COLUMN NUMBERS	1	2	3	4	5	6	7	8	9	10	11

Figure 4

Description of Fields of Employee Record

1.	Name	Name of the employee
2.	Social Security Number	Social Security Number of the employee
3.	Rank	One digit specifying the rank of the employee
4.	Position	Classified = 1, Unclassified = 2
5.	Period	Salary spread period (9 or 12 months)
6.	Annual Tenths	Number of tenths of appointment.
7.	Annual Salary	Salary Total annual rate in pennies
8.	Dept Code	Code of the employer department
9.	Budget	Annual Budget for the Employee
10.	Fund Code	Not used
11.	Date 1	Beginning date
12.	Date 2	Contract termination date

BYTE STRUCTURE OF EMPLOYEE RECORD



One Employee Entry = 64 Bytes

Maximum Number of Entries = 200

Name & EREC are Two Separate Matrices

Manipulated by Employee Record Functions

Figure 5

DEPARTMENT RECORD:

The information structure stores all related information concerning a certain department which is in the College of Arts and Sciences.

In actuality, the structure consists of two matrices one of them being the character matrix, which holds all the different names of departments, while the other is the integer matrix holding all the numeric information concerning the departments.

DEPARTMENT RECORD STRUCTURE WITH SAMPLE ENTRIES

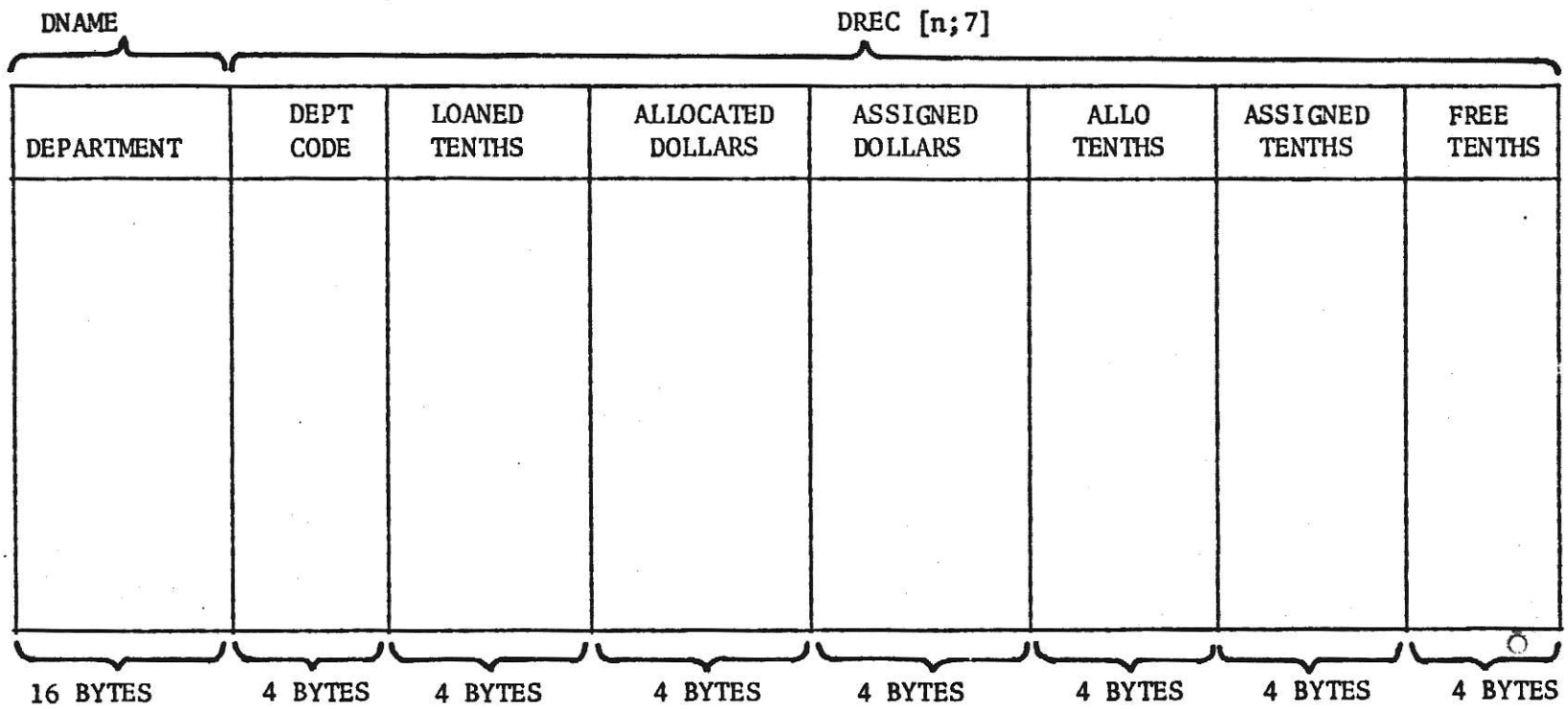
DEPARTMENT NAME	DEPT CODE	LOANED TENTHS	ALLOCATED DOLLARS	ASSIGNED DOLLARS	ALLO TENTHS	ASSIGNED TENTHS	FREE TENTHS
ANTHROPOLOGY	88	0	4454677	3467888	123	116	7
BIOLOGY	21	0	5667200	4632460	124	103	21
COLUMN NUMBERS	1	2	3	4	5	6	7

Figure 6

Description of Department Record Fields

1.	Department Name	Name of the Department
2.	Department Code	2 digit unique code which identifies the department
3.	Loaned Tenths	Tenths loaned out by the department
4.	Allocated Dollars	Total annual allocation of money in pennies
5.	Assigned Dollars	Total annual assigned money in pennies
6.	Allocated Tenths	Total allocated tenths
7.	Assigned Tenths	Total assigned tenths
8.	Free Tenths	Unused tenths within a department

BYTE STRUCTURE OF DEPARTMENT RECORD



One Department Record Entry = 44 Bytes

Maximum Number of Entries = 300

DNAME & DREC are Two Separate Matrices

Manipulated by Department Record Functions

Figure 7

AUXI:

The information structure stores all related information concerning department updates which become necessary during the process of Employee record updates and additions. The structure is common to the two workspaces and serves as the basis for communication.

The AUXI structure consists of a n x 3 matrix as shown on the next page. It contains only numeric information concerning a department, regarding changes in assigned tenths and dollars.

AUXI STRUCTURE WITH SAMPLE ENTRIES

DEPARTMENT CODE	CHANGE IN ASSIG. TENTHS	CHANGE IN ASSIG. DOLLARS
18	-4	-416500
6	-2	-221300
9	0	-14465

BYTE STRUCTURE OF AUXI

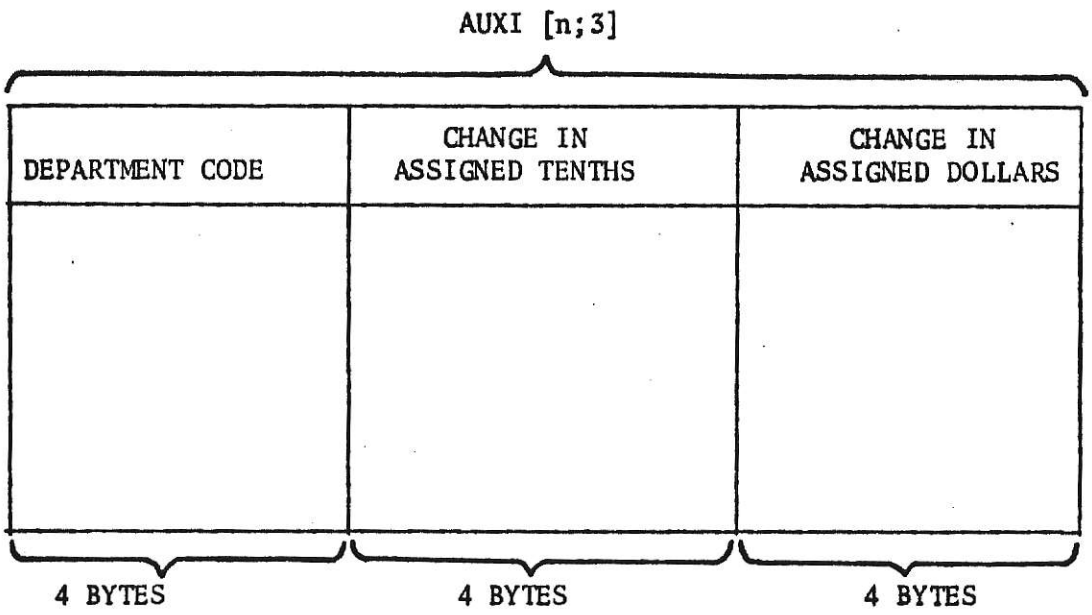


Figure 8

Description of AUXI Fields

1.	Department code	Numeric code identifying the department uniquely
2.	Change in assigned tenths	Increase or decrease in the assigned tenths figure of a department
3.	Change in assigned dollars	Increase or decrease in the assigned money figure of a department, in pennies

CHAPTER IV

DESCRIPTION OF EMPLOYEE RECORD FUNCTIONS

LIST OF EMPLOYEE RECORD FUNCTIONS

All functions have been described in the same manner. Information has been divided into different groups.

USER INFORMATION groups all the information which a novice user may need. Calling form is the name or names by which the command may be called. User response lists the first typical response which the function may expect from the user; all the responses have not been listed due to the possibility of large combinations. The output response title indicates the expected or unexpected output. The function paragraph defines the basic purpose of the command along with some of the important requirements and characteristics of the function.

PROGRAM INFORMATION groups information important from the programmer's point of view. This information is not recommended for reading by the novice user. Parametric inputs, variable outputs, global variables and sub-functions invoked have been listed.

Figure 9 shows the difference between user inputs and outputs from programming point of view. The column numbers are an important input response during most function executions, which are shown in Figure 4 and Figure 6 for Employee record structure and Department record structure, respectively.

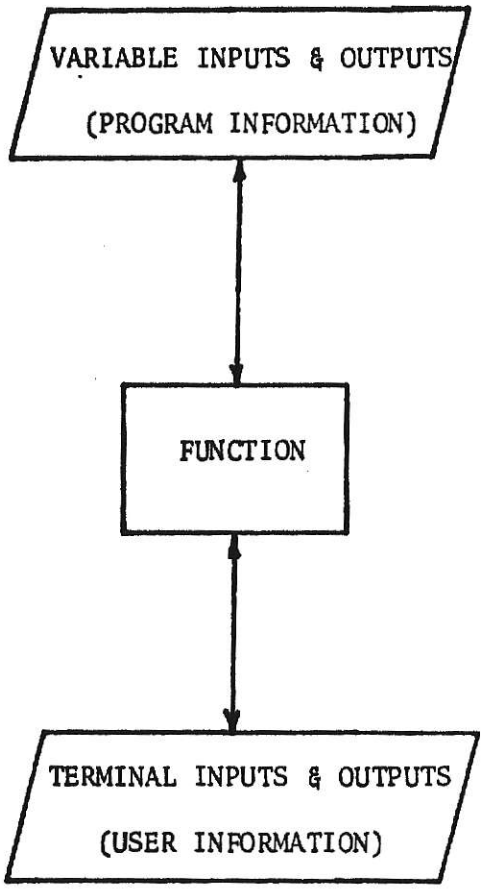

VARIABLE INPUTS & OUTPUTS
(PROGRAM INFORMATION)
FUNCTION
TERMINAL INPUTS & OUTPUTS
(USER INFORMATION)

Figure 9

DESCRIBE

USER INFORMATION:

1) Calling form: DESCRIBE
2) User Response: None
3) Output Response: One line description of each function present within that work space along with the short name.
4) Function: The describe function is a helper function and is designed to help beginner users who need primary information on the type of functions which are at his disposal. One line description of each user funciton is given.

PROGRAM INFORMATION:

1) Inputs: None
2) Outputs: None
3) Global Variables: None
4) Sub-functions Invoked: None

CODES

USER INFORMATION:

1) Calling Form: CODES
2) User Response: None
3) Output Response: Complete explanation of all short names used in labeling the output.
4) Function: The function was implemented to provide the user with general information, regarding labels used in outputs produced by different functions.

PROGRAM INFORMATION:

1) Inputs: None
2) Outputs: None
3) Global Variables: None
4) Sub-functions Invoked: None

PRINTALL

USER INFORMATION:

1) Calling Forms: PRINTALL or PRTALL
2) User Response: Input Column Values between 1 & 11 or type ALL.
3) Output Response: Complete listing of all employee entries, each having the specified columns, as input by the user, or error message indicating wrong column values as input.
4) Function: The PRINTALL function has been designed to provide the user with a facility that permits him to print out the complete employee record. The user has the choice of columns which he needs to get printed. If the user does not insert column values between 1 & 11 separated by blanks, then an appropriate error message is printed out and function comes to halt.

PROGRAM INFORMATION:

1) Inputs: None
2) Outputs: None
3) Global Variables: EREC
   COL
   NAME
4) Sub-functions Invoked: HEADING

TOTALS

USER INFORMATION:

1) Calling Forms: TOTALS or TOTS

2) User Response: Input Column Values between 1 & 11 or type ALL.

3) Output Response: Totals of all legal columns as input by the user, or an error message indicating wrong column values.

4) Function: The totals function has been designed to provide the user with useful totals. All totals when printed are appropriately labeled; money totals are in pennies. There are certain columns over which the totals are not available, such columns within the range of 1 to 11 if provided as input would be automatically masked out by the function. However, if the columns are not within the range of 1 to 11, then an error message is printed out and function comes to halt.

PROGRAM INFORMATION:

1) Inputs: None

2) Outputs: None

3) Global Variables: EREC

COL

4) Sub-functions Invoked: None

PRINTSEL

USER INFORMATION:

1) Calling Forms: PRINTSEL or PRTSEL

2) User Response: (i) Input column Values between 1 & 11 separated by blanks or type ALL.

(ii) Input selection condition

3) Output Response: All employee entries fulfilling the selection condition are printed, or a message indicating no one entry fulfills the given condition.

4) Function: The function has been implemented to provide the user with full control over printing. The entries when printed would include only those columns as specified by the user. Before printing starts all columns to be printed are appropriately labeled. If the given user condition is not satisfied by any entry then a message indicating this is printed.

PROGRAM INFORMATION:

1) Inputs: None

2) Outputs: None

3) Global Variables: EREC

COL

NAME

4) Sub-functions Invoked: HEADING

PRINTLINE

USER INFORMATION:

1) Calling Forms: PRINTLINE or PRTLN
2) User Response: Input the social security of the employee whose information is to be printed.
3) Output Response: Complete entry of the employee whose social security was input is printed, or a message indicating no such social security number exists.
4) Function: The function was implemented to provide the user with a single selection entry facility for printing. The entry when printed is appropriately labeled so as to indicate the meaning of different columns. If the given social security number is not found, then an appropriate message indicating so is printed out.

PROGRAM INFORMATION:

1) Inputs: None
2) Outputs: None
3) Global Variables: EREC
   COL
   NAME
4) Sub-functions Invoked: HEADING

INITIALIZE

USER INFORMATION:

1) Calling Forms: INITIALIZE
2) User Response: Enter the name of the employee whose entry is to be inserted. The INITIALIZE function is to be only called when a new data structure employee record is to be created.
3) Normal Output Response: Echo check of the first employee entry which was inserted, or an error message indicating incomplete information insertion.
4) Function: The function was designed to provide the user with a command which would provide an easy starting, and initializing of the data base.

PROGRAM INFORMATION:

1) Inputs: None
2) Outputs: EREC
   NAME
   AUXI
3) Global Variables: EREC
   NAME
   AUXI
4) Sub-functions Invoked: ADD
   DELETE

DELETE

USER INFORMATION:

1) Calling Forms: DELETE or DEL
2) User Response: Input the social security of the employee whose information is to be deleted, and date as specified.
3) Normal Output Response: A message indicating that the entry has been deleted, along with the deleted entry as output or a message indicating entry to be deleted not found.
4) Function: The delete function was designed to implement the deletion of unwanted information from the employee record information structure. Social security number of an employee is input to identify the entry to be deleted. The function responses back by indicating that the entry has been deleted and also outputs the deleted entry. In case the entry is not found which was to be deleted, then in such a case a message indicating this fact is output. All information effecting Department record is stored in AUXI at deletion time.

PROGRAM INFORMATION:

1) Inputs: None
2) Outputs: AUXI
3) Global Variables: EREC
   COL
   NAME
   AUXI
4) Sub-functions Invoked: HEADING
   DATE

ALTER

USER INFORMATION:

1) Calling Forms: ALTER or ALT
2) User Response: Insert the social security number of the employee whose information is to be altered.
3) Output Response: Echo check of the old entry and the altered entry, or an error message indicating wrong column selection, or an error message indicating incomplete information, or an error message indicating the entry to be updated not found.
4) Function: The function was designed to provide the user with the facility of updating employee information. The function besides updating the entry in the employee record structure also keeps track of information by which a particular department may be affected. All such information is stored in AUXI, where it is processed by the UPDATE function. In case complete updating is not performed by the user, an error message is output and function comes to halt. The requested column selection should be between 1 and 11 inclusive, columns 6 and 8 need not be altered by the user since their values depend upon other columns and the function takes care of them.

PROGRAM INFORMATION:

1) Inputs: None
2) Outputs: AUXI

3) Global Variables: EREC

NAME

COL

AUXI

4) Sub-functions invoked: HEADING

DATE

ADD

USER INFORMATION:

1) Calling Forms: ADD
2) User Response: (i) Enter the name of the employee, Last, First, Middle, whose information is to be inserted. (ii) Insert the complete information regarding the employee. Columns 6 and 8 need not be inserted (calculated by the function).
3) Normal Output Response: Echo check of the employee entry which was just inserted or message indicating incomplete information insertion.
4) Function: The function has been designed to provide the user with the facility of inserting new employee entries. ADD calls the ARRANGE function to find the exact location of insertion in alphabetic order. The function makes a check to see if complete information was inserted or not. As the entry is being inserted in the employee record structure, all necessary information relating to the employee's department is stored in the auxiliary storage array AUXI.

PROGRAM INFORMATION:

1) Inputs: None
2) Outputs: AUXI
3) Global Variables: EREC
   NAME
   COL
   AUXI
4) Sub-functions Invoked: ARRANGE
   HEADING
   DATE

UPDATE

USER INFORMATION:

1) Calling Forms: UPDATE or UPD
2) User Response: None
3) Output Response: * * * and a list of system commands to be executed or an error message indicating the absence of a department to be updated.
4) Function: The function was implemented to perform all outstanding updating which results from new or updated employee entries. The function takes information from AUXI, updates respective department records accordingly. If the function does not find a department which is to be updated, then an appropriate error message is output. After updating is complete, a list of system commands is output which are to be executed immediately.

PROGRAM INFORMATION:

1) Inputs: AUXI
2) Outputs: DREC
3) Global Variables: DREC
   AUXI
4) Sub-functions: None

INSERT

USER INFORMATION:

1) Calling Forms: INSERT or INS
2) User Response: Insert the name of the employee whose information is to be inserted.
3) Output Response: Echo check of the inserted entry or an error message indicating incomplete information insertion.
4) Function: The function has been designed to provide the user with the facility of inserting new employee entries. INSERT calls the ARRANGE function to locate the correct position of insertion in alphabetic order. The function checks to make sure if complete information was inserted or not. If the inserted information is incomplete, then an appropriate error message is output, indicating the user to try again.

PROGRAM INFORMATION:

1) Inputs: None
2) Outputs: None
3) Global Variables: EREC
   NAME
   COL
4) Sub-functions Invoked: ARRANGE
   HEADING

CHANGE

USER INFORMATION:

1) Calling Forms: CHANGE or CHG
2) User Response: Input the social security number of the employee whose information is to be changed.
3) Output Response: Complete information entry regarding the employee is shown after updating has been performed or an error message indicating incomplete information insertion.
4) Function: The change function was implemented to provide the user with a general updating facility. Social security number of an employee is input to identify the particular employee. Then the appropriate column numbers are inserted which the user plans to change. The function comes back and prints out the old information and waits for new inputs. On receiving the new inputs it carries on a number of validity checks. If the inputs are valid, then the updated entry is printed else an appropriate error message is output and function terminates.

PROGRAM INFORMATION:

1) Inputs: None
2) Outputs: None
3) Global Variables: EREC
   NAME
   COL
4) Sub-functions Invoked: HEADING

INTERNAL FUNCTIONS

HEADING

USER INFORMATION:

1) Calling Forms: NOT TO BE CALLED BY USER.
2) User Response: None
3) Output Response: None
4) Function: The HEADING function was designed to provide the user with a nice labeled output. A number of functions call the HEADING function to produce labels before printing out the required information. Complete label information is stored as a character vector and used what is needed.

PROGRAM INFORMATION:

1) Inputs: None
2) Outputs: HEAD
3) Global Variables: COL
4) Sub-functions Invoked: None

DATE

USER INFORMATION:

1) Calling Forms: NOT TO BE CALLED BY USER.

2) User Response: None

3) Output Response: None

4) Function: The DATE function was designed to calculate the difference in months, between two given dates in a regular fashion. The dates which are input to the function should have days as 28, 31, 30 or 1, since these are the only legal days of the month for hiring or relieving an employee. The DATE function is invoked by the ADD and ALTER functions.

PROGRAM INFORMATION:

1) Inputs: None

2) Outputs: MTHS, difference in months between DATE 1 and DATE 2 of EMPREC structure entry.

3) Global Variables: EREC

MTHS

4) Sub-function Invoked: None

ARRANGE

USER INFORMATION:

1) Calling Forms: NOT TO BE CALLED BY USER.
2) User Response: None
3) Output Response: None
4) Function: The ARRANGE function was designed to provide INSERT and ADD functions, with the capability of entering all new entries in the Employee record in an alphabetic order. ARRANGE function simply looks at the first 5 characters of the last name of an employee and finds the correct location of insertion in the Employee Record Structure.

PROGRAM INFORMATION:

1) Inputs: Name of the employee whose entry is to be inserted in alphabetic order, in the Employee Record Structure.
2) Outputs: Integer number indicating the correct position of insertion. (Index into the Employee Record Structure).
3) Global Variables: EREC
   NAME
   I (Index returned to the INSERT function.)
4) Sub-functions Invoked: None

CHAPTER V

DESCRIPTION OF

DEPARTMENT RECORD FUNCTIONS

Following is the complete listing of all the Department record functions. Each functions' normal and short names have been listed. The page number refers to an equivalent Employee record function whose purpose is exactly the same for the Employee record structure.

LIST OF DEPARTMENT RECORD FUNCTIONS

All the Department record functions are similar to the Employee record functions which have been described earlier in the report. Department record structure as seen by the user has been shown on page 27 of this report.

The Calling Forms are similar to that of Employee record functions, a list of all functions and their short names is given on page 56. The structure has only seven columns, therefore column values inserted by the user should always be between 1 and 7 inclusive. All functions help the user by printing out appropriate messages at appropriate times.

The Department record functions use three global variables, namely DREC, DNAME and COL which store the numeric information of all departments, the name of the departments and column values respectively.

In real life situations the total number of entries in the Department record would not be more than 25 as compared to Employee record, which may have several thousand entries. Thus the comparative response time may be low in this case.

APPENDIX I

USER'S GUIDE

Appendix I contains comprehensive execution examples of all the user and system APL/360 commands. The first two pages of the Appendix lists the contents of the Appendix.

Appendix I can be subdivided into three sub-groups:

(i) Employee-Record commands.

(ii) Department-Record commands.

(iii) APL/360 system commands.

Appendix I which has also been labeled as the user manual with examples and commands is quite self explanatory.

All the keying which is done by the user has been shown with a shaded background. After each line which is keyed by the user a RETURN key hit is implied. Since a RETURN hit on system APL/360 does not produce any printed symbol, the user, while going through the execution examples, must bear this fact in mind.

The user gives a command by typing the short or long name of a command. The system would reply back appropriately and ask for a specific type of input. All dollar figures to be input should be first converted to pennies by the user and then input. A good thing to remember is that the use of a decimal point, while inputting any numeric information is taboo. The same holds true for the information output by the system. All dollar figures are in pennies and should be converted appropriately for any real life situation use.

All the individual execution examples terminate with a * * * termination symbol which clearly indicates to the user that the function has terminated and he should proceed with the next command. The symbol * * * does not in

any way indicate a successful termination.

All the commands have also been illustrated with abnormal terminations. An appropriate error message is given to the user to indicate his problem. The message(s) are self explanatory and easy to read.

In the very last section of Appendix I, some of the very useful system APL/360 commands have been illustrated. All commands have been appropriately labeled. A more detailed discussion of these system commands can be found in [2], [3].

USER MANUAL WITH EXAMPLES & COMMANDS

EXECUTION EXAMPLES EMPLOYEE RECORD FUNCTIONS

DESCRIBE

NAME	SHORT_FORM	FUNCTION
CHANGE	CHG	UPDATE'S INFORMATION IN THE EMPLOYEE RECORD STRUCTURE.
DELETE	DEL	DELETE'S INFORMATION IN THE EMPLOYEE RECORD STRUCTURE.
INSERT	INS	INSERT'S INFORMATION IN THE EMPLOYEE RECORD STRICTURE.
PRINTSEL	PRTSEL	PRINT'S SELECTED COLUMNS FROM THE EMPLOYEE RECORD STRUCTURE.
PRINTLINE	PRTLN	PRINT'S A SELECTED LINE FROM THE EMPLOYEE RECORD STRUCTURE.
PRINTALL	PRTALL	PRINT'S THE WHOLE EMPLOYEE RECORD STRUCTURE.
TOTALS	TOTS	COMPUTES TOTALS OVER SELECTED COLUMNS.
CODES	CODES	DESCRIBE'S THE HEADING CODES.
INITIALIZE	INITIALIZE	INITIALIZES THE EMPLOYEE RECORD DATABASE.
ADD	ADD	INSERT'S NEW ENTRIES IN THE EMPLOYEE RECORD STRUCTURE.
ALTER	ALT	UPDATE'S INFORMATION IN THE EMPLOYEE RECORD STRUCTURE.
UPDATE	UPD	UPDATE'S DEPARTMENT RECORDS USING AUXI.

CODES

CODE	MEANING
SSN	SOCIAL SECURITY NUMBER.
RNK	RANK OF AN EMPLOYEE.
POS	POSITION OF AN EMPLOYEE.
PER	SALARY SPREAD PERIOD.
ATN	ANNUAL TENTHS.
ANSAL	ANNUAL SALARY.
DEPT	DEPARTMENT CODE.
BUDGET	ANNUAL BUDGET.
FCODE	FUND CODE.
DATE1	APPOINTMENT DATE.
DATE2	TERMINATION DATE.

```
     INITIALIZE

*** EXECUTE THE SYSTEM COMMAND AFTER THIS ADDITION ***
)COPY DEPREC DREC

INSERT THE NAME.......LAST,FIRST MIDDLE

TODSON, MARK

INSERT EMPLOYEE'S RATE IN PENNIES PER TENTH PER MONTH
□:
      9000

INSERT EMPLOYEE'S INFORMATION AS REQUIRED

   SSN     RNK POS PRD ATN DEPT FCODE  DATE1  DATE2

□:
      555678787 4 2 9 4 8 10175 93075

******* COMPLETE INFORMATION WAS NOT INSERTED, TRY AGAIN *******

□:
      555678787 4 2 9 4 8 43213 10175 93075

**** INSERTED ENTRY LOOKS AS ****

   SSN     RNK POS PRD ATN ANSAL  DEPT BUDGET  FCODE  DATE1  DATE2

3    TODSON, MARK
555678787  4  2  9  4  432000  8  324000  43213  10175  93075

***


***
```

```
      PRINTALL

INSERT COLUMNS TO BE SEPARATED BY BLANKS OR TYPE IN ALL

□:
      ALL

   SSN     RNK POS PRD ATN ANSAL  DEPT BUDGET  FCODE  DATE1  DATE2

1   BOWEN, JOHN
515678888  5  2  12  9  1245767  5  1245767  41121  10175  123175

2   COOL, JOE
5 15629777  4  1  9  6  720000  8  180000  42231  90175  53176

3   DOE, JOHN
515629778  4  2  9  4  432000  8  324000  43212  10175  93075

4   MCMILLEN, BRUCE
505679778  7  2  9  4  1200000  12  1200000  42211  90175  53176

***


      PRINTALL

INSERT COLUMNS TO BE SEPARATED BY BLANKS OR TYPE IN ALL

□:
      123

**** COLUMN VALUES INSERTED ARE WRONG ****

***
```

```
      PRINTALL

INSERT COLUMNS TO BE SEPARATED BY BLANKS OR TYPE IN ALL

[]:
      1 6 8


   SSN     ANSAL  BUDGET

1    BOWEN, JOHN
515678888  1245767  1245767

2    COOL, JOE
515629777  720000  180000

3    DOE, JOHN
515629778  432000  324000

4    MCMILLEN, BRUCE
505679778  1200000  1200000

***


      PRINTLINE

INSE RT SOCIAL SECURITY NUMBER

[]:
      515629778


   SSN     RNK POS PRD ATN ANSAL  DEPT BUDGET  FCODE  DATE1  DATE2

3    DOE, JOHN
515629778  4  2  9  4  432000  8  324000  43212  10175  93075

***
```

```
     PRINTLINE

INSERT SOCIAL SECURITY NUMBER

□:
      515629867

 ******* SOCIAL SECURITY NUMBER NOT FOUND *******

***


     PRINTSEL

INSERT COLUMNS TO BE PRINTED SEPARATED BY BLANKS OR TYPE ALL

□:
      4 5 6

INSERT THE CONDITION LIKE   EREC[;3]=1
□:
       EREC[;4]>9

 PRD ATN ANSAL


1    BOWEN, JOHN
12  9  1245767

***
```

```
      PRINTSEL

INSERT COLUMNS TO BE PRINTED SEPARATED BY BLANKS OR TYPE ALL
□:
      45

**** COLUMN VALUES INSERTED ARE WRONG ****
***



      ADD

INSERT THE NAME.....LAST,FIRST MIDDLE
RASHID, HAROON
INSERT EMPLOYEE'S RATE IN PENNIES PER TENTH PER MONTH
□:
      9000
INSERT EMPLOYEE'S INFORMATION AS REQUIRED

   SSN     RNK POS PRDATN DEPT FCODE   DATE1   DATE2

□:
      515629667  4 1 9 4   12    2314  90176    53176

**** INSERTED ENTRY LOOKS AS ****

   SSN     RNK POS PRDATN ANSAL   DEPT BUDGET   FCODE   DATE1   DATE2

6    RASHID, HAROON
515629667   4   1   9   4   432000   12   108000   2314   90176   53176
```

```
      ADD

INSERT THE NAME.....LAST,FIRST MIDDLE

MOOSES, JAMES

INSERT EMPLOYEE'S RATE IN PENNIES PER TENTH PER MONTH
□:
      10000

INSERT EMPLOYEE'S INFORMATION AS REQUIRED

   SSN     RNK POS PRIATN DEPT FCODE  DATE1  DATE2

□:
      555129886  4 1 12 6 8   12342

******* COMPLETE INFORMATION WAS NOT INSERTED, TRY AGAIN *******

□:
      555129886  4 1 12 6 8  43123  10175 123175


**** INSERTED ENTRY LOOKS AS ****

   SSN     RNK POS PRIATN ANSAL   DEPT BUDGET   FCODE   DATE1   DATE2

6     MOOSES, JAMES
555129886   4   1   12   6   720000   8   720000   43123   10175   123175

***
```

```
      ALTER

INSERT SOCIAL SECURITY NUMBER

□:
      512343232

***** ENTRY TO BE CHANGED NOT FOUND *****

***


      ALTER

INSERT SOCIAL SECURITY NUMBER

□:
      555129886

INSERT COLUMNS TO BE CHANGED SEPARATED BY BLANKS OR TYPE ALL

COLUMNS 6 AND 8, THE ANNUAL SALARY AND BUDGET
NEED NOT TO BE INSERTED

□:
      7

*****  OLD ENTRY LOOKS AS  *****

 DEPT

6     MOOSES, JAMES
13

*****  ENTER NEW VALUES FOR THE SELECTED COLUMNS  *****
□:
      13
```

```
INSERT THE RATE PER TENTH, PER MONTH

IF RATE NOT CHANGED TYPE IN ZERO

□:
      0

*** CHANGED ENTRY LOOKS AS ***

   SSN     RNK POS PRDATN ANSAL   DEPT BUDGET   FCODE   DATE1   DATE2

6    MOOSES, JAMES
555129886  4  1   12   6   720000   13   720000   43123   10175   123175

***


     ALTER

INSERT SOCIAL SECURITY NUMBER

□:
      555129886

INSERT COLUMNS TO BE CHANGED SEPARATED BY BLANKS OR TYPE ALL

COLUMNS 6 AND 8, THE ANNUAL SALARY AND BUDGET
NEED NOT TO BE INSERTED

□:
      3 4

**** COLUMN VALUES ARE WRONG, TRYAGAIN ****

***
```

```
UPDATE



**** DEPARTMENT  12  WAS NOT FOUND ****


**** DEPARTMENT  13  WAS NOT FOUND ****

**** DEPARTMENT  13  WAS NOT FOUND ****

**** DEPARTMENT  13  WAS NOT FOUND ****

**** EXECUTE THE FOLLOWING SYSTEM COMMANDS ****
**** BEFORE WORKING FURTHER                ****
)SAVE EMPREC
)LOAD DEPREC
)COPY EMPREC DREC
)SAVE DEPREC
)LOAD EMPREC

***
```

```
     TOTALS

INSERT COLUMNS SEPARATED BY BLANKS OR TYPE ALL
□:
      ALL
****** TOTAL ANNUAL TENTHS ARE *******
          19
******* TOTAL OF ANNUAL SALARIES IS *******
          3165767
******* TOTAL BUDGET IS *******
          2625767
***

     TOTALS

INSERT COLUMNS SEPARATED BY BLANKS OR TYPE ALL
□:
      5 6 7
****** TOTAL ANNUAL TENTHS ARE *******
          19
******* TOTAL OF ANNUAL SALARIES IS *******
          3165767
***
```

```
     TOTALS

INSERT COLUMNS SEPARATED BY BLANKS OR TYPE ALL
[]:
     123
******* COLUMNS INSERTED NOT LEGAL *******
***


     TOTALS


INSERT COLUMNS SEPARATED BY BLANKS OR TYPE ALL
[]:
     1 2 3
******* TOTALS NOT AVAILABLE ON REQUESTED COLUMNS *******
***
```

```
     DELETE
NSERT SOCIAL SECURITY NUMBER

:
     515629778

******  ENTRY NOT FOUND FOR DELETION  *******

**


     DELETE
NSERT SOCIAL SECURITY NUMBER

:
     555678787


*** THE ENTRY DELETED IS ****

  SSN     RNK POS PRD ATN ANSAL  DEPT BUDGET  FCODE  DATE1  DATE2

     TODSON, MARK
55678787  4   2   9   4   432000  8   324000  43213  10175  93075

** ENTER THE DATE AS_ THE LAST DAY OF THIS MONTH,YEAR 76....63076 ***

:
     63075

**
```

```
      DELETE


INSERT SOCIAL SECURITY NUMBER.

□:
      515678888



*** EXECUTE THE SYSTEM COMMAND AFTER THIS DELETION ***

)COPY DEPREC DREC

**** THE ENTRY DELETED IS ****

   SSN     RNK POS PRD ATN ANSAL   DEPT BUDGET   FCODE   DATE1   DATE2

1    BOWEN, JOHN
515678888  5   2   12   9   1245767   5   1245767   41121   10175   123175

*** ENTER THE DATE AS_ THE LAST DAY OF THIS MONTH,YEAR 76....63076 ***

□:

      63075

***
```

```
      INSERT

INSERT THE NAME.....LAST,FIRST MIDDLE
COOL, JOE
INSERT EMPLOYEE'S INFORMATION SEPARATED BY BLANKS
   SSN     RNK POS PRD ATN ANSAL  DEPT BUDGET  FCODE  DATE1  DATE2
□:
      555213213 4 12 4 360000 12 2700000 41532    90175 53176
****** ** COMPLETE INFORMATION WAS NOT INSERTED, TRY AGAIN *******
   SSN     RNK POS PRD ATN ANSAL  DEPT BUDGET  FCODE  DATE1  DATE2
□:
      55521321 4 1 12 4 360000 12 270000 43212   90175 53176

**** INSERTED ENTRY LOOKS AS ****

   SSN     RNK POS PRD ATN ANSAL  DEPT BUDGET  FCODE  DATE1  DATE2

1   COOL, JOE
55521321  4  1  12  4  360000  12  270000  43212  90175  53176

***
```

```
      CHANGE

INSERT SOCIAL SECURITY NUMBER
□:
      515620987
*******  ENTRY TO BE CHANGED NOT FOUND  *******
***


      CHANGE

INSERT SOCIAL SECURITY NUMBER
□:
      515629778
INSERT COLUMNS TO BE CHANGED SEPARATED BY BLANKS OR TYPE ALL
□:
      123
****  C OLUMN VALUES ARE WRONG ****
***
```

```
      CHANGE

INSERT SOCIAL SECURITY NUMBER

□:
      555129886

INSERT COLUMNS TO BE CHANGED SEPARATED BY BLANKS OR TYPE ALL

□:
      4

*****  OLD ENTRY LOOKS AS  *****

 PRD

6     MOOSES, JAMES
12

*****  ENTER NEW VALUES FOR THE SELECTED COLUMNS  *****

 PRD
□:
      12

***
```

EXECUTION EXAMPLES OF DEPARTMENT RECORD FUNCTIONS

DESCRIBE

NAME	SHORT_FORM	FUNCTION
DCHANGE	DCH	UPDATE'S INFORMATION IN THE DEPARTMENT RECORD.
DDELETE	DDEL	DELETE'S INFORMATION FROM THE DEPARTMENT RECORD STRUCTURE.
DINSERT	DIN	INSERT'S NEW DEPARTMENT ENTRY.
DPRINTALL	DPALL	PRINT'S THE WHOLE DEPARTMENT RECORD STRUCTURE.
DPRINTLINE	DPL	PRINT'S A SELECTED LINE FROM THE DEPARTMENT RECORD STRUCTURE.
DPRINTSEL	DPSEL	PRINT'S SELECTED COLUMNS FROM THE DEPARTMENT R CORD STRUCTURE.
DTOTALS	DTOTS	COMPUTES TOTALS OVER SELECTED COLUMNS.
DCODES	DCODES	EXPLAIN'S CODES USED IN HEADING.

DCODES

CODE	MEANING
NO	NUMBER.
DCO	DEPARTMENT CODE.
LTH	LOANED TENTHS.
AO DOL	ALLOCATED DOLLARS.
AS DOL	ASSIGNED DOLLARS.
AOTN	ALLOCATED TENTHS.
ASTN	ASSIGNED TENTHS.
FRTEN	FREE TENTHS.

***

```
     DINITIALIZE

INSERT THE NAME OF THE NEW DEPARTMENT
COMPUTER SCIENCE
INSERT NEW DEPARTMENT'S INFORMATION SEPARATED BY BLANKS
□:
       23 0 4567800 3856700 68 68 0

***** INSERTED ENTRY LOOKS AS ****

NO    DEPARTMENT     DCO LTH  AO DOL  AS DOL AOTN ASTN FRTEN

3     COMPUTER SC     23   0  4567800  3856700  68   68   0
***

***
```

```
     DINITIALIZE

INSERT THE NAME OF THE NEW DEPARTMENT

MODERN LANGUAGES

INSERT NEW DEPARTMENT'S INFORMATION SEPARATED BY BLANKS

□:
      46  2 31345689 3523412

******* COMPLETE INFORMATION WAS NOT INSERTED, TRY AGAIN *******

□:
      46 2 4576800 3056789 56 54 0

***** INSERTED ENTRY LOOKS AS ****

NO   DEPARTMENT    DCO LTH  AO DOL  AS DOL AOTN ASTN FRTEN

3    MODERN LANG    46  2  4576800  3056789  56  54  0

***

***
```

```
      DPSEL

INSERT THE COLUMNS TO BE PRINTED SEPARATED BY BLANKS

□:
      145

**** COLUMNS INSERTED ARE WRONG ****

***
      DPSEL

INSERT THE COLUMNS TO BE PRINTED SEPARATED BY BLANKS

□:
      1 2

INSERT THE CONDITION LIKE  DREC[;1]=11
□:
      DREC[;1]=55

NO    DEPARTMENT     DCO LTH



4     FOODS NUTRI     55   0

***
```

```
      DTOTS

INSERT COLUMNS SEPARATED BY BLANKS OR TYPE ALL

□:
      123
**** COLUMN VALUES INSERTED ARE WRONG ****

***

      DTOTS

INSERT COLUMNS SEPARATED BY BLANKS OR TYPE ALL

□:
      ALL

**** SUM OF LOANED TENTHS ****
         60

**** SUM OF ALLOCATED DOLLARS ****
         12391257

**** SUM OF ASSIGNED DOLLARS ****
         49387522

**** SUM OF ALLOCATED TENTHS ****
         946

**** SUM OF ASSIGNED DOLLARS ****
         766

**** SUM OF FREE TENTHS ****
          102


***
```

```
      DTOTS

INSERT COLUMNS SEPARATED BY BLANKS OR TYPE ALL

□:
      1 2 3

**** SUM OF LOANED TENTHS ****
          60

**** SUM OF ALLOCATED DOLLARS ****
          12391257
**** SUM NOT AVAILABLE ****

***
```

```
      DTOTS

INSERT COLUMNS SEPARATED BY BLANKS OR TYPE ALL

□:
      1
**** SUM NOT AVAILABLE ****

***
```

```
      DPALL
INSERT COLUMNS TO BE PRINTED SEPARATED BY BLANKS OR TYPE IN ALL
□:
      ALL

NO   DEPARTMENT     DCO LTH  AO DOL  AS DOL AOTN ASTN FRTEN

1    AGRICULTURE     23  7  4454578  34897600  123  112  8
2    ANTHRAPOLOG     88  6  4556678  3456566   56   57   6
3    BIOLOGY         21  26  1000000  563175   124  120  4
4    MEDICAL TEC     36  12  256734  256457   200  187  30
5    ZOOLOGY         44  7  456767  5645789   231  158  50
***

      DPALL
INSERT COLUMNS TO BE PRINTED SEPARATED BY BLANKS OR TYPE IN ALL
□:
      123
**** COLUMN VALUES INSERTED ARE WRONG ****
***
```

```
      DPALL
INSERT COLUMNS TO BE PRINTED SEPARATED BY BLANKS OR TYPE IN ALL
□:
      1 4 7

NO    DEPARTMENT     DCO  AS DOL FRTEN

1     AGRICULTURE     23  34897600  8
2     ANTHRAPOLOG     88  3456566   6
3     BIOLOGY         21  563175    4
4     MEDICAL TEC     36  256457    30
5     ZOOLOGY         44  5645789   50
***

      DPL
INSERT DEPARTMENT CODE
□:
      21

NO    DEPARTMENT     DCO LTH  AO DOL  AS DOL AOTN ASTN FRTEN

3     BIOLOGY         21  26  1000000  563175  124  120  4
***
```

```
      DPL
INSERT DEPARTMENT CODE
□:
      33

******* DEPARTMENT NOT FOUND *******
***

      DCH
INSERT THE DEPARTMENT CODE
□:
      23
INSERT COLUMNS TO BE CHANGED SEPARATED BY BLANKS OR TYPE ALL
□:
      2 3 5
*****  OLD ENTRY LOOKS AS  *****

NO   DEPARTMENT     LTH  AO DOL AOTN

1    AGRICULTURE    7  4454578  123
*****  ENTER NEW VALUES FOR THE SELECTED COLUMNS *****
□:
      8 445488 240
***
```

ILLEGIBLE

THE FOLLOWING DOCUMENT (S) IS ILLEGIBLE DUE TO THE PRINTING ON THE ORIGINAL BEING CUT OFF

ILLEGIBLE

DCH

INSERT THE DEPARTMENT CODE

□:
23

INSERT COLUMNS TO BE CHANGED SEPARATED BY BLANKS OR TYPE ALL

□:
45

***** COLUMN NUMBERS ARE WRONG *****

***

DCH

INSERT THE DEPARTMENT CODE

□:
23

INSERT COLUMNS TO BE CHANGED SEPARATED BY BLANKS OR TYPE ALL

□:
4 6 7

***** OLD ENTRY LOOKS AS *****

NO DEPARTMENT AS DOL ASTN FRTEN

1 AGRICULTURE 34897600 112 8

***** ENTER NEW VALUES FOR THE SELECTED COLUMNS *****

□:
34897666 115

****** COMPLETE INFORMATION WAS NOT INSERTED, TRY AGAIN

34897666 115 9

***

DCH

INSERT THE DEPARTMENT CODE

□:

56

******* ENTRY TO BE CHANGED NOT FOUND *******

***

DCH

INSERT THE DEPARTMENT CODE

□:

23

INSERT COLUMNS TO BE CHANGED SEPARATED BY BLANKS OR TYPE ALL

□:

ALL

***** OLD ENTRY LOOKS AS *****

NO DEPARTMENT DCO LTH AO DOL AS DOL AOTN ASTN FRTEN

1 AGRICULTURE 23 8 445488 34897666 240 115 9

***** ENTER NEW VALUES FOR THE SELECTED COLUMNS *****

□:

23 9 444578 34897645 212 124 9

```
      DIN

 INSERT THE NAME OF THE NEW DEPARTMENT

 FOODS NUTRITION

 INSERT NEW DEPARTMENT'S INFORMATION SEPARATED BY BLANKS

 □:
      55 0 5676500 4567890 123 120 3

 ***** INSERTED ENTRY LOOKS AS ****

 NO   DEPARTMENT    DCO LTH  AO DOL  AS DOL AOTN ASTN FRTEN

 4    FOODS NUTRI    55  0  5676500  4567890  123  120  3

 ***


    DIN

INSERT THE NAME OF THE NEW DEPARTMENT

EDUCATION

INSERT NEW DEPARTMENT'S INFORMATION SEPARATED BY BLANKS

□:
      56 0

******* COMPLETE INFORMATION WAS NOT INSERTED, TRY AGAIN *******

□:
      56 0 5768700 4356700 49 49 0

***** INSERTED ENTRY LOOKS AS ****
```

```
4     EDUCATION       56   0   5768700   4356700   49   49   0

***



      DDEL

INSERT DEPARTMENT CODE

□:
      57

******* ENTRY NOT FOUND FOR DELETION *******

***
      DDEL

INSERT DEPARTMENT CODE

□:
      56

***** THE ENTRY DELETED IS *****

NO    DEPARTMENT     DCO LTH   AO DOL   AS DOL AOTN ASTN FRTEN

4     EDUCATION       56   0   5768700   4356700   49   49   0

***
```

EXECUTION EXAMPLES OF APL/360 SYSTEM COMMANDS

THIS BOOK CONTAINS NUMEROUS PAGES WITH THE ORIGINAL PRINTING ON THE PAGE BEING CROOKED.

THIS IS THE BEST IMAGE AVAILABLE.

HOW TO INITIALIZE YOUR ACCOUNT NUMBER

```
VM/370 ONLINE    LJH359 QSYOSU

DIAL APL
⍳⍺⎕∈⍳ ~○ ⍺*⎕       022
)INITIALIZE
002) 19.55.44 03/10/76 ΔΔΔ 106171

ENTER (1) )KSUACCT AAAAAAAA III (WHERE AAAAAAAA IS YOUR ACCOUNT NUMBER AND III ARE YOUR INITIALS)
      (2) )OFF OR )OFF HOLD
      (3) )AAAAAAAA SSN

      )KSUACCT DPF0CJD3 HRS
      )OFF
002  19.56.39 03/10/76 ΔΔΔ
CONNECTED     0.00.56  TO DATE  559.34.36
CPU TIME      0.00.00  TO DATE    0.24.17
TOTAL COST      $0.06  BALANCE   $8549.30
⍳⍴○* _⍴○| ⍺*⎕       022
]
```

HOW TO SIGN ON.

```
VM/370 ONLINE     LJH359 QSYOSU

DIAL APL
⌊⍳⍺⍞∈⌊ ~○ ⍺*⍞       021
)MASK
RRRRRRRRRRRRRRRRRRRRR
)DP85C7M6 515629778
001) 20.45.03 03/03/76 HRS 107226

   A P L \ 3 6 0

SAVED  20.43.28 03/03/76
```

HOW TO SIGN OFF.

```
      )CONTINUE
  20.45.22 03/03/76 CONTINUE
001  20.45.23 03/03/76 HRS
CONNECTED     0.00.21  TO DATE    38.01.44
CPU TIME      0.00.00  TO DATE     0.03.31
TOTAL COST      $0.04  BALANCE     $128.61
⌊⍜* _⍜| ⍺*⍞       021
```

WORKSPACE IDENTIFICATION COMMAND

```
      )WSID
EMPREC

      )WSID EMPREC1
WAS EMPREC

      )WSID
EMPREC1
```

COPY COMMAND

```
      )COPY DEPREC DREC
SAVED  14.10.59 06/07/76
```

SAVE COMMAND

```
)SAVE EMPREC1
14.27.51 06/07/76
```

LOAD COMMAND

```
)LOAD EMPREC
SAVED  13.57.17 06/07/76
```

CORRECTION OF MISTAKES

```
)SAVE EMPREEC
      v
      EC
14.31.30 06/07/76
```

WORKSPACE LIBRARY COMMAND

```
      )LIB

CONTINUE
DP8
EMPREC
DEPREC


      )WSID EMPREC1

WAS EMPREC

      )SAVE EMPREC1

20.41.28 03/03/76


      )LIB

CONTINUE
DP8
EMPREC
DEPREC
EMPREC1
```

CONCLUSIONS

Maximum size of the database was limited to 200-250 entries, which was a serious implementation restriction. The average cost of an update was 64 cents, including the 6 cent cost for sign-on and sign-off.

I think that the interactive system is far easier to use as compared to a conventional batch system. There is continuous guidance available to the user when working with an interactive system.

The cost of development for this prototype was estimated to be significantly less than for the related batch system [4]. Of course, there is not a good basis for comparison because

(i) The batch system implemented several more functions which were not included in the interactive prototype.

(ii) The batch system included considerable code for the schema and sub schema required for the IDMS system. The code provides considerable execution efficiency and protection of shared data which is not available in the interactive prototype.

The question was raised whether a batch or an interactive system is more effective. In general we can observe that if high volume of data is to be processed and if a knowledgeable user can prepare data with few errors, then fewer reruns of a program would be needed, which would considerably decrease linkage overhead and long delays. Also if the user has other work he can do while waiting for the runs to complete, then a batch system is effective.

If, on the other hand, low volume of data is to be processed and the user is a novice and prone to errors (for example, due to infrequent use,

the user forgets the commands) then an interactive system would be more effective, because it can afford the user a chance to correct errors and provide immediate results. Further, if the user has no other work to do while waiting for the results (for example, his next activity depends upon results of the computation) then the immediate processing of a transaction is very effective.

Other questions related to user acceptance of the interactive system were answered during development of the system. Representatives from the Dean's office were requested to watch a demonstration and try a few commands. (Dean Carpenter tried executing few commands; he had previous experience with a time sharing BASIC terminal. Mrs. Olson watched but did not try the terminal; she had never used a computer terminal). From the demonstration we reached the following conclusions:

(i) Dollar figures in pennies were acceptable as long as consistency was maintained throughout the database in specifying employee salaries, allocated dollars, assigned dollars, etc. in terms of pennies.

(ii) Department codes were acceptable since they are used in the present system, used in the Dean's office.

(iii) It seems (opinion) that commands are acceptable if there is one-to-one correspondence with the logical operations performed in the office. A user would not tolerate having to memorize a sequence of commands to perform what is logically a single step. Several of the commands in the prototype do have a one-to-one correspondence with the operations commonly performed in the Dean's office.

Example: Hire an employee . . . . . ADD

Fire an employee . . . . . DELETE

Allocate money to a department . DCHANGE

Unfortunately, in the prototype the user has to execute several other commands such as

UPDATE

)SAVE

)LOAD, etc.

But these could be eliminated when a better version of APL/360 becomes available at Kansas State University.

Some approximate figures are given regarding the programming effort of this prototype:

(i)	Design	30 hours
(ii)	Programming & Debugging	120 hours
(iii)	Documentation	20 hours

The above figure for programming and debugging was low due to

(i) Assorted built-in operators offered by APL/360, which greatly reduce the size of the programs and the associated programming effort.

(ii) Interactive debugging, which allows the interrogation of the values of all the associated variables at the termination of a program, which significantly helps the debugging process.

The code generated for the whole system was approximately 1600 lines. There were a total of 27 modules with an average of 55 lines. The number of lines coded per hour were 8.

BIBLIOGRAPHY

1. Gilman, L. and Rose, A. J., APL An Interactive Approach, 2nd Ed., Wiley, New York, 1974.

2. Berry, P., APL/360 Primer, 3rd Ed., IBM, New York 10604 (GH20-0689-2).

3. APL/360-OS and APL/360-DOS User's Manual, IBM, New York 10604 (GH20-0906-1).

4. Weber, P. William, An Implementation of a Budgetary System for the College of Arts & Sciences, M.S. Report, Kansas State University, 1976.

ACKNOWLEDGMENTS

I am indebted to Dr. William J. Hankly for his invaluable guidance and kind supervision throughout the course of this work.

It is only due to the tremendous moral support and constant encouragement of Mrs. Safia Khokhar and Mr. Mohammad S. Khokhar--my lovely parents--that I have been able to achieve my goal. I do not have words to thank them for the sweet words they have been writing to me from Pakistan.

A PROTOTYPE TO ILLUSTRATE INTERACTION WITH A PERSONNEL DATABASE

by

HAROON RASHID

B. S., Kansas State University, 1974

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1976

A prototype to illustrate interaction with a personnel database for the college of Arts and Sciences.

The report deals with the implementation of a prototype interactive system, using the interactive language APL/360 which was chosen because

(i) It is highly interactive

(ii) It is available at KSU

(iii) It has built-in structures which facilitate operations on data arrays.

The system is composed of two major data structures, Employee record and Department record. Due to the restrictions of available version of APL/360, the structures had to be stored in separate workspaces (files) with a limit of 250 to 200 entries per structure. Each workspace includes local functions which manipulate the local data structures. Limited inter-communication between the two workspaces has also been implemented with the help of

(i) ADD function, to add additional employee entries

(ii) ALTER function, to update the existing employee entries.

All other functions work with information structures within a single workspace.

The problem was coded as small modules, each module being an APL function, the largest one being 90 lines. All input and output is through the IBM-2741 interactive terminal. The system does demonstrate the advantages which a person has, when using an interactive system.

The programs as they stand now do not implement a complete solution to all the data maintenance problems.

The report includes a description of programs, inputs, outputs, and sample execution of all the programs.