REPORT OF TERM PROJECT

ABSTRACT

In this article, The lightweight RDMS (Relational Database Management System) and development steps of Project with using C programming language is mentioned. The nature of Project based on well design, flexibilitiy(using macros for string and lines sizes) and compactness. Thus, developer should design very compatible and detailed system. In addition, Project has table processing steps so it creates different approaches for the solution scope(file based or memory based or hybrid). In this Project, Developed solution scope based on memory usage and file operations. I aimed fast processing and decrease code complexity. So I read table from file to the memory (using structs) then processed it. Because primary key and digit,alpanumerical character controls are works really effective in memory comparing to reaching file permanently. One of the important step is modularity. Modules should be designed and developed well related to each other if necessary. I developed 20 modules for database processing and table processing. In addition, I used two structs for complementary development capability. At last, some modules that developed in Project are based on file operations, some of modules are based on memory operations and effective algorithm.

DEVELOPED FUNCTIONS

1-) void print_menu() ;
Prints menu on the screen. It helps to code clearness.

2-)void create_database(DATABASE db,int *database_num);**
It takes database struct array address and number of database address created dynamically in main function. In function, it takes database name from user and controls database existence. If same named database not created before. It creates database and increases database number. Then it formats file name as "databasename_tablelist.txt" and creates tablelist file of database with 0 numbered table then closes file.

3-) int display_database(DATABASE db,int *database_num);**
It takes database struct array address and number of database address created dynamically in main function. If there is no database. It prints "NO DATABASE" on the screen then returns 0. If database(s) exists prints all of them on the screen and returns 1.

4-) void select_database(DATABASE db,int *database_num,DATABASE** select);**
It takes database struct array adress, selected database struct array(1 elemented) adress and number of database adress created dynamically in main function. In this function calls **display_database** function to show created database list and takes index value from user. Then controls the index value correct or not. If correct, it assigns to the created database named selected_db dynamically in main.

5-)void save_file(FILE **fp,DATABASE db,int *database_num);**
It takes file pointer, database struct array adress and number of database address created dynamically in main function.It opens taken file pointer in "w" mode and firstly writes total database number and then writes the database names in file sequentially. Then it closes file.

6-) void read_file(FILE **fp,DATABASE db,int *database_num);**
It takes file pointer, database struct array address and number of database created dynamically in main function.It opens taken file pointer in "r" mode in main function and firstly reads total database number and reallocs database array then reads the database names from file sequentially to memory(assigns to struct array arguments). Then it closes file.

7-) char* create_table(DATABASE select,TABLE** selected_table);**
It takes database struct array address and selected database struct array(1 elemented) address created dynamically in main function. In function, it takes table name from user and controls same name existence. Then it opens file name (formatted as "tablename_databasename_table.txt") in "w+" mode and take table

inputs(row number, column number, column type characteristic, column primary key characteristics,table contents) from user and controls assume that the column if type is integer inputs are compatible in form of integer. It controls that using isDigit,isAlpha functions in ctype.h library. In addition if the primary key exists it control the other inputs given before. Using memory by the way of creating struct is faster for this operations. But it uses more memory comparing to file reaching operations. After all, if the input is correct it writes in opened table file and closes file at the end of processing. Then it frees used memory for table operations.(The primary key control algorithm is developed effectively)Then it returns table name to main function for appending operations of tablelist file.

8-) void add_table_to_list(DATABASE ** select);
It just takes selected database struct array(1 elemented) address as a parameter.In function, it takes table number from selected struct and increases value of it and opens file then writes value in it.Then it closes file at the end of function.

9-) int display_tablelist(DATABASE selected_db);**
It just takes selected database struct array(1 elemented) address as a parameter. It helps to display tablelist selection to main or other functions. It prints loaded table from file to memory on the screen. If database has no table it returns 0 else returns 1.

10-) int load_tablelst(DATABASE selected_db);**
It just takes selected database struct array(1 elemented) address as a parameter. Then it reaches database name form struct. After that it formats name compatible with filename then it opens and allocates memory for the list in memory and load contents to tablelist struct element by passing allocated adress in it for tablelist. After assigment of elements completed it closes file. In addition, it controls if tablelist not created it returns menu. If operations completed succesfully it returns 1 value.

11-) void select_table(DATABASE selected_db,TABLE** selected_table);**
It takes selected database(DATABASE typed) struct array(1 elemented) address and selected table (TABLE typed) struct array(1 elemented) address as a parameters. It uses **display_tablelist** function in it to help user choice. User selects index value and selected table and table name copied from selected database structs' tablelist element to selected_table TABLE typed structs' table_name element by using strcpy function. That step is preparing step for load table from memory.

12-) void load_table(TABLE selected,DATABASE** selectedb);**
It takes selected database(DATABASE typed) struct array(1 elemented) address and selected table (TABLE typed) struct array(1 elemented) address as a parameters. Then it creates variables to help code readability as a temporary variable. Then it reads written datas from file then it passes datas (row number, column number, column type characteristic, column primary key characteristics,table contents) to the selected table struct in main according to written order.

13-)void display_table(TABLE selected,DATABASE** selected_db);**
It takes selected database(DATABASE typed) struct array(1 elemented) address and selected table (TABLE typed) struct array(1 elemented) address as a parameters. After that it prints the arguments of selected table struct as well formatted on the screen to show or help to select rows etc.

14-) void delete_table(DATABASE selected_db);**
It just takes selected database struct array(1 elemented) address as a parameter. It controls table existence of selected database if not exist give attention to user and returns menu. If table exists takes the index of table that will be delete from user. Then it shifts row and reallocates tablelist array. Then it creates new tablelist file compatible with chosen formatted name. After all, it removes table file from hard drive by passing file name of table to remove(); function included in "stdio.h" library in C.

15-)void insert_row(TABLE selected_table,DATABASE** selected_db);**
It takes selected database(DATABASE typed) struct array(1 elemented) address and selected table (TABLE typed) struct array(1 elemented) address as a parameters. Then it takes index value from user to insertion operation. Then it re-allocates table in memory and takes inputs from the last row.While inputs that will be insert taking from user it controls primary key and format compatibility(for integer,double,boolean). After getting inputs step finished it inserts row to the taken row index from the user using designed algorithm by myself. It takes created row to temp and shifts row from given index to end then it assigns temporary row to the given indexed row. Then it calls **save_table function** to save changes in file of table and it calls **display_table function** to show operation results to the user.

16-) int* select_rows(int max_row,int* row_count);
It takes row size, row count address from used functions **update_rows** and **delete_rows** then it **returns selected index array** that contains controlled index values taken from user in it. After all, it prints on the screen the message that contains operation success condition.

17-) update_rows(TABLE selected_table,DATABASE** selected_db);**
It takes selected database(DATABASE typed) struct array(1 elemented) address and selected table (TABLE typed) struct array(1 elemented) address as a parameters. This function uses **display_table function** to show the selected table condition to user and asks user to determine the method of selection by using **where_function or select_rows function.** After the updating operations compatible with type matching and primary key condition if exists(the breakpoint in that operation is same row values can be acceptable for columns that has primary key feature), it uses **save_table function** to apply changes to table file.

18-) void save_table(TABLE selected_table,DATABASE** selected_db);**
It takes selected database(DATABASE typed) struct array(1 elemented) address and selected table (TABLE typed) struct array(1 elemented) address as a parameters. Then it helps to save processed table in memory to name formatted table.txt file. Then it closes file at the end.

19-) int* where_function(TABLE selected_table,int *row_count);**
It takes selected table (TABLE typed) struct array(1 elemented) address and row count address as a parameter from used functions **update_rows** and **delete_rows**. After it takes column index and keyword that will be search in table, It assigns the matchings(using **strcmp** function) to the reallocated (if matching found) dynamic index array. After operations completed it return selected rows array that contains matching index values.

20-) void delete_rows(TABLE selected_table,DATABASE** selected_db);**
It takes selected database(DATABASE typed) struct array(1 elemented) address and selected table (TABLE typed) struct array(1 elemented) address as a parameters. This function uses **display_table function** to show the selected table condition to user and asks user to determine the method of selection by using **where_function or select_rows function.** (The breakpoint of that function is the condition that user's manual row selection may not be sorted. It can change algorithm because if row selection is not sorted, deleting algorith that i designed lost the given index. The deleting algorithm work like so : It shifts all arguments from end until given index so it have to start to delete indexes from descending order of selected indexes. So, I used selection_sort that named in program **Sel_sort function** because it is most convenient sorting algorithm for this operation that i experienced and observed until now.) At last, it uses **save_table function** to apply changes to table file and uses **display_table function** to show operation results to the user.

21-) void Sel_sort(int *arr,int size);
It takes array address and array size as a parameter. Then it sorts array and returns sorted array. For the working principle of algorithms : https://en.wikipedia.org/wiki/Selection_sort link has enough information included space and performance case complexities of algorithm.

DESCRIPTION

In this Project, the capabilities shown in main menu then user select below mentioned operations (it means capabilities of functions) and it applies operations and save the results in formatted named files.

OPERATION LIST:

1.DISPLAY LIST OF DATABASES
2.SELECT ONE OF THE LISTED DATABASES
3.CREATE NEW DATABASE
4.CREATE NEW TABLE WITH COLUMNS
5.DISPLAY THE LIST OF TABLES IN SELECTED DATABASE
6.DELETE SELECTED TABLE AND THEIR CONTENTS
7.INSERT A ROW TO A SELECTED TABLE
8.SELECT & UPDATE ROW(S) FROM A SELECTED TABLE(INCLUDED WHERE FUNCTIONALITY)
9.SELECT & DELETE ROW(S) IN A SELECTED TABLE(INCLUDED WHERE FUNCTIONALITY)

- Above mentioned functions have primary key control, type matching control, same named argument existence control, correct index and choice controls, where functionality to search keyword in given table.

- I used two struct to implement complementary development capability for solution. Listed below:

//STRUCT USED FOR TABLE CONTENTS PROCESSING

```
typedef struct{
        char table_name[NAME_LENGTH];//KEEPS NEME OF TABLE
        char*** table;    //KEEPS TABLE CONTENTS
        int row,column; //KEEPS ROW AND COLUMN NUMBER OF TABLE
        int *table_types; //KEEPS TYPES OF COLUMNS ARRAY'S ADDRESS
        int *p_keys;//KEEPS PRIMARY KEY EXISTENCE FEATURE OF COLUMNS ARRAY'S ADDRESS
        char* headline;//KEEPS COLUMN NAMES ADDRESS
}TABLE;
```

```
//STRUCT USED FOR REACHING TABLE FILE NAMES AND TABLELIST PROCESSING IN SELECTED DATABASE
typedef struct{
        char dbname[NAME_LENGTH];//KEEPS DATABASE NAMA
        int table_num;//KEEPS TABLE NUM
        char** table_list;//KEEPS TABLE LIST IN DATABASE
}DATABASE;
```

- In addition, I defined MACROS named LINE_LENGTH and NAME_LENGTH for require flexibility of Project.
- Project operations designed based on memory usage (because Project assigned as **lightweight**) so, works faster than designs based file operations reaches file permanently. But the cons is memory is limited so it is not convenient for big sized table operations and primary key controls. I preferred memory based because of primary control in update rows, insert rows and create table operations.

-Program may be improved by adding or exchanging more hybrid processing modules.

-In this Project; arrays, nested loops, macros, structures, pointer arrays(3 dimensional), functions, global and const variables, file operations, string functions employed as features of C Programming Language related with course.

SCREENSHOTS

1. FIRST OPEN AND DISPLAY TABLE & 3.CREATE DATABASE

```
HELLO!!!
EXISTING DATABASE FILE NOT FOUND.
PROBABLY FIRST USE OF PROGRAM AND THERE IS NO DATABASE CREATED UNTIL NOW...



PLEASE ENTER YOUR CHOICE NUMBER LISTED BELOW
1.DISPLAY LIST OF DATABASES
2.SELECT ONE OF THE LISTED DATABASES
3.CREATE NEW DATABASE
4.CREATE NEW TABLE WITH COLUMNS
5.DISPLAY THE LIST OF TABLES IN SELECTED DATABASE
6.DELETE SELECTED TABLE AND THEIR CONTENTS
7.INSERT A ROW TO A SELECTED TABLE
8.SELECT & UPDATE ROW(S) FROM A SELECTED TABLE
9.SELECT & DELETE ROW(S) IN A SELECTED TABLE

ENTER '0' TO QUIT PROGRAM

Choice :

DATABASE LIST
---------------
NO DATABASE
Press any key to continue . . .



PLEASE ENTER NEW DATABASE NAME : PROJE
DATABASE << PROJE >> CREATED SUCCESFULLY...
Press any key to continue . . .


DATABASE LIST
---------------
1-) --PROJE--
Press any key to continue . . .
```

2-)SELECT DATABASE

```
LAST CHOSEN DATABASE :   LAST CHOSEN TABLE :

PLEASE ENTER YOUR CHOICE NUMBER LISTED BELOW
1.DISPLAY LIST OF DATABASES
2.SELECT ONE OF THE LISTED DATABASES
3.CREATE NEW DATABASE
4.CREATE NEW TABLE WITH COLUMNS
5.DISPLAY THE LIST OF TABLES IN SELECTED DATABASE
6.DELETE SELECTED TABLE AND THEIR CONTENTS
7.INSERT A ROW TO A SELECTED TABLE
8.SELECT & UPDATE ROW(S) FROM A SELECTED TABLE
9.SELECT & DELETE ROW(S) IN A SELECTED TABLE

ENTER '0' TO QUIT PROGRAM

Choice : 2
DATABASE LIST
---------------
1-) --PROJE--


PLEASE ENTER INDEX OF DATABASE THAT WILL BE SELECTED : 1


DATABASE PROJE IS SELECTED SUCCESSFULLY


Press any key to continue . . .
```

4-)CREATING TABLE

```
DATABASE : PROJE        TABLE : SCHOOL
ENTER COLUMN NUMBER : 5
ENTER ROW NUMBER : 4

ENTER THE NAME OF COLUMN 1 : NAME

CHOOSE TYPE OF COLUMN 1
(1.INTEGER,2.DOUBLE,3.STRING,4.BOOLEAN(ONE CHARACTER 'T' OR 'F')) : 3
ENTER 1 TO ADD PRIMARY KEY, 0 TO CONTINUE :0

ENTER THE NAME OF COLUMN 2 : SURNAME

CHOOSE TYPE OF COLUMN 2
(1.INTEGER,2.DOUBLE,3.STRING,4.BOOLEAN(ONE CHARACTER 'T' OR 'F')) : 3
ENTER 1 TO ADD PRIMARY KEY, 0 TO CONTINUE :1

ENTER THE NAME OF COLUMN 3 : ID

CHOOSE TYPE OF COLUMN 3
(1.INTEGER,2.DOUBLE,3.STRING,4.BOOLEAN(ONE CHARACTER 'T' OR 'F')) : 1
ENTER 1 TO ADD PRIMARY KEY, 0 TO CONTINUE :1

ENTER THE NAME OF COLUMN 4 : AVERAGE

CHOOSE TYPE OF COLUMN 4
(1.INTEGER,2.DOUBLE,3.STRING,4.BOOLEAN(ONE CHARACTER 'T' OR 'F')) : 4
ENTER 1 TO ADD PRIMARY KEY, 0 TO CONTINUE :0

ENTER THE NAME OF COLUMN 5 : BOOL

CHOOSE TYPE OF COLUMN 5
(1.INTEGER,2.DOUBLE,3.STRING,4.BOOLEAN(ONE CHARACTER 'T' OR 'F')) : 4
ENTER 1 TO ADD PRIMARY KEY, 0 TO CONTINUE :0
```

```
row : 3 column : SURNAME
value :
TYPE: STRING
SAGLAM

Input exists.Column has primary key.Enter new input : BILINMEZEDOGRU

row : 3 column : ID
value :
TYPE: INTEGER
15011804

Input exists.Column has primary key.Enter new input : ADDADDAFWE

just digits allowed please enter again :
```

```
row : 2 column : RATE
value :
TYPE: DOUBLE
GFJSHDKJLK

just digits allowed please enter again :

just digits allowed please enter again : 21312

row : 4 column : RATE
value :
TYPE: DOUBLE
34.34

Input exists.Column has primary key.Enter new input :
```

```
        ---------TABLE SCHOOL CREATED SUCCESFULLY-------------
file name : PROJE_tablelist.txt Press any key to continue . . .
```

5-)DISPLAY TABLE

```
DATABASE:PROJE  TABLE:SCHOOL
Row :4 Column :5

CHARACTER : 3  3  1  4  4
1-INTEGER,2-DOUBLE,3-STRING,4-BOOL

PRIMARY KEYS :  0  1  1  0  0
1-YES,0-NO
```

	NAME	SURNAME	ID	AVERAGE	BOOL
1.	YASIN	SAGLAM	15011804	T	T
2.	EMRE	DEMIRBASOGLU	54324234	T	T
3.	ISMAIL	BILINMEZEDOGRU	5432123	T	F
4.	AMAC	GUVENSAN	987654	T	T

```
Press any key to continue . . .
```

6-)DELETE TABLE

```
TABLE LIST OF DATABASE : PROJE
1. SCHOOL
2. TRY
ENTER INDEX OF TABLE TO DELETE ('0' TO RETURN MENU): 2
```

```
TABLE LIST OF DATABASE : PROJE
1. SCHOOL
2. TRY
ENTER INDEX OF TABLE TO DELETE ('0' TO RETURN MENU): 2
Deleted file : TRY_PROJE_table.txt
Press any key to continue . . .
file name : PROJE_tablelist.txt
Press any key to continue . . .
```

```
TABLE LIST OF DATABASE : PROJE
1. SCHOOL
Please enter index number to select table :
```

7-)INSERT ROW

```
DATABASE:PROJE  TABLE:SCHOOL
Row :5 Column :5

CHARACTER : 3  3  1  4  4
1-INTEGER,2-DOUBLE,3-STRING,4-BOOL

PRIMARY KEYS :  0  1  1  0  0
1-YES,0-NO


       NAME                SURNAME             ID                  AVERAGE             BOOL
-------------------------------------------------------------------------------------------------------
1.     YASIN               SAGLAM              15011804            T                   T
2.     UZUNISIMLIBIRI      BILMEMNE            2222222             T                   T
3.     EMRE                DEMIRBASOGLU        54324234            T                   T
4.     ISMAIL              BILINMEZEDOGRU      5432123             T                   F
5.     AMAC                GUVENSAN            987654              T                   T

PLEASE ENTER THE ROW INDEX TO INSERT ROW IN A TABLE : SCHOOL
ROW INDEX(ENTER 0 TO RETURN): 3

HEADLINE IS :
NAME                SURNAME             ID                  AVERAGE             BOOL
TYPE: STRING
KISAAD
TYPE: STRING
NEKI
TYPE: INTEGER
2222222

Input exists.Column has primary key.Enter new input : 3333333
TYPE: BOOL(ONE CHARACTER)
TTYPE: BOOL(ONE CHARACTER)
```

```
       NAME                SURNAME             ID                  AVERAGE             BOOL
-------------------------------------------------------------------------------------------------------
1.     YASIN               SAGLAM              15011804            T                   T
2.     UZUNISIMLIBIRI      BILMEMNE            2222222             T                   T
3.     KISAAD              NEKI                3333333             T                   T
4.     EMRE                DEMIRBASOGLU        54324234            T                   T
5.     ISMAIL              BILINMEZEDOGRU      5432123             T                   F
6.     AMAC                GUVENSAN            987654              T                   T

GIVEN ROW INSERTED SUCCESFULLY TO GIVEN INDEX...
Press any key to continue . . .
```

8-)UPDATE ROW(S)

MANUAL SELECTION

```
DATABASE:MAG    TABLE:NUMANBABA
Row :4 Column :4

CHARACTER : 3  3  1  4
1-INTEGER,2-DOUBLE,3-STRING,4-BOOL

PRIMARY KEYS :  1  0  1  0
1-YES,0-NO


        AD                  SOYAD               TC                  EVLI
-------------------------------------------------------------------------------
1.      YASIN               SAGLAM              344354              T
2.      UZUNBIRADIMVAR      SOYADIMDAOYLE       12343234            T
3.      EMRE                DEMIR               32421               F
4.      ALEX                DESOUZA             324342              T
Press any key to continue . . .

SELECT METHOD FOR UPDATE(ENTER 0 TO RETURN) :
1.WHERE FUNCTION
2.MANUEL SELECTION
2

HOW MANY ROWS WILL SELECT : 2
ENTER 1th row index: 2
ENTER 2th row index: 4

ROWS SELECTED SUCCESFULLY...
Press any key to continue . . .
```

```
        AD                  SOYAD               TC                  EVLI
-------------------------------------------------------------------------------
1.      YASIN               SAGLAM              344354              T
2.      KISALTTIM           SOYADIDA            12343234            F
3.      EMRE                DEMIR               32421               F
4.      ALEX                DESOUZA             324342              T

ROW: 4 COLUMN: 1
TYPE: STRING
```

```
        AD                  SOYAD               TC                  EVLI
-------------------------------------------------------------------------------
1.      YASIN               SAGLAM              344354              T
2.      KISALTTIM           SOYADIDA            12343234            F
3.      EMRE                DEMIR               32421               F
4.      JEREMAIN            LENS                77                  T

ROW(S) UPDATED SUCCESFULLY...
Press any key to continue . . .
```

```
        AD                  SOYAD               TC                  EVLI
-------------------------------------------------------------------------------
1.      YASIN               SAGLAM              344354              T
2.      KISALTTIM           SOYADIDA            12343234            F
3.      EMRE                DEMIR               32421               F
4.      JEREMAIN            LENS                77                  T

SELECTED ROWS TO UPDATE :  3

ROW: 3 COLUMN: 1
TYPE: STRING
YASIN

Input exists.Column has primary key.Enter new input :
```

```
ROW: 3 COLUMN: 3
TYPE: INTEGER
DAKJFADSG

just digits allowed please enter again :
```

WHERE FUNCTIONALITY

```
DATABASE:MAG    TABLE:NUMANBABA
Row :6 Column :4

CHARACTER : 3  3  1  4
1-INTEGER,2-DOUBLE,3-STRING,4-BOOL

PRIMARY KEYS :  1  0  1  0
1-YES,0-NO


        AD                  SOYAD               TC                  EVLI
-------------------------------------------------------------------------------
1.      SELMA               SAGLAM              212121              T
2.      NUMAN               KAZANCI             21334               T
3.      YASIN               SAGLAM              3434                F
4.      AHMET               SAGLAM              1234213             T
5.      EMRE                DEMIR               32421               F
6.      VELI                BEYOGLU             1232                T
Press any key to continue . . .

SELECT METHOD FOR UPDATE(ENTER 0 TO RETURN) :
1.WHERE FUNCTION
2.MANUEL SELECTION
1

ENTER COLUMN INDEX TO SEARCH KEYWORD: 2

PLEASE ENTER KEYWORD FOR SEARCH & ROW SELECTION :SAGLAM
```

```
        AD                  SOYAD               TC                  EVLI
-------------------------------------------------------------------------------
1.      SELMA               SAGLAM              212121              T
2.      NUMAN               KAZANCI             21334               T
3.      YASIN               SAGLAM              3434                F
4.      AHMET               SAGLAM              1234213             T
5.      EMRE                DEMIR               32421               F
6.      VELI                BEYOGLU             1232                T

SELECTED ROWS TO UPDATE :  1  3  4

ROW: 1 COLUMN: 1
TYPE: STRING
```



```
        AD                  SOYAD               TC                  EVLI
-------------------------------------------------------------------------------
1.      SELMA               SAGLAM              212132              T
2.      NUMAN               KAZANCI             21334               T
3.      YASIN               SAGLAM              344354              T
4.      HUSEYIN             TANIMIYORUM         3243243             F
5.      EMRE                DEMIR               32421               F
6.      VELI                BEYOGLU             1232                T

ROW(S) UPDATED SUCCESFULLY...
Press any key to continue . . .
```

9-)DELETE ROWS

WHERE FUNCTIONALITY

```
DATABASE:MAG    TABLE:NUMANBABA
Row :6 Column :4

CHARACTER : 3  3  1  4
1-INTEGER,2-DOUBLE,3-STRING,4-BOOL

PRIMARY KEYS :  1  0  1  0
1-YES,0-NO


        AD                  SOYAD               TC                  EVLI
-------------------------------------------------------------------------------
1.      SELMA               SAGLAM              212132              T
2.      NUMAN               KAZANCI             21334               T
3.      YASIN               SAGLAM              344354              T
4.      HUSEYIN             TANIMIYORUM         3243243             F
5.      EMRE                DEMIR               32421               F
6.      VELI                BEYOGLU             1232                T

SELECT METHOD FOR UPDATE :
1.WHERE FUNCTION
2.MANUEL SELECTION
1

ENTER COLUMN INDEX TO SEARCH KEYWORD: 1

PLEASE ENTER KEYWORD FOR SEARCH & ROW SELECTION :HUSEYIN
```

```
        AD                  SOYAD               TC                  EVLI
-------------------------------------------------------------------------------
1.      SELMA               SAGLAM              212132              T
2.      NUMAN               KAZANCI             21334               T
3.      YASIN               SAGLAM              344354              T
4.      HUSEYIN             TANIMIYORUM         3243243             F
5.      EMRE                DEMIR               32421               F
6.      VELI                BEYOGLU             1232                T

SELECTED ROWS TO DELETE :  4
Press any key to continue . . .
```

```
DATABASE:MAG    TABLE:NUMANBABA
Row :5 Column :4

CHARACTER : 3  3  1  4
1-INTEGER,2-DOUBLE,3-STRING,4-BOOL

PRIMARY KEYS :  1  0  1  0
1-YES,0-NO


        AD                  SOYAD               TC                  EVLI
-------------------------------------------------------------------------------
1.      SELMA               SAGLAM              212132              T
2.      NUMAN               KAZANCI             21334               T
3.      YASIN               SAGLAM              344354              T
4.      EMRE                DEMIR               32421               F
5.      VELI                BEYOGLU             1232                T

ROW(S) DELETED SUCCESFULLY...
Press any key to continue . . .
```

MANUAL SELECTION

```
DATABASE:MAG    TABLE:NUMANBABA
Row :5 Column :4

CHARACTER : 3  3  1  4
1-INTEGER,2-DOUBLE,3-STRING,4-BOOL

PRIMARY KEYS :  1  0  1  0
1-YES,0-NO


        AD                      SOYAD                   TC                      EVLI
-------------------------------------------------------------------------------------
1.      AHMET                   SAGLAM                  123213                  T
2.      NUMAN                   KAZANCI                 21334                   T
3.      YASIN                   SAGLAM                  344354                  T
4.      EMRE                    DEMIR                   32421                   F
5.      VELI                    KASIMPASA               54342                   F

SELECT METHOD FOR UPDATE :
1.WHERE FUNCTION
2.MANUEL SELECTION
2

HOW MANY ROWS WILL SELECT : 3
ENTER 1th row index: 5
ENTER 2th row index: 1
ENTER 3th row index: 2

ROWS SELECTED SUCCESFULLY...
Press any key to continue . . .
```

```
SELECTED ROWS TO DELETE :  1  2  5
Press any key to continue . . .
```

```
DATABASE:MAG    TABLE:NUMANBABA
Row :2 Column :4

CHARACTER : 3  3  1  4
1-INTEGER,2-DOUBLE,3-STRING,4-BOOL

PRIMARY KEYS :  1  0  1  0
1-YES,0-NO


        AD                      SOYAD                   TC                      EVLI
-------------------------------------------------------------------------------------
1.      YASIN                   SAGLAM                  344354                  T
2.      EMRE                    DEMIR                   32421                   F

ROW(S) DELETED SUCCESFULLY...
Press any key to continue . . .
```

REFERENCES:

1-PROGRAMMING LANGUAGES LECTURE DOCUMENTS

2- http://msdis.missouri.edu/resources/gis_advanced/pdf/relational.pdf

3- https://www.tutorialspoint.com

SOURCE CODE

```
/**
@file
Programming Languages Lecture(Fall-2016)---- TermProject.
Relational Database Management System designed as a project.
In this Project, the capabilities shown in main menu then user select below mentioned
operations(it means capabilities of functions) and
it applies operations and save the results in formatted named files.

OPERATION LIST:

1.DISPLAY LIST OF DATABASES
2.SELECT ONE OF THE LISTED DATABASES
3.CREATE NEW DATABASE
4.CREATE NEW TABLE WITH COLUMNS
5.DISPLAY THE LIST OF TABLES IN SELECTED DATABASE
6.DELETE SELECTED TABLE AND THEIR CONTENTS
7.INSERT A ROW TO A SELECTED TABLE
8.SELECT & UPDATE ROW(S) FROM A SELECTED TABLE(INCLUDED WHERE FUNCTIONALITY)
9.SELECT & DELETE ROW(S) IN A SELECTED TABLE(INCLUDED WHERE FUNCTIONALITY)

-   Above mentioned functions have primary key control, type matching control,
same named argument existence control, correct index and choice controls, where
functionality to search keyword in given table.


@author

Name                :Muhammed Yasin SAGLAM
Student No          :15011804
Date                :11/01/2017
E-Mail              :myasinsaglam1907@gmail.com
Compiler Used       :GCC
IDE                 :DEV-C++(Version 5.11)
Operating System    :Windows Embedded 10 (64-BIT)
*/

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include <stdbool.h>

#define LINE_LENGTH 255
#define NAME_LENGTH 30
char const filename[]="Databases.txt";

//STRUCT USED FOR TABLE CONTENTS PROCESSING
typedef struct{
    char table_name[NAME_LENGTH];//KEEPS NEME OF TABLE
    char*** table;  //KEEPS TABLE CONTENTS
    int row,column; //KEEPS ROW AND COLUMN NUMBER OF TABLE
    int *table_types; //KEEPS TYPES OF COLUMNS ARRAY'S ADDRESS
    int *p_keys;//KEEPS PRIMARY KEY EXISTENCE FEATURE OF COLUMNS ARRAY'S ADDRESS
    char* headline;//KEEPS COLUMN NAMES ADDRESS
}TABLE;
//STRUCT USED FOR REACHING TABLE FILE NAMES AND TABLELIST PROCESSING IN SELECTED
DATABASE
typedef struct{
    char dbname[NAME_LENGTH];//KEEPS DATABASE NAMA
    int table_num;//KEEPS TABLE NUM
    char** table_list;//KEEPS TABLE LIST IN DATABASE
}DATABASE;


void print_menu();
void create_database(DATABASE** db,int *database_num);
```

```
int display_database(DATABASE** db,int *database_num);
void select_database(DATABASE** db,int *database_num,DATABASE** select);
void save_file(FILE **fp,DATABASE** db,int *database_num);
void read_file(FILE **fp,DATABASE** db,int *database_num);
char* create_table(DATABASE** select,TABLE** selected_table);
int display_tablelist(DATABASE** select);
void add_table_to_list(DATABASE ** select);
int load_tablelst(DATABASE** selected_db);
void select_table(DATABASE** selected_db,TABLE **selected_table);
void load_table(TABLE** selected,DATABASE** selectedb);
void display_table(TABLE** selected,DATABASE** selected_db);
void delete_table(DATABASE** selected_db);
void insert_row(TABLE** selected_table,DATABASE** selected_db);
int* select_rows(int max_row,int* row_count);
void update_rows(TABLE** selected_table,DATABASE** selected_db);
void save_table(TABLE** selected_table,DATABASE** selected_db);
int* where_function(TABLE** selected_table,int *row_count);
void delete_rows(TABLE** selected_table,DATABASE** selected_db);
void Sel_sort(int *arr,int size);

int main(){

system("color a");
DATABASE *db,*selected_db;
db=(DATABASE*)calloc(1,sizeof(DATABASE)*1);
selected_db=(DATABASE*)malloc(sizeof(DATABASE)*1);

TABLE *selected_table;
selected_table=(TABLE*)calloc(1,sizeof(TABLE));

int database_num=0;
int choice;

char *table_fname;
table_fname=(char*)calloc((NAME_LENGTH),sizeof(char));
FILE *tablelist;
FILE *file_r;
file_r=fopen(filename,"r+");
if(!file_r){
    printf("HELLO!!!\nEXISTING DATABASE FILE NOT FOUND.\nPROBABLY FIRST USE OF PROGRAM
AND THERE IS NO DATABASE CREATED UNTIL NOW...\n\n\n\n");
    file_r=fopen(filename,"w+");
        if(!file_r){
            printf("FILE NOT OPENED. QUITTING...");
            exit(0);
        }
}
else{
    read_file(&file_r,&db,&database_num);
}
    strcpy(selected_db[0].dbname,"");
    print_menu();
    printf("\nChoice : ");
    scanf("%d",&choice);
    while(choice!=0){//MENU DESIGN

            if(choice==3){//CREATE DATABASE
                create_database(&db,&database_num);
                save_file(&file_r,&db,&database_num);
            }
            if(choice==1){//DISPLAY DATABASES
                display_database(&db,&database_num);
                system("PAUSE");
            }
            if(choice==2){//SELECT DATABASE
                select_database(&db,&database_num,&selected_db);
            }
```

```
            if(choice==4){//CONTROLS IF DATABASE SELECTED OR NOT THEN IT CREATES TABLE
AFTER SELECTION
                system("CLS");
                if(display_database(&db,&database_num)){
                    if(strcmp(selected_db[0].dbname,"")==0){
                        printf("\n\tPLEASE SELECT DATABASE TO ADD TABLE IN IT\n");
                        system("PAUSE");
                        select_database(&db,&database_num,&selected_db);
                    }
                strcpy(table_fname,create_table(&selected_db,&selected_table));

                char temp_dbname[NAME_LENGTH];
                strcpy(temp_dbname,selected_db[0].dbname);
                strcat(temp_dbname,"_tablelist.txt");
                add_table_to_list(&selected_db);
                tablelist=fopen(temp_dbname,"a+");//FILE OPENS IN APPEND MODE TO ADD
TABLENAME TO TABLELIST FILE IT MAY BE DONE IN FUNCTION. SHOULD BE IMPROVED...
                if(!tablelist){
                    tablelist=fopen(temp_dbname,"w+");
                    fprintf(tablelist,"%s \n",table_fname);
                }
                else{
                    fprintf(tablelist,"%s \n",table_fname);
                }
                    fclose(tablelist);
                    load_tablelst(&selected_db);
                }
            }
            if(choice==5){//DISPLAY TABLE LIST THEN SELECT TABLE AND DISPLAY CONTENTS
OF TABLE
                system("CLS");
                if(strcmp(selected_db[0].dbname,"")==0){
                    printf("\n\tPLEASE SELECT DATABASE TO DISPLAY TABLES IN IT\n");
                    system("PAUSE");
                    select_database(&db,&database_num,&selected_db);
                }
                if(display_tablelist(&selected_db)){//TABLE EXISTENCE CONTROL
                    select_table(&selected_db,&selected_table);
                    load_table(&selected_table,&selected_db);
                    display_table(&selected_table,&selected_db);
                }
                system("PAUSE");
                system("CLS");
            }
            if(choice==6){//DELETE SELECTED TABLE AND THEIR CONTENTS
                if(strcmp(selected_db[0].dbname,"")==0){//SELECTED DATABASE CONTROL
                    printf("\n\tPLEASE SELECT DATABASE TO DISPLAY TABLES IN IT\n");
                    system("PAUSE");
                    select_database(&db,&database_num,&selected_db);
                }
                load_tablelst(&selected_db);
                system("CLS");
                delete_table(&selected_db);
                system("color a");
            }
            if(choice==7){//INSERTING ROW STEP
                if(strcmp(selected_db[0].dbname,"")==0){//SELECTED DATABASE CONTROL
                    printf("\n\tPLEASE SELECT DATABASE TO DISPLAY TABLES IN IT\n");
                    system("PAUSE");
                    select_database(&db,&database_num,&selected_db);
                }
                if(display_tablelist(&selected_db)){//TABLE EXISTENCE CONTROL
DISPLAY_TABLELIST FUNCTION RETURNS 1 IF TABLE EXISTS
                    select_table(&selected_db,&selected_table);
                    load_table(&selected_table,&selected_db);
                    display_table(&selected_table,&selected_db);
                    insert_row(&selected_table,&selected_db);
```

```
                }

            system("PAUSE");
            }
            if(choice==8){//UPDATE ROWS PART
                if(strcmp(selected_db[0].dbname,"")==0){//SELECTED DATABASE CONTROL
                    printf("\n\tPLEASE SELECT DATABASE TO DISPLAY TABLES IN IT\n");
                    system("PAUSE");
                    select_database(&db,&database_num,&selected_db);
                }
                if(display_tablelist(&selected_db)){//TABLE EXISTENCE CONTROL DISPLAY
TABLELIST_FUNCTION RETURNS 1 IF TABLE EXISTS
                    select_table(&selected_db,&selected_table);
                    load_table(&selected_table,&selected_db);
                    display_table(&selected_table,&selected_db);
                    update_rows(&selected_table,&selected_db);
                }

            system("PAUSE");
            }
            if(choice==9){//DELETE ROWS PART
                if(strcmp(selected_db[0].dbname,"")==0){//SELECTED DATABASE CONTROL
                    printf("\n\tPLEASE SELECT DATABASE TO DISPLAY TABLES IN IT\n");
                    system("PAUSE");
                    select_database(&db,&database_num,&selected_db);
                }
                if(display_tablelist(&selected_db)){//TABLE EXISTENCE CONTROL DISPLAY
TABLELIST_FUNCTION RETURNS 1 IF TABLE EXISTS
                    select_table(&selected_db,&selected_table);
                    load_table(&selected_table,&selected_db);
                    display_table(&selected_table,&selected_db);
                    delete_rows(&selected_table,&selected_db);
                }

            system("PAUSE");
            }
        system("CLS");
        system("color a");
        printf("\nLAST CHOSEN DATABASE :%s\t LAST CHOSEN TABLE :
%s\n",selected_db[0].dbname,selected_table[0].table_name);
        print_menu();
        printf("\nChoice : ");
        scanf("%d",&choice);
        if(choice==0){
            system("CLS");
            printf("GOOD BYE....\n");
            save_file(&file_r,&db,&database_num);
        }
    }

    free(db);//FREE
    system("PAUSE");
    return 0;
}

//CREATES DATABASE IN MEMORY
void create_database(DATABASE** db,int *database_num){
    system("CLS");
    system("color b");
    char *db_name,*tempn1,*tempn2;
    int i;
    bool name_control=false;

    db_name=(char*)malloc(sizeof(char)*(NAME_LENGTH));
    tempn1=(char*)malloc(sizeof(char)*(NAME_LENGTH));// temporary name variable to
control case sensitivity for win OS
```

```
    tempn2=(char*)malloc(sizeof(char)*(NAME_LENGTH)); // temporary name variable to
control case sensitivity for win OS
        if(!db_name){
            printf("ALLOCATION ERROR OCCURED. QUITTING...");
            exit(0);
        }

    //control same named database existence and case sensitivity
    while(name_control==false){
        printf("\nPLEASE ENTER NEW DATABASE NAME : ");
        scanf("%s",db_name);
        strcpy(tempn2,db_name);
        name_control=true;
        for(i=0;i<(*database_num);i++){
            strcpy(tempn1,((*db)[i].dbname));
            if(strcmp(strlwr(tempn1),strlwr(tempn2))==0){
                name_control=false;
            }
        }
        if(name_control==false){
            printf("\nDatabase created before....");
        }
    }

    //increment total database number
    (*database_num)++;
    (*db)=(DATABASE*)realloc((*db),sizeof(DATABASE)*(*database_num));
        if(!(*db)){
            printf("\nDatabase not created. Quitting...");
            exit(0);
        }
    strcpy((*db)[(*database_num-1)].dbname,db_name);
    printf("DATABASE << %s >> CREATED SUCCESFULLY...\n",db_name);
    system("PAUSE");
    strcat(db_name,"_tablelist.txt");
    FILE *t_list;
    t_list=fopen(db_name,"w");
        if(!t_list){
            printf("\nError occured. There is not enough disk space for database table
list file.Quitting");
            exit(0);
        }
    fprintf(t_list,"%d \n",0);
    fclose(t_list);
    free(db_name);
    free(tempn1);
    free(tempn2);
    system("color a");
}

//PRINTS DATABASE LIST ON SCREEN
int display_database(DATABASE** db,int *database_num){
    system("CLS");
    int i;
    printf("DATABASE LIST\n---------------\n");
    if((*database_num)==0){
        printf("NO DATABASE\n");
        system("PAUSE");
        return 0;
    }
    for(i=0;i<(*database_num);i++){
        printf("%d-) --%s-- \n",i+1,(*db)[i].dbname);
    }
    return 1;
}

//SELECTS DATABASE FROM DATABASE LIST
```

```
void select_database(DATABASE** db,int *database_num,DATABASE** select){
    int index;
    system("color b");
    display_database(db,database_num);
    if((*database_num)!=0){
        printf("\n\n\nPLEASE ENTER INDEX OF DATABASE THAT WILL BE SELECTED : ");
        scanf("%d",&index);
        while(index<=0 || index>(*database_num)){
            printf("\n\nPLEASE ENTER RIGHT INDEX VALUE FOR GIVEN DATABASE LIST : ");
            scanf("%d",&index);
            system("CLS");
            display_database(db,database_num);
        }
        (**select)=(*db)[index-1];
        printf("\n\n\nDATABASE %s IS SELECTED SUCCESSFULLY\n\n\n",(**select).dbname);
        system("PAUSE");
        load_tablelst(select);//loading table list to DATABASE STRUCT element that
named selected database
    }
    else{
        printf("\n\nTHERE IS NO DATABASE TO SELECT \n\n\n");
        return;
        system("PAUSE");
    }
    system("color a");
}

//PRINTS MAIN MENU
void print_menu(){
printf("\nPLEASE ENTER YOUR CHOICE NUMBER LISTED BELOW \n1.DISPLAY LIST OF
DATABASES\n2.SELECT ONE OF THE LISTED DATABASES\n3.CREATE NEW DATABASE ");
printf("\n4.CREATE NEW TABLE WITH COLUMNS\n5.DISPLAY THE LIST OF TABLES IN SELECTED
DATABASE\n6.DELETE SELECTED TABLE AND THEIR CONTENTS");
printf("\n7.INSERT A ROW TO A SELECTED TABLE\n8.SELECT & UPDATE ROW(S) FROM A SELECTED
TABLE");
printf("\n9.SELECT & DELETE ROW(S) IN A SELECTED TABLE\n\nENTER '0' TO QUIT PROGRAM
\n");
}

//SAVES DATABASES TO TXT FILE
void save_file(FILE **fp,DATABASE** db,int *database_num){
    int i;
    (*fp)=fopen(filename,"w");
        if(!(*fp)){
            printf("\nFILE NOT SAVED BEFORE QUIT...\n");
            exit(0);
        }
    fprintf((*fp),"%d ",(*database_num));
    for(i=0;i<(*database_num);i++)
        fprintf((*fp),"%s\t ",(*db)[i].dbname);
    fclose((*fp));
}

//READS DATABASES FROM FILE
void read_file(FILE **fp,DATABASE** db,int *database_num){
    int i=0;
    char name[30];
    fscanf((*fp),"%d",&i);
    (*database_num)=i;
    (*db)=(DATABASE*)realloc((*db),sizeof(DATABASE)*(*database_num));
        if(!(*db)){
            printf("\nPLEASE ADD DATABASE... \n");
            system("PAUSE");
            system("CLS");
            return;
        }
    else{
```

```
    for(i=0;i<(*database_num);i++){
        fscanf((*fp),"%s",name);
        strcpy((*db)[i].dbname,name);
    }
}
    fclose((*fp));

}

//CREATE TABLE IN SELECTED DATABASE
char* create_table(DATABASE** select,TABLE** selected_table){
system("color f");
int control;
char *t_name,*tempn1,*tempn2;//temporary variables for table names and controlling
existence;
int i,j,k;
int type_chooser,column,row;
bool name_control=false;
    system("CLS");
    printf("\nDATABASE : %s",(((*select)[0].dbname)));
    t_name=(char*)malloc(sizeof(char)*(NAME_LENGTH));
    tempn1=(char*)malloc(sizeof(char)*(NAME_LENGTH));// temporary name variable to
control case sensitivity for win OS
    tempn2=(char*)malloc(sizeof(char)*(NAME_LENGTH)); // temporary name variable to
control case sensitivity for win OS
            if(!t_name){
            printf("ERROR OCCURED... RETURNING...");
            return;
        }

        //control same named database existence and case sensitivity
    while(name_control==false){
        printf("\nPLEASE ENTER NEW TABLE NAME : ");
        scanf("%s",t_name);
        strcpy(tempn2,t_name);
        name_control=true;
        for(i=0;i<(*select)[0].table_num;i++){
            strcpy(tempn1,((*select)[0].table_list[i]));
            if(strcmp(strlwr(tempn1),strlwr(tempn2))==0){
                name_control=false;
            }
        }
        if(name_control==false){
            printf("\nDatabase created before....\n");
            system("PAUSE");
        }
    }
    system("CLS");
    printf("\nDATABASE : %s TABLE : %s",(((*select)[0].dbname)),t_name);
    ((*select)[0].table_num)++; //increments table count in database struct
    //NAME FORMATTING
    strcat(t_name,"_");
    strcat(t_name,((*select)[0].dbname));
    strcat(t_name,"_table.txt");

    FILE *table_file;
    table_file=fopen(t_name,"w+");

    printf("\nENTER COLUMN NUMBER : "); scanf("%d",&column);
    printf("ENTER ROW NUMBER : "); scanf("%d",&row);
    fprintf(table_file,"%d\t ",column);
    fprintf(table_file,"%d\t ",row);
    putc('\n',table_file);

    char column_names[column][NAME_LENGTH];
    int type_order[column];
    int primary_order[column];
```

```
    for(i=0;i<column;i++){
        printf("\nENTER THE NAME OF COLUMN %d : ",i+1);
        scanf("%s",column_names[i]);
        printf("\nCHOOSE TYPE OF COLUMN %d \n(1.INTEGER,2.DOUBLE,3.STRING,4.BOOLEAN(ONE
CHARACTER 'T' OR 'F')) : ",i+1);
        scanf("%d",&type_order[i]);
        while(type_order[i]<1 || type_order[i]>4 ){
            printf("\nEnter type number in given scale again : ");
            scanf("%d",&type_order[i]);
        }
        printf("ENTER 1 TO ADD PRIMARY KEY, 0 TO CONTINUE :");
        scanf("%d",&primary_order[i]);
        while(primary_order[i]!=1 && primary_order[i]!=0){
            printf("\nEnter primary key option in given scale again : ");
            scanf("%d",&primary_order[i]);
        }
    }

//table_file TYPE CHARACTERISTICS, COLUMN NAMES AND CONTENTS WRITING ON FILE ACCORDING
TO A DETERMINED FORMAT BY PROGRAMMER...
    for(i=0;i<column;i++)
        fprintf(table_file,"%d ",type_order[i]);
    putc('\n',table_file);
    //ADDING PRIMARY KEY FEATURE TO COLUMNS BY USER SELECTION
    for(i=0;i<column;i++)
        fprintf(table_file,"%d ",primary_order[i]);
    putc('\n',table_file);

    for(i=0;i<column;i++){
        fprintf(table_file,"%-20s ",column_names[i]);
        printf("%s\t",column_names[i]);
    }
    printf("\n");
    putc('\n',table_file);

    //CREATING table_file ON MEMORY
    char*** table=(char***)calloc(row,sizeof(char**));
    for(i=0;i<row;i++){
        table[i]=(char**)calloc(column,sizeof(char*));
        for(j=0;j<column;j++){
            table[i][j]=(char*)calloc(NAME_LENGTH,sizeof(char));
          }
    }

    //CONTROL(FOR PRIMARY AND TYPE) INDEX SETTED AS START INDEX '0'
    for(i=0;i<row;i++){
        for(j=0;j<column;j++){
            k=0;
            control=0;
            printf("\nrow : %d column : %s \nvalue : \n",i+1,column_names[j]);
            switch (type_order[j]){

                case 1:
                    printf("TYPE: INTEGER\n");
                    scanf("%s",table[i][j]);
                    do{//PRIMARY AND DIGIT CONTROL
                        while(k<strlen(table[i][j])){//digit control for integer
                            if(!isdigit(table[i][j][k++])){
                                printf("\njust digits allowed please enter again : ");
                                scanf("%s",table[i][j]);
                                k=0;
                            }
                        }
                        control=1;
                        if(primary_order[j]){
                            k=0;
```

```
                        while(k<i){//PRIMARY CONTROL
                            if(strcmp(table[k++][j],table[i][j])==0){
                                control=0;
                                printf("\nInput exists.Column has primary key.Enter
new input : ");
                                scanf("%s",table[i][j]);
                                k=0;
                            }
                        }
                    }
                    k=0;
                }while(control==0);
                fprintf(table_file,"%-20s ",table[i][j]);
                break;


            case 2:
                printf("TYPE: DOUBLE\n");
                scanf("%s",table[i][j]);
                do{
                    while(k<strlen(table[i][j])){//digit control for double
                        if(isalpha(table[i][j][k++])){
                            printf("\njust digits allowed please enter again : ");
                            scanf("%s",table[i][j]);
                            k=0;
                        }
                    }
                    control=1;
                    if(primary_order[j]){
                        k=0;
                        while(k<i){
                            if(strcmp(table[k++][j],table[i][j])==0){
                                control=0;
                                printf("\nInput exists.Column has primary key.Enter
new input : ");
                                scanf("%s",table[i][j]);
                                k=0;
                            }
                        }
                    }
                    k=0;
                }while(control==0);
                fprintf(table_file,"%-20s ",table[i][j]);
                break;

            case 3:
                printf("TYPE: STRING\n");
                scanf("%s",table[i][j]);
                    if(primary_order[j]){
                        while(k<i){
                            if(strcmp(table[k++][j],table[i][j])==0){
                                printf("\nInput exists.Column has primary key.Enter
new input : ");
                                scanf("%s",table[i][j]);
                                k=0;
                            }
                        }
                    }
                k=0;
                fprintf(table_file,"%-20s ",table[i][j]);
                break;

            case 4:
                printf("TYPE: BOOL(ONE CHARACTER)\n");
                table[i][j][0]=getche();
                while(table[i][j][0]!='T' && table[i][j][0]!='F'){
                    printf("\nPlease enter value 'T' OR 'F' ");
```

```
                        table[i][j][0]=getche();
                    }
                    fprintf(table_file,"%-20c",table[i][j][0]);
                    break;
                }
            }
        system("CLS");
        putc('\n',table_file);
    }
    printf("\n\t---------TABLE %s CREATED SUCCESFULLY-------------
\t\n",strtok(t_name,"_"));
    system("color a");
    fclose(table_file);//CLOSING FILE
    free(table);
    return t_name;
}

//INCREMENTS TABLE NUMBER IN FILE
void add_table_to_list(DATABASE ** select){
    char *name;
    int table_number;
    name=(char*)calloc((NAME_LENGTH),sizeof(char));
    strcat(name,(*select)[0].dbname);
    strcat(name,"_tablelist.txt");
    printf("file name : %s ",name);
    system("PAUSE");
    FILE *list;

    list=fopen(name,"r+");
     if(!list){
        printf("list can not opened to add a table in it");
        return;
    }
    fscanf(list,"%d",&table_number);
    rewind(list);
    table_number++;
    fprintf(list,"%d \n",table_number);
    fclose(list);
    free(name);
}

//DISPLAYS LIST OF TABLE
int display_tablelist(DATABASE** selected_db){
    FILE *file_table_list;
    char *name;
    int table_number=1;
    int index=0;
    int i;
        if((*selected_db)[0].table_num==0){
            printf("\nTHERE IS NO TABLE IN SELECTED DATABASE\n");
            return 0;
        }

    printf("\nTABLE LIST OF DATABASE : %s\n",(*selected_db)[0].dbname);
    for(i=0;i<(*selected_db)[0].table_num;i++){
        printf("%d. %s\n",i+1,(*selected_db)[0].table_list[i]);
    }
    return 1;
}

//LOADS LIST OF TABLES FROM FILE
int load_tablelst(DATABASE** selected_db){
    int i;
    char* fname=(char*)calloc(NAME_LENGTH,sizeof(char));
    if(!fname){
        printf("ALLOCATION ERROR!!! RETURNING MENU....\n");
        return;
```

```
    }
    strcat(fname,(*selected_db)[0].dbname);
    strcat(fname,"_tablelist.txt");
    FILE *ftable_list;
    ftable_list=fopen(fname,"r");
        if(!ftable_list){
            printf("There is no table in selected database : %s\nReturning
menu...\n",(*selected_db)[0].dbname);
            system("PAUSE");
            return ;//IF THERE IS NO TABLE RETURNS MENU
        }
    //table number of table reading from selected database's tablelist file
    fscanf(ftable_list,"%d",&(*selected_db)[0].table_num);
    //ALLOCATING MEMORY FOR SELECTED DATABASE'S TABLELIST ARGUMENTS
    char** list=(char**)calloc((*selected_db)[0].table_num,sizeof(char*));
        for(i=0;i<(*selected_db)[0].table_num;i++){
            list[i]=(char*)calloc(NAME_LENGTH,sizeof(char));
        }
    for(i=0;i<(*selected_db)[0].table_num;i++){
        fscanf(ftable_list,"%s",list[i]);
    }
    (*selected_db)[0].table_list=list;//FREE OPERATION INCLUDED IN FREEING SELECTED
DATABASE STRUCT AT THE END OF PROGRAM
    free(fname);
    fclose(ftable_list);
    //printf("\nTABLELIST LOADED ON MEMORY...\n");
    return 1;
}

//SELECTS TABLE FROM TABLELIST OF SELECTED DATABASE. THEN IT ASSIGN NAME FROM LIST TO
table_name ARGUMENT OF TABLE TYPED SELECETED TABLE STRUCT
void select_table(DATABASE** selected_db,TABLE** selected_table){
    int index;
    system("CLS");
    display_tablelist(selected_db);//TABLELIST OF SELECTED DATABASE SHOWN USER BEFORE
TABLE SELECTION
    printf("Please enter index number to select table : ");
    scanf("%d",&index);

    while(index<=0 || index>(*selected_db)[0].table_num){//INDEX VALUE CONTROL
        printf("\nPlease enter right index value : ");
        scanf("%d",&index);
    }
//IF TABLE SELECTED SUCCESFULLY NAME OF IT COPYING IN SELETECTED TABLE'S TABLE NAME
ARGUMENT FROM TABLELIST
    strcpy((*selected_table)[0].table_name,(*selected_db)[0].table_list[index-1]);
    printf("TABLE %s selected...\n",(*selected_table)[0].table_name);
}

//THIS FUNCTION READS SELECTED TABLE NAME FROM MEMORY. THEN IT READS AND ASSIGNS TABLE
TO MEMORY IN TABLE ARGUMENT OF TABLE TYPED SELECETED TABLE STRUCT
void load_table(TABLE** selected,DATABASE** selectedb){
    char* table_fname=(char*)calloc(NAME_LENGTH,sizeof(char));
    char* line=(char*)calloc(LINE_LENGTH,sizeof(char));
    int i,j;
    int* table_types=(int*)calloc(1,sizeof(int));
    int* p_keys=(int*)calloc(1,sizeof(int));
    FILE *tablefp;

    //allocation controls
    if(!table_fname && !line && !table_types && !p_keys){
        printf("ERROR OCCURED.... Returning menu....");
        return;
    }

    //table name is formatted to ready for opening
    strcpy(table_fname,(*selected)[0].table_name);
```

```
    strcat(table_fname,"_");
    strcat(table_fname,(*selectedb)[0].dbname);
    strcat(table_fname,"_table.txt");

    //opening table file
    tablefp=fopen(table_fname,"r");

    //taking column and row number from file to global table struct
    fscanf(tablefp,"%d",&(*selected)[0].column);
    fscanf(tablefp,"%d",&(*selected)[0].row);

    //reallocating primary key and type characteristics array
    table_types=(int*)realloc(table_types,sizeof(int)*(*selected)[0].column);
    p_keys=(int*)realloc(p_keys,sizeof(int)*(*selected)[0].column);

//  printf("\nROW : %d COLUMN : %d ",(*selected)[0].row,(*selected)[0].column);

//  TYPE CHARACTERISTICS OF TABLE COLUMNS READING FROM FILE TO TABLE_TYPES ARRAY (1-
INT, 2-DOUBLE, 3-STRING, 4-BOOL)
    for(i=0;i<(*selected)[0].column;i++){
        fscanf(tablefp,"%d",&table_types[i]);
    //  printf(" %d ",table_types[i]); //print to control
    }
//PRIMARY KEY EXISTENCE OF TABLE COLUMNS READING FROM FILE TO P_KEYS ARRAY (0-NO, 1-
YES)
    for(i=0;i<(*selected)[0].column;i++){
        fscanf(tablefp,"%d",&p_keys[i]);
    //  printf(" %d ",p_keys[i]);
    }
    system("PAUSE");



    fgets(line,LINE_LENGTH,tablefp);
//  system("CLS");

    //creating table on memory for table processing
    char*** table=(char***)calloc((*selected)[0].row,sizeof(char**));
    for(i=0;i<(*selected)[0].row;i++){
        table[i]=(char**)calloc((*selected)[0].column,sizeof(char*));
        for(j=0;j<(*selected)[0].column;j++){
            table[i][j]=(char*)calloc(NAME_LENGTH,sizeof(char));
          }
    }

    //displaying contents and assigning table
    //printf("\nDATABASE :%s\tTABLE :%s
\nCONTENTS\n\n",(*selectedb)[0].dbname,(*selected)[0].table_name);

//  HEADLINE MEANS COLUMN NAMES LINE. THEN IT'S FETTING FROM FILE TO
    fgets(line,LINE_LENGTH,tablefp); //printing columnn names on screen in good format
    //printf("\t%s",line);
    //for(i=0;i<(strlen(line)*2);i++)
        //printf("-");
        //printf("\n");

    //TABLE CONTENTS READING FROM FILE TO  DYNAMIC (char*** typed) TABLE ARRAY
    for(i=0;i<(*selected)[0].row;i++){ //prints table on screen
        //printf("\n%d.\t",i+1);
        for(j=0;j<(*selected)[0].column;j++){
            fscanf(tablefp,"%s",table[i][j]);
        //  printf("%s\t",table[i][j]);
        }
    }
    fclose(tablefp);
    //loaded data adresses passing to global selected table struct
    (*selected)[0].table_types=table_types;
```

```
    (*selected)[0].p_keys=p_keys;
    (*selected)[0].headline=line;
    (*selected)[0].table=table;

}

//TABLE FEATURES AND TABLE CONTENTS PRINTING ON THE SCREEN FROM MEMORY
void display_table(TABLE** selected,DATABASE** selected_db){
    int i,j;
    system("CLS");
    if(!(selected)){
        printf("NO TABLE SELECTED PLEASE SELECT TABLE");
        return;//IF THERE IS NO SELECTED TABLE IT RETURNS MENU
    }
    else{//TABLE FEATURES AND TABLE CONTENTS PRINTING ON THE SCREEN FROM MEMORY

printf("DATABASE:%s\tTABLE:%s",(*selected_db)[0].dbname,(*selected)[0].table_name);
        printf("\nRow :%d Column :%d \n\nCHARACTER
:",(*selected)[0].row,(*selected)[0].column);
        for(i=0;i<(*selected)[0].column;i++){
            printf(" %d ",(*selected)[0].table_types[i]);
        }
        printf("\n1-INTEGER,2-DOUBLE,3-STRING,4-BOOL\n");
        printf("\nPRIMARY KEYS : ");
        for(i=0;i<(*selected)[0].column;i++){
            printf(" %d ",(*selected)[0].p_keys[i]);
        }
        printf("\n1-YES,0-NO\n");
        printf("\n\n");
        printf("\t%s",(*selected)[0].headline);
        for(i=0;i<strlen((*selected)[0].headline);i++)
            printf("-");
        printf("\n");

        for(i=0;i<(*selected)[0].row;i++){
            printf("%d.\t",i+1);
            for(j=0;j<(*selected)[0].column;j++){
                printf("%-20s ",(*selected)[0].table[i][j]);
            }
            printf("\n");
        }
    }

}

//DELETES TABLE FILE FROM MEMORY AND IT'S NAME INTO TABLE LIST.TXT FILE....
void delete_table(DATABASE** selected_db){
    system("color b");
    int index;
    int i;
    FILE *list;
    char *temp_name=(char*)calloc(NAME_LENGTH,sizeof(char));
        if(!temp_name){
            printf("\nerror occurred.Returning menu....\n");
            return;
        }
    if(!display_tablelist(selected_db)){
        printf("\nPLEASE SELECT DIFFERENT DATABASE \n");
        system("PAUSE");
        return;
    }
    printf("ENTER INDEX OF TABLE TO DELETE ('0' TO RETURN MENU): ");
    scanf("%d",&index);
    while(index<=0 || index>(*selected_db)[0].table_num){
        if(index==0)
            return;
        printf("\nPLEASE ENTER RIGHT INDEX VALUE('0' TO RETURN MENU) : ");
```

```
        scanf("%d",&index);
    }
    //TABLE FILE NAME FORMATTED AND REMOVED FROM HARD DRIVE
    strcpy(temp_name,(*selected_db)[0].table_list[index-1]);
    strcat(temp_name,"_");
    strcat(temp_name,(*selected_db)[0].dbname);
    strcat(temp_name,"_table.txt");
    printf("Deleted file : %s\n",temp_name);
    remove(temp_name);  //tablename_databasename_table.txt named table file deleted
from hard drive
    system("PAUSE");

    //PROCESSING(SHIFTING AND REALLOCATING) TABLE LIST IN MEMORY AND CONTENTS WRITING
NEW TABLE
    for(i=index-1;i<((*selected_db)[0].table_num);i++){
        (*selected_db)[0].table_list[i]=(*selected_db)[0].table_list[i+1];
    }
    (*selected_db)[0].table_num--;

(*selected_db)[0].table_list=(char**)realloc((*selected_db)[0].table_list,sizeof(char*)
*(*selected_db)[0].table_num);
        for(i=0;i<(*selected_db)[0].table_num;i++)

(*selected_db)[0].table_list[i]=(char*)realloc((*selected_db)[0].table_list[i],sizeof(c
har)*NAME_LENGTH);

    //writing new list to tablelist file compatible with the determined format of
filename
    strcpy(temp_name,(*selected_db)[0].dbname);
    strcat(temp_name,"_tablelist.txt");
    printf("file name : %s \n",temp_name);
    list=fopen(temp_name,"w+");
    if(!list){
        printf("\nerror occurred.Returning menu....\n");
        return;
        system("PAUSE");
    }
    fprintf(list,"%d \n",(*selected_db)[0].table_num);//writing table number
    for(i=0;i<(*selected_db)[0].table_num;i++){
        fprintf(list,"%s \n",(*selected_db)[0].table_list[i]);//writing table names
        //system("PAUSE");
    }
    fclose(list);

    system("color a");
    system("PAUSE");
}

//INSERTS ROW ON GIVEN INDEX OF SELECTED TABLE OF SELECTED DATABASE
void insert_row(TABLE** selected_table,DATABASE** selected_db){
    int index,i,j,k,control;
    //THIS STEP FOR CODE READABILITY... NOT NECESSARY, BUT GOOD FOR READABILITY
    int row=(*selected_table)[0].row;
    int column=(*selected_table)[0].column;
    int *table_types=(*selected_table)[0].table_types;
    int *p_keys=(*selected_table)[0].p_keys;
    char* headline=(*selected_table)[0].headline;

    //index control
    printf("\nPLEASE ENTER THE ROW INDEX TO INSERT ROW IN A TABLE : %s\nROW INDEX(ENTER
0 TO RETURN): ",(*selected_table)[0].table_name);
    scanf("%d",&index);
    while(index<=0 || index>row+1){
        if(index==0)
            return;
        printf("\nPLEASE ENTER RIGHT INDEX VALUE TO INSERT ROW (ENTER 0 TO RETURN): ");
        scanf("%d",&index);
```

```
    }

    //table reallocating on memory to add new row.it is necessary for primary key
control for new rows
row++;
//I TRIED REALLOCATION BUT FAILED.SO I CREATED NEW TABLE AND FREE OLD ONE. ASK QUESTION
TO LECTURER ABOUT THIS PROBLEM...
char*** table=(char***)calloc(row,sizeof(char**));
    for(i=0;i<row;i++){
        table[i]=(char**)calloc(column,sizeof(char*));
        for(j=0;j<column;j++){
            table[i][j]=(char*)calloc(NAME_LENGTH,sizeof(char));
          }
    }
    for(i=0;i<row-1;i++){
        for(j=0;j<column;j++){
            strcpy(table[i][j],(*selected_table)[0].table[i][j]);
        }
    }
    free((*selected_table)[0].table);   //FREEING OLD ARRAY IN STRUCT
    //CONTENTS COPIED NEW TABLE THAT HAS EMPTY ONE ROW AT LAST.


    i=row-1;//REAL ROW INDEX OF TABLE ASSIGNED I VARIABLE
    k=0;//CONTROL VARIABLE FOR PRIMARY KEY,DIGIT OR ALPHABETIC CHARACTERS CORRECTNESS
    printf("\nHEADLINE IS :\n%s",headline);

            for(j=0;j<column;j++){
            k=0;
            control=0;
        //  printf("\nrow number :%d column number :%d \nvalue : \n",i+1,j);

            switch (table_types[j]){

                case 1://INTEGER CASE
                    printf("TYPE: INTEGER\n");
                    scanf("%s",table[i][j]);
                    //PRIMARY KEY AND ISDIGIT CONTROL FOR INTEGER TYPED DATA.
                    do{
                        while(k<strlen(table[i][j])){//digit control for integer
                            if(!isdigit(table[i][j][k++])){
                                printf("\njust digits allowed please enter again : ");
                                scanf("%s",table[i][j]);
                                k=0;
                            }
                        }
                        control=1;//IF ALL CHARACTERS DIGIT CONTROL IS 1
                        if(p_keys[j]){//IF THE COLUMN HAS PRIMARY KEY('0' MEANS NO, '1'
MEANS YES)
                            k=0;
                            while(k<i){
                                if(strcmp(table[k++][j],table[i][j])==0){
                                    control=0;//IF GIVEN DATA ENTERED BEFORE CONTROL IS
                                    printf("\nInput exists.Column has primary key.Enter
new input : ");
                                    scanf("%s",table[i][j]);
                                    k=0;
                                }
                            }
                        }
                        k=0;
                    }while(control==0);
                //  fprintf(table_file,"%s\t ",table[i][j]);
                    break;

                case 2://DOUBLE CASE
                    printf("TYPE: DOUBLE\n");
```

```
                    scanf("%s",table[i][j]);
                    do{
                        while(k<strlen(table[i][j])){//digit control for double
                            if(isalpha(table[i][j][k++])){
                                printf("\njust digits allowed please enter again : ");
                                scanf("%s",table[i][j]);
                                k=0;
                            }
                        }
                        control=1;
                        if(p_keys[j]){
                            k=0;
                            while(k<i){
                                if(strcmp(table[k++][j],table[i][j])==0){
                                    control=0;
                                    printf("\nInput exists.Column has primary key.Enter
new input : ");
                                    scanf("%s",table[i][j]);
                                    k=0;
                                }
                            }
                        }
                        k=0;
                    }while(control==0);
                //  fprintf(table_file,"%s\t ",table[i][j]);
                    break;

                case 3://STRING CASE
                    printf("TYPE: STRING\n");
                    scanf("%s",table[i][j]);
                        if(p_keys[j]){
                            while(k<i){
                                if(strcmp(table[k++][j],table[i][j])==0){
                                    printf("\nInput exists.Column has primary key.Enter
new input : ");
                                    scanf("%s",table[i][j]);
                                    k=0;
                                }
                            }
                        }
                    k=0;
                //  fprintf(table_file,"%s\t ",table[i][j]);
                    break;

                case 4://BOOLEAN(ONE CHARACTER CASE)
                    printf("TYPE: BOOL(ONE CHARACTER)\n");
                    table[i][j][0]=getche();
                    while(table[i][j][0]!='T' && table[i][j][0]!='F'){
                        printf("\nPlease enter value 'T' OR 'F' ");
                        table[i][j][0]=getche();
                    }
                //  fprintf(table_file,"%c\t",table[i][j][0]);
                    break;
            }
        }
    //INSERTING ROW STEP
    char** temp_row=table[row-1];
    //SHIFTING ALL ROWS FROM GIVEN INDEXED ROW
    for(i=row-1;i>index-1;i--)
        table[i]=table[i-1];
    table[index-1]=temp_row;

        //CHANGES APPLYING TO STRUCT...
    (*selected_table)[0].table=table;
    (*selected_table)[0].row=row;
    (*selected_table)[0].column=column;
    (*selected_table)[0].table_types=table_types;
```

```
    (*selected_table)[0].p_keys=p_keys;
    (*selected_table)[0].headline=headline;
    //NEW TABLE SAVING FILE
    save_table(selected_table,selected_db);
    display_table(selected_table,selected_db);
    printf("\nGIVEN ROW INSERTED SUCCESFULLY TO GIVEN INDEX...\n");
}

//USED IN UPDATE AND DELETE ROWS. SELECTS GIVEN ROW INDEXES OF TABLE FROM USER...
RETURNS SELECTED INDEX ARRAY
int* select_rows(int max_row,int* row_count){
    int* selected_rows;
    int i,j=0;
    while((*row_count)<0 || (*row_count)>max_row){
        printf("\nPLEASE ENTER CORRECT VALUE FOR ROW NUMBER ");
        scanf("%d",&(*row_count));
    }
    selected_rows=(int*)calloc((*row_count),sizeof(int));
    //selected index control
    for(i=0;i<(*row_count);i++){
        printf("ENTER %dth row index: ",i+1);
        scanf("%d",&selected_rows[i]);
        while(j<i){
            if(selected_rows[j++]==selected_rows[i]){
                j=0;
                printf("\nROW SELECTED BEFORE ENTER DIFFERENT ROW INDEX : ");
                scanf("%d",&selected_rows[i]);
            }
        }
        j=0;
    }
    printf("\nROWS SELECTED SUCCESFULLY...\n");
    system("PAUSE");
    system("CLS");
    return selected_rows;
}

//UPDATE ROWS ACCORDING TO WHERE FUNCTIONALITY OR SELECTED INDEXES
void update_rows(TABLE** selected_table,DATABASE** selected_db){
    int row_count,i,j,k,control; //CONTROL AND INDEX VARIABLES
    load_table(selected_table,selected_db);
    char* temp=(char*)malloc(sizeof(char)*NAME_LENGTH);//TEMPORARY VARIABLE USED TO
UPDATE EACH COLUMN ELEMENTS OF ROW
    int* row_list; //ROW INDEX LIST THAT WILL BE UPDATE
    int choice;
    //THIS STEP FOR CODE READABILITY... NOT NECESSARY, BUT GOOD FOR READABILITY
    int row=(*selected_table)[0].row;
    int column=(*selected_table)[0].column;
    int *table_types=(*selected_table)[0].table_types;
    int *p_keys=(*selected_table)[0].p_keys;
    char* headline=(*selected_table)[0].headline;
    char*** table=(*selected_table)[0].table;


    printf("\nSELECT METHOD FOR UPDATE(ENTER 0 TO RETURN) :\n1.WHERE FUNCTION\n2.MANUEL
SELECTION\n");
    scanf("%d",&choice);
    while(choice<0 || choice>2){
        if(choice==0)
            return;
        printf("\nSELECT METHOD FOR UPDATE(ENTER 0 TO RETURN) :\n1.WHERE
FUNCTION\n2.MANUEL SELECTION\n");
        scanf("%d",&choice);
    }
    if(choice==2){
        printf("\nHOW MANY ROWS WILL SELECT : ");
        scanf("%d",&row_count);
```

```
        row_list=select_rows((*selected_table)[0].row,&row_count);
    }
    if(choice==1){
        row_list=where_function(selected_table,&row_count);
    }
    display_table(selected_table,selected_db);
    printf("\nSELECTED ROWS TO UPDATE : ");
    for(i=0;i<row_count;i++)
        printf(" %d ",row_list[i]);
    printf("\n");
    //GETTING INPUTS APPLYING TYPE AND PRIMARY KEY CONTROL
    for(i=0;i<row_count;i++){
        for(j=0;j<column;j++){
            k=0;
            control=0;
            printf("\nROW: %d COLUMN: %d\n",row_list[i],j+1);
            switch(table_types[j]){
                case 1://INTEGER DATA INPUT
                printf("TYPE: INTEGER\n");
                scanf("%s",temp);
                    do{
                    while(k<strlen(temp)){//digit control for integer
                        if(!isdigit(temp[k++])){
                            printf("\njust digits allowed please enter again : ");
                            scanf("%s",temp);
                            k=0;
                        }
                    }
                    control=1;
                    if(p_keys[j]){
                            k=0;
                            while(k<row){
                                if(k==(row_list[i]-1)){
                                    k++;
                                }
                                else{
                                    if(strcmp(table[k++][j],temp)==0){
                                        control=0;
                                        printf("\nInput exists.Column has primary
key.Enter new input : ");
                                        scanf("%s",temp);
                                        k=0;
                                    }
                                }
                            }
                        }
                        k=0;
                    }while(control==0);
                    strcpy(table[(row_list[i]-1)][j],temp);
                break;//END OF INTEGER DATA INPUT

                case 2://DOUBLE DATA INPUT
                    printf("TYPE: DOUBLE\n");
                    scanf("%s",temp);
                    do{
                    while(k<strlen(temp)){//digit control for integer
                        if(isalpha(temp[k++])){
                            printf("\njust digits allowed please enter again : ");
                            scanf("%s",temp);
                            k=0;
                        }
                    }
                    control=1;
                    if(p_keys[j]){
                            k=0;
                            while(k<row){
```

```
                                if(k==(row_list[i]-1)){//CONTROL TO ADD SAME ROW SAME
DATA UPDATE
                                    k++;
                                }
                                else{
                                    if(strcmp(table[k++][j],temp)==0){
                                        control=0;
                                        printf("\nInput exists.Column has primary
key.Enter new input : ");
                                        scanf("%s",temp);
                                        k=0;
                                    }
                                }
                            }
                        }
                        k=0;
                    }while(control==0);
                    strcpy(table[(row_list[i]-1)][j],temp);
                break;// END OF DOUBLE DATA INPUT

                case 3://STRING DATA INPUT
                    printf("TYPE: STRING\n");
                    scanf("%s",temp);
                    if(p_keys[j]){
                        while(k<row){
                            if(k==(row_list[i]-1)){
                                k++;
                            }
                            else{
                                if(strcmp(table[k++][j],temp)==0){
                                    printf("\nInput exists.Column has primary key.Enter
new input : ");
                                    scanf("%s",temp);
                                    k=0;
                                }
                            }
                        }
                    }
                k=0;
                strcpy(table[(row_list[i]-1)][j],temp);
                    break;//END OF STRING DATA INPUT

                case 4://BOOL(ONE CHARACTER) DATA INPUT
                    printf("TYPE: BOOL(ONE CHARACTER)\n");
                    strcpy(temp," ");
                    temp[0]=getche();
                    while(temp[0]!='T' && temp[0]!='F'){
                        printf("\nPlease enter value 'T' OR 'F' ");
                        temp[0]=getche();
                    }
                    strcpy(table[(row_list[i]-1)][j],temp);
                    break;  //END OF BOOL DATA INPUT
            }
        }
        system("CLS");
        display_table(selected_table,selected_db);
    }

    //NEW TABLE SAVING FILE WITH FUNCTION
    save_table(selected_table,selected_db);
    //CHANGES APPLYING...
    (*selected_table)[0].table=table;
    (*selected_table)[0].row=row;
    (*selected_table)[0].column=column;
    (*selected_table)[0].table_types=table_types;
    (*selected_table)[0].p_keys=p_keys;
    (*selected_table)[0].headline=headline;
```

```
    display_table(selected_table,selected_db);
    printf("\nROW(S) UPDATED SUCCESFULLY...\n");
}

//SAVES TABLE FROM MEMORY TO FILE
void save_table(TABLE** selected_table,DATABASE** selected_db){
    int i,j;
    //THIS STEP FOR CODE READABILITY... NOT NECESSARY, BUT GOOD FOR READABILITY
    int row=(*selected_table)[0].row;
    int column=(*selected_table)[0].column;
    int *table_types=(*selected_table)[0].table_types;
    int *p_keys=(*selected_table)[0].p_keys;
    char* headline=(*selected_table)[0].headline;
    char*** table=(*selected_table)[0].table;

        //filename process...
    char* filename=(char*)calloc(NAME_LENGTH,sizeof(char));
    strcpy(filename,(*selected_table)[0].table_name);
    strcat(filename,"_");
    strcat(filename,(*selected_db)[0].dbname);
    strcat(filename,"_table.txt");

        //opening table folder as empty
        FILE *file_w;
            file_w=fopen(filename,"w+");
            if(!file_w){
                printf("ERROr occured retuning menu...");
                system("PAUSE");
            }
        //writing new table to file
        fprintf(file_w,"%d\t ",column);
        fprintf(file_w,"%d\t \n",row);
        for(i=0;i<column;i++)
            fprintf(file_w,"%d ",table_types[i]);
            putc('\n',file_w);
        for(i=0;i<column;i++)
            fprintf(file_w,"%d ",p_keys[i]);
            putc('\n',file_w);
            fputs(headline,file_w);
        for(i=0;i<row;i++){
            for(j=0;j<column;j++){
                fprintf(file_w,"%-20s ",table[i][j]);
            }
            putc('\n',file_w);
        }

        fclose(file_w);
}

//WHERE FUNCTION TO FIND KEYWORD. IT CHANGES ROW_COUNT AND RETURNS INDEX ARRAY OF
FOUNDED KEYWORDS
int* where_function(TABLE** selected_table,int *row_count){
    int i,j,k;
    //THIS STEP FOR CODE READABILITY... NOT NECESSARY, BUT GOOD FOR READABILITY
    int row=(*selected_table)[0].row;
    int column=(*selected_table)[0].column;
    int *table_types=(*selected_table)[0].table_types;
    int *p_keys=(*selected_table)[0].p_keys;
    char* headline=(*selected_table)[0].headline;
    char*** table=(*selected_table)[0].table;
    char* keyword=(char*)calloc(NAME_LENGTH,sizeof(char));
    int selected_column;
    int *selected_rows=(int*)calloc(1,sizeof(int));
        printf("\nENTER COLUMN INDEX TO SEARCH KEYWORD: ");
        scanf("%d",&selected_column);
        while(selected_column<0 || selected_column>column){
            printf("\nENTER CORRECT COLUMN NUMBER TO SEARCH KEYWORD: ");
```

```
            scanf("%d",&selected_column);
        }
    do{
        printf("\nPLEASE ENTER KEYWORD FOR SEARCH & ROW SELECTION :");
        scanf("%s",keyword);
        k=0;
            for(i=0;i<row;i++){
                if(strcmp(table[i][selected_column-1],keyword)==0){
                    selected_rows=(int*)realloc(selected_rows,sizeof(int)*(k+1));
                    selected_rows[k++]=i+1;
                }
            }
        }while(k==0);

    (*row_count)=k;
    return selected_rows;

}

//DELETE ROWS FROM SELECTED TABLE
void delete_rows(TABLE** selected_table,DATABASE** selected_db){
    int row_count,i,j;
    char* temp=(char*)malloc(sizeof(char)*NAME_LENGTH);
    //THIS STEP FOR CODE READABILITY... NOT NECESSARY, BUT GOOD FOR READABILITY
    int row=(*selected_table)[0].row;
    int column=(*selected_table)[0].column;
    int *table_types=(*selected_table)[0].table_types;
    int *p_keys=(*selected_table)[0].p_keys;
    char* headline=(*selected_table)[0].headline;
    char*** table=(*selected_table)[0].table;
    int* row_list;
    int choice;
    printf("\nSELECT METHOD FOR UPDATE :\n1.WHERE FUNCTION\n2.MANUEL SELECTION\n");
    scanf("%d",&choice);
    while(choice<0 || choice>2){
        printf("\nSELECT METHOD FOR UPDATE :\n1.WHERE FUNCTION\n2.MANUEL SELECTION\n");
        scanf("%d",&choice);
    }
    if(choice==2){
        printf("\nHOW MANY ROWS WILL SELECT : ");
        scanf("%d",&row_count);
        row_list=select_rows((*selected_table)[0].row,&row_count);
        Sel_sort(row_list,row_count);
    }
    if(choice==1){
        row_list=where_function(selected_table,&row_count);
    }
    display_table(selected_table,selected_db);
    printf("\nSELECTED ROWS TO DELETE : ");
    for(i=0;i<row_count;i++)
        printf(" %d ",row_list[i]);
    printf("\n");

    for(i=0;i<row_count;i++){
        for(j=row_list[row_count-1-i];j<row;j++){
            table[j-1]=table[j];
        }
    }
    row-=row_count;
    table=(char***)realloc(table,sizeof(char**)*row);

    //CHANGES APPLYING...
    (*selected_table)[0].table=table;
    (*selected_table)[0].row=row;
    (*selected_table)[0].column=column;
    (*selected_table)[0].table_types=table_types;
    (*selected_table)[0].p_keys=p_keys;
```

```
    (*selected_table)[0].headline=headline;
    //NEW TABLE SAVING FILE WITH FUNCTION
    save_table(selected_table,selected_db);
    system("PAUSE");
    display_table(selected_table,selected_db);
    printf("\nROW(S) DELETED SUCCESFULLY...\n");
}
//SELECTION SORT FOR MANUAL ROW DELETE CHOICE'S CORRECTNESS
void Sel_sort(int *arr,int size){
    int i,j,temp,Mindex;
        for(i=0;i<size;i++){
            Mindex=i;
            for(j=i+1;j<size;j++){
                if(arr[Mindex]>arr[j])
                    Mindex=j;
                }
            temp=arr[Mindex];
            arr[Mindex]=arr[i];
            arr[i]=temp;
        }
}
```