

**TRIBHUVAN UNIVERSITY  
 INSTITUTE OF ENGINEERING PULCHOWK CAMPUS  
DEPARTMENT OF   
ELECTRONICS AND COMPUTER ENGINEERING**

PROJECT ON  
BANK MANAGEMENT SYSTEM   
(Computer Programming- CT401)

Submitted by:

Sandesh Thapa 076BEI036  
Sanim Kumar Khatri 076BEI037  
Suvash Joshi 076BEI045

1styear/Istpart  
Year 2076 B.S./2020A.D.  
Submitted date:2076/11/15(27th February 2020)

# **ABSTRACT**

The bank management system is a program for maintaining a person's account in a bank. In this project we tried to show the working of a banking account system and cover the basic functionality of a bank account management system. The main aim of this project is to develop software for bank management system. This project has been developeed to carry out the processes easily and quickly, which is not possible with the manuals systems, which are overcome by this software. This project si developed using C programming language and text files as flat-database. The subject of the project i.e. 'Bank Management' is treated as a sensational matter. The bank is responsible for the protection of the savings and earning of the clients/people. It would be very hard to keep track of all the transactions and clients' data if it is done manually or written into record books by human themselves. So it is necessary for banks to have a system to store those data in computers so that the data of the clients and transactions can be safe as well as easily assessable and editable. But only storing the data into computers/memory will not make the system effective, as it would to tedious to search for the data in the files and edit it, so there must be an interface between the user and stored data to search, edit and view the stored data. The required interface is simply the project we made as a team: 'Bank Management System'. Creating and managing requirements is a challenge of IT, systems and product development projects or indeed for any activity where you have to manage a contractual relationship. Organization need to effectively define and manage requirements to ensure they are meeting needs of the customer, while providing compliance and staying on the schedule and within budget. The project analyzes the system requirements and then comes up with the requirements specifications. It studies other related systems and then come up with system specifications. The system is then designed in accordance with specifications to satisfy the requirement.

# **ACKNOWLEDGEMENT**

This opportunity we had to make this C project was a great chance for learning and professional development. We are sure the experience from this project will help us in our further lives to deal with project handlings and project development. Therefore we consider ourselves lucky as a team to get a chance to do this project.

Bearing in mind first and foremost we would like to express our deepest gratitude and special thanks to our C programming teacher Anku Jaiswal ma’am who taught us C programming language. Due to her efforts in teaching we learnt many ideas and finally implemented them to make this project successful.

We would like to express deepest thanks to our Lab teacher Biswas Pokharel sir whose guidance in this project helped us to complete this project. Without his necessary guidance in lab we were very less likely to complete this project on time.

We perceive as this opportunity as a big milestone in our career development. We will strive to use gained knowledge in the best possible way, and we will continue to work on their improvements.

**TABLE OF CONTENTS**

[**ABSTRACT** 2](#_Toc33682735)

[**ACKNOWLEDGEMENT** 3](#_Toc33682736)

[**LIST OF TABLES** 5](#_Toc33682737)

[**LIST OF FIGURES** 5](#_Toc33682738)

[**INTRODUCTION** 6](#_Toc33682739)

[**REVIEW OF RELATED LITERATURES** 7](#_Toc33682740)

[**PROBLEM ANALYSIS AND SOLVING** 10](#_Toc33682741)

[**FUNCTIONS AND DATA STRUCTURES** 10](#_Toc33682742)

[**ALGORITHM** 12](#_Toc33682743)

[**CODES** 21](#_Toc33682744)

[**TESTING AND DEBUGGING** 55](#_Toc33682745)

[**DETAILS ABOUT THE MANAGEMENT SYSTEM** 57](#_Toc33682746)

[**DISCUSSION AND CONCLUSION** 62](#_Toc33682747)

# **LIST OF TABLES**

* **Functions………………………………………………………………………………..10**
* **Structures……………………………………………………………………………….11**
* **Library files…………………………….……………………………………………….12**
* **Files…………………………………………………………………………………….12**
* **Testing………………………………………………………………...………...……55**
* **Acc\_type………………………………………………………………………..………57**

# **LIST OF FIGURES**

* **Flowchart of main() function…………………………………………………………13**
* **Figure 1…………………………………………………………………………………56**
* **Figure 2…………………………………………………………………………………56**
* **Figure 3…………………………………………………………………………………58**
* **Figure 4………………………………………………………………………………....58**
* **Figure 5………………………………………………………………………………....59**
* **Figure 6…………………………………………………………………………………59**
* **Figure 7…………………………………………………………………………………60**
* **Figure 8…………………………………………………………………………………60**
* **Figure 9…………………………………………………………………………………61**
* **Figure 10……………………………………………………………………………..…61**

# **INTRODUCTION**

The project entitled 'Bank Management System' is a program written in C programming language for bank management which can handle accounts for customers. It uses files to handle the daily transactions, account management and client management. It is not a complete accounting software just like implemented in the banks but still it can manage the accounts of the customers using files at the back end. Though this particular project does not have broad scope in the near future but the concept involved in the problem solving and analysis can be very useful and has a huge scope in the immediate future. Being an engineering student it is necessary to be good at technical stuffs and this very project marks the beginning of the big journey.

The subject of the project i.e. 'Bank Management' is treated as a sensational matter. The bank is responsible for the protection of the savings and earning of the clients/people. It would be very hard to keep track of all the transactions and clients' data if it is done manually or written into record books by human themselves. So it is necessary for banks to have a system to store those data in computers so that the data of the clients and transactions can be safe as well as easily assessable and editable. But only storing the data into computers/memory will not make the system effective, as it would to tedious to search for the data in the files and edit it, so there must be an interface between the user and stored data to search, edit and view the stored data. The required interface is simply the project we made as a team: 'Bank Management System'. As mentioned above this project is written in C programming language, the more information about this programming language is given in the "Review of related literatures" section.

In this project we covered approximately every areas of services that a bank would provide. This program can be used to register a new client as an account holder, this data would be stored into a flat-database file, which can be used to view the clients' detail, to edit the details of a client when necessary, delete the account of client when they do not want the services of the bank anymore and to carry out daily transactions like withdrawal and deposit. Also another flat-database file is made to keep records of the loan taken by the people/clients. The system is protected by a login system i.e. only authorized personnel can only use this system to use the above mentioned features, this makes the system secured. This project is created without the use of graphics making its interface less attractive.

# **REVIEW OF RELATED LITERATURES**

This project is based on high level language i.e. c programming. In this project we use important parts of c programming which are control statement, looping, function, structure, string and data file.  
**C programming language:** C is structured programming based computer programming language was developed by Dennis Ritchie at Bell laboratories in 1972.Structured programming refers to programming that produce program with clean flow, clear and a degree of modularity or hierarchical structure is a simple, contained, versatile, excellent, efficient, fast general purpose language. It has high degree of language of C is a function oriented additional task including input and output, graphics, math computation and access to peripheral devices are placed as library function.

**Control Statement:** Logical operation is carried out by several symmetrical or logical statements. There are two types of control statement based on their function.   
**Selective structure:** Selective structures are used when we have a number of situations where we need to change the order of execution of statements based on certain condition. The selective statements make a decision to take the right path before changing the order of execution. C provides the following statements for selective structure:

1. If statements
2. Switch statements

**If statements:** The if statement is a powerful decision making statement and it is used to control the flow of execution of statements. It is a two way statement and is used in conjunction with an expression.  
If statement allows the computer to evaluate the expression first and then on depending whether the value of the expression is true or false it transfer the control to the particular statement. At this point of the program has two paths to follow: one for true condition and other for false condition. The types of if statements are explained below:   
Simple if statement: The simple if statement is used to conditionally excite a block of code based on whether a test condition is true or false. If the condition is true the block of code is executed, otherwise it is skipped. The syntax of if statement is given below:   
if(test expression)   
 {  
 statement-block;   
 } statement-x;  
**If else statement:** The if else statement extends the idea of the if statement by specifying another section of code that should be executed only if the condition is false i.e. conditional branching. True- block statements are to be executed only if the test expression is true and false block statements to be executed only if the condition is false. The syntax of if else statement is given below:

if(test expression)   
 {  
 true block statement;

}

else   
 {   
 false block statement;   
 }

**Looping:** Loop caused a section of code to be repeated for a specified number of times or until some condition holds true. When a condition becomes false, the loop terminates and control passes to statement below loop. Different types of loops are discussed below with their major characteristics and syntax used in C:

**While loop:** The “while loop” specifies that a section of code should be executed while a certain condition holds true. The syntax of while loop is given below:   
while(test expression)   
 {   
 body of loop ( statements block)   
 }

**For loop:** The for loop is used to execute a block of code for a fixed number of repetitions. Initialization is generally an assignment statement used to set loop control variable. Test expression is a relational expression that determines when loop exits. Update expression defines how the loop variable changes each time when the loop is repeated. The syntax of for loop is given below:   
for(initialization-expression; test-expression; update-expression)   
 {   
 body of loop;   
 }

**Break statement:** The break statement is used to jump out of loop. The break statement terminates the execution of the nearest enclosing loop. Control passes to the statement that follows the terminated statement. in a switch break statement causes the program to execute the next statement after switch.  
 break;

**Function:** A function is a self-contained program segment that carries out some specific well defined task. Every c program consists of one or functions. Execution of program always begins by carrying out instruction in main. Function makes program significantly easier to understand and maintain. A well written function may be reused in multiple programs. Program that are easier to design, debug and maintain.

**Switch statement:** The function of switch is similar to that of else if ladder but in switch, a test expression is evaluated first and its value is matched against a series of case constants. The only one case whose case constant matches with the value of test expression will get executed. However, if the value of test expression do not match with any of the case constants, default case is executed. The syntax of return statement is given below:   
Switch(expression)   
 {   
 case constant1:   
 block of Case constant1;   
 break:   
 case constan2:   
 block of Case constant2;   
 break:   
 case constant3:   
 block of Case constant3;   
 break:   
 --------------   
 --------------   
 default:   
 default block;   
 }

**File:** Many applications require that information be written to or read from an auxiliary memory device. Such information is stored on the memory device in the form of a data file. The data files allow us to store information permanently and to access and alter that information whenever necessary.  
Opening a file: Before performing any input/output operation, file must be opened. While opening file, the following must be specified:

1. The name of file
2. The manner in which it should be opened (that for reading ,writing ,both reading and writing ,appending at the end of file, overwriting the file)   
   Syntax for opening a file:   
   ptr\_variable=fopen(file\_name,file\_mode);   
   This associates a file name with the buffer and specifies how the data file will be utilized(File Opening Mode).The function fopen() returns a pointer to the beginning of the buffer area associated with the file.
3. When working with a stream oriented data file ,the first step is to establish a buffer area, where information is temporary stored while being transferred between the computer’s memory and data file .the buffer area is established by writing:  
    FILE \*ptr\_variable;   
   Here, FILE is a special structure declared in header file stdio.h.

# **PROBLEM ANALYSIS AND SOLVING**

Here are the functions, structures and header files/library files used in the program as a part of problem analysis also to be mentioned test and i were declared globally as integers so that every functions could use these variables.

## **FUNCTIONS AND DATA STRUCTURES**

1. Functions

|  |  |  |  |
| --- | --- | --- | --- |
| Function name | Data type | Arguments | Return/Remarks |
| simple\_interest | float | float amt, float time, float rate | Returns interest,  used to calculate the monthly simple interest of the saving accounts. |
| CA\_forF | float | float amt, float time, float rate | Returns CA,  used to calculate total compound amount for the fixed accounts. |
| fdelay | void | int a | Used to delay the process in order to freeze the screen for some time(by input int a) |
| main | void |  | Main function with the login page |
| main\_menu | void | - | Function created to list all the functions available and to choose one action after successful login |
| new\_account | void | - | Function created to create new account of the client in the bank system |
| update | void | - | Function created to update the updateable information the account holders. |
| check | void | - | Function created to view detail of a particular account holder. |
| delete\_acc | void | - | Function created to delete the bank account of the account holder |
| view\_list | void | - | Function created to view the list of all account holders of the bank and all the loan granted. |
| out | void | - | Function created to exit the system. |
| loan | void | - | Function created to grant loan to the client, check detail of the loan granted and to clear loan after the successful clearance of the loan. |
| transact | void | - | Function created to withdraw and deposit the amount from and into the bank account of the account holders. |

2. Structures

a. struct date

|  |  |
| --- | --- |
| Members | Data types |
| y | int |
| m | int |
| d | int |

b.struct loanee

|  |  |
| --- | --- |
| Members | Data types |
| loan\_id | int |
| first\_name | string i.e char[20] |
| last\_name | string i.e. char [20] |
| dob | struct date |
| loan\_taken | struct date |
| loan\_deposit | string i.e. char[20] |
| address | string i.e. char[20] |
| citizenship | string i.e. char[20] |
| loan\_amt | float |
| phone | double |

c. struct data

|  |  |
| --- | --- |
| Members | Data types |
| amt | float |
| acc\_no | int |
| first\_name | string i.e. char[20] |
| last\_name | string i.e. char[20] |
| address | string i.e. char[20] |
| acc\_type | string i.e. char[3] |
| dob | struct date |
| citizenship | string i.e. char[20] |
| deposit | struct date |
| phone | double |

3. Library files

|  |
| --- |
| Header files |
| stdio.h |
| math.h |
| string.h |
| stdlid.h |
| conio.h |

4. Files used in storing records

|  |  |
| --- | --- |
| File name | Remarks |
| record.txt | Flat-database text file used to store records of the account holders |
| loan.txt | Flat-database text file used to store records of the loan and clients |
| tempfile.txt | Temporary file used in loan function |
| time.txt | Temporary file used in transact file |
| newr.txt | Temporary file used in update function |
| newrec.txt | Temporary file used in delete\_account function |

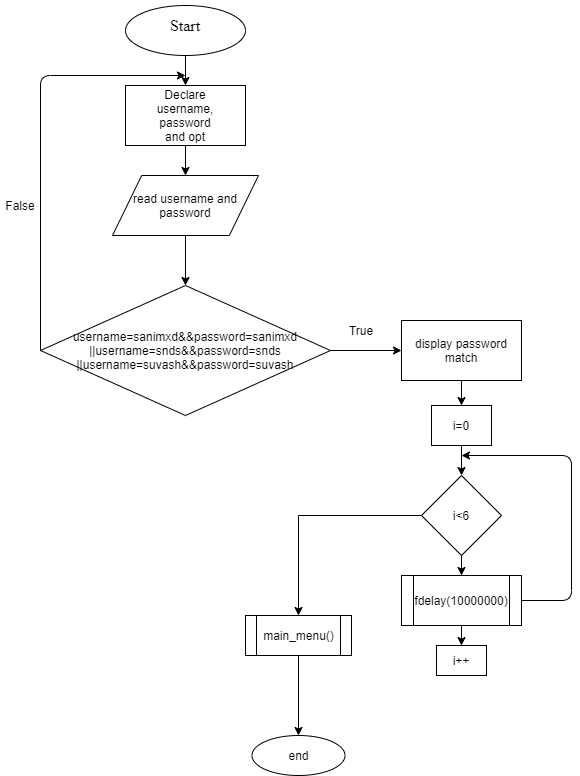
After the knowledge of the used functions and data structures here is the algorithm used in this program:

## **ALGORITHM**

main() funtion

This main function is mainly used as a login page for the bank management system, the login is done by using username and password. In this program we have set the username as "sanimxd", "snds", "suvash" and their password as "sanimxd", "snds", "suvash" respectively.

Flowchart (algorithm in pictorial form) for main() function is given in the next page:



main\_menu() function

In this function the menu is displayed, i.e. the actions are given and one action is to be chosen;. The algorithm for the main\_menu() is given below:

Step 1: Start  
Step 2: declare action  
Step 3: display the menu  
Step 4: read action  
Step 5: if action=1  
 call new\_account()  
 else if action=2  
 call update()  
 else if action=3  
 call check()  
 else if action=4  
 call transact()  
 else if action=5  
 call delete\_account()  
 else if action=6  
 call view\_list()  
 else if action=7  
 call loan()  
 else if action=8  
 call out()  
 else  
 goto step 1  
Step 6: Stop

new\_account() function

This function is created to make new account of the client, it reads the data of the client and stores it into the flat-database. The algorithm for this function is given below:

Step 1: Start  
Step 2: declare file pointer add and integer check\_acc\_no  
Step 3: declare structure data new\_acc, test\_acc  
Step 4: open file "record.txt" as append and read mode in file pointer add  
Step 5: Read check\_acc\_no   
Step 6: If the account number exits in the file goto step 5  
Step 7: Else assign new\_acc.acc\_no as check\_acc\_no  
Step 8: Read new\_acc.name, new\_acc.dob, new\_acc.phone, new\_acc.acc\_type, new\_acc.citizenship, new\_acc.amt, new\_acc.deposit, new\_acc.address  
Step 9:write the structure in file "record.txt"  
Sterp 10: display "account created"  
Step 11: Read test  
Step 12: if test=1  
 call main\_menu()  
 else if test=0  
 call out()  
 else display "invalid output" and goto Step 11  
Step 13:Stop

update() function  
This function is used to update the information of the account holder i.e. the personal information of the account holder. The algorithm of the update() function is given below:

Step 1: Start  
Step 2: Declare j, k, re as integers  
Step 3: Declare real and upd as file pointer  
Step 4: Decalre structure data up  
Step 5: Open record.bin as read mode in file pointer real  
Step 6: Open newr.bin as write mode in upd file pointer  
Step 7: Read j  
Step 8: Read structure up from the file record.bin 1 at a time  
Step 9: if j=up.acc\_no  
 test++  
Step10: Read k  
Step 11: If k=1  
 read up.address and write the structure in the file newr.bin  
Step 12: else if k=2  
 read up.name and write the structure in the file newr.bin  
Step 13: else if k=3  
 read up.citizenship and write the structure in newr.bin  
Step 14: else if k=4  
 read up.phone and write the structure in the file newr.bin  
Step 15: else read re  
Step 16: If re=1  
 call main\_menu()  
Step 17: else if re=0  
 call out()  
Step 18: else if re=2  
 goto step 15  
Step 19: else j

check() function

This function is created to check or view the information of a particular person via his/her account number, name or citizenship number. The algorithm for this function is given below:

Step 1: Start  
Step 2: Declare structure data check\_acc  
Step 3: Declare string check\_ac, check\_accitizen  
Step 4: Declare integer check\_accno, check\_test=0 and float total\_amount  
Step 5: Declare file pointer chec  
Step 6: Open file record.bin in read mode in file pointer chec  
Step 7: Read test  
Step 8: if test=1 read check\_ac  
Step 9: read structure from the file record.bin  
Step 10: if check\_acc.name=check\_ac  
 check\_test++  
 display check\_acc.name, check\_acc.acc.no, check\_acc.address, check\_acc.phone, check\_acc.dob  
Step 11:if check\_test=0  
 read i  
Step 12: If i=2  
 call check()  
Step 13: else if i=1  
Step 14: else if i=0  
 call out()  
Step 15:end if|  
Step 16: else if test=2 read check\_acno  
Step 17: read structure from the file record.bin  
Step 18: if check\_acno=check\_acc.acc\_no  
 check\_test++  
 display check\_acc.name, check\_acc.acc.no, check\_acc.address, check\_acc.phone, check\_acc.dob  
Step 19: if check\_test=0  
 read i  
Step 20: if i=2  
 call check()  
Step 21: else if i=1  
 Call main\_menu()  
Step 22: else if i=0  
 Call out()  
Step 23:end if  
Step 24: else if test=3 read check\_accitizen  
Step 25:read structures from the file record.bin  
Step 26: if check\_acc.citizenship=check\_accitizen  
 check\_test++  
 display check\_acc.name, check\_acc.acc.no, check\_acc.address, check\_acc.phone, check\_acc.dob  
Step 27: if check\_test=0  
 read i  
Step 28: if i=2  
 call check()  
Step 29: else if i=1  
 Call main\_menu()  
Step 30: else if i=0  
 Call out()  
Step 31: else goto step 7  
Step 32: close record.bin file  
Step 33: read test  
Step 34: if test=0  
 call out()  
Step 35: else if test=1  
 call main\_menu()  
Step 36: else goto step 33  
Step 37: Stop

delete\_account() function

This function is created to delete the account of the account holder if s/he wants to discontinue the service of the bank with come protocol fulfilled. The algorithm for this function is given below:

Step 1: Start  
Step 2: Declare integer delac  
Step 3: Declare structure data del\_acc  
Step 4: Declare file pointer del and delt  
Step 5: Open file record.bin in read mode in del file pointer  
Step 6: Open file newrec.bin in write mode in delt file pointer  
Step 7: read delac  
Step 8: read structures from the record.bin file  
Step 9:if delac!=del\_acc.acc\_no  
 write the structure in the newrec.bin file  
Step 10: Close file record.bin and newrec.bin  
Step 11: Open record.bin in write mode and newrec.bin in read mode  
Step 12: Read structures from newrec.bin and write it into the record.bin file  
Step 13:close record.bin and newrec.bin  
Step 14: Remove newrec.bin  
Step 15: read test  
Step 16: if test=1  
 call main\_menu()  
Step 17: else if test=0  
 call out()  
Step 18: else goto step 15  
Step 19:Stop

view\_list() function

In this function the user can view the list of all clients of the bank, all the account holders and debtors. The algorithm of thisa function is given below:

Step 1: Start  
Step 2: declare pc and retry as integer  
Step 3: Declare structure data view and structure loanee viewloan  
Step 4: Declare fview and fviewloan as file pointers  
Step 5: Read pc  
Step 6: if pc!=1234   
 read retry  
Step 7: if retry=1  
 call main|\_menu()  
Step 8: else if retry=0  
 call out()  
Step 9: else call view\_list()  
Step 10: else display password match  
Step 11: i=0  
Step 12: if i<6  
 fdelay(100000000) and display "."  
Step 13: i++  
Step 14: goto step 12  
Step 15: else  
 system("cls")  
Step 16: read structures from record.bin and display view, acc\_no and view.name  
Step 17: read structures from loan.bin and display viewloan.loan\_id and viewloan.name  
Step 18: read i  
Step 19: if i=1  
 Call main\_menu()  
Step 20: else if i=0  
 call out()  
Step 21: else goto step 18  
Step 22: Stop

out() function

It is just a function which clears the screen and terminates the program, it's algorithm is simple and given below:

Step 1: Start  
Step 2: Clear screen  
Step 3: Display "you have exited the system"  
Step 4: Stop

loan() function

It is function which deals with the loan taken by the client, in this function can be granted to the client the detail of loan taken by the client can be viewed and the loan cacn be cleared after successful payment of loan and the loan interest. The algorithm for theis function if given below:

Step 1: Start  
Step 2: Declare structure loanee  
Step 3: Read i  
Step 4: if i=1  
 declare lon file pointer and open loan.bin in append mode  
 read the structure a and write in the file loan.bin

Step 5: end if  
Step 6: else if i=2  
 declare check\_name  
Step 7: Open loan.bin in reada mode  
Step 8: read check\_name  
Step 9: read structure from the loan.bin  
Step 10: if check\_name=a.name  
 test++  
 display structure a  
Step 11: if test=0  
 read i  
Step 12: if i=1 call main\_menu()  
Step 13: else if i=0 call out()  
Step 14: else goto step 8  
Step 15: close loan.bin file  
Step 16: read i  
Step 17: if i=1 call main\_menu()  
Step 18: else if i=0 call out()  
Step 19: else goto 17  
Step 20: end else if  
Step 21: else if i=3  
 declare acc\_d  
Step 22: open loan.bin in read mode and tempfile.bin in write mode  
Step 23: read acc\_d  
Step 24: read structures from loan.bin  
Step 25: if acc\_d!=a.loan\_id  
 write the structure in tempfile.bin  
Step 26: close tempfile.bin and loan.bin  
Step 27: open tempfile.bin in read mode and loan.bin in write mode  
Step 28: Read structures from tempfile.bin and write it to loan.bin  
Step 29: remove tempfile.txt  
Step 30: read i  
Step 31: If i=1  
 call main\_menu()  
Step 32: else   
 call out()  
Step 33: Stop

transact() function

This function is created to perform withdrawal and deposit to the account of the account holder. The algorithm of the function is given below:

Step 1: Start  
Step 2: Declare acc as integer, withdraw\_acc and deposit\_acc as float  
Step 3: declare structure data transact\_acc  
Step 4: declare fptr and temp as file pointer  
Step 5: open record.bin in read mode and tine.bin in write mode  
Step 6: read i  
Step 7: if i=1  
 read acc  
Step 8: read structure from record.bin file  
Step 9: if transact\_acc.acc\_no=acc  
 test++  
Step 10: if transact\_acc.acc\_type=F1||F2||F5  
 goto step 13  
Step 11: else read withdraw\_amt  
Step 12: transact\_acc.amt-=withdraw\_amt  
Step 13: write the structure in time.bin  
Step 14: else goto step 13  
Step 15: close time.bin and record.bin files  
Step 16: open record.bin in write mode and tie.bin in read mode  
Step 17: write all structures of time.bin into record.bin  
Step 18: remove record.bin  
Step 19:else if i=2  
 read acc  
Step 20: read structure from record.bin file  
Step 21: if transact\_acc.acc\_no=acc  
 test++  
Step 22: if transact\_acc.acc\_type=F1||F2||F5  
 goto step 25  
Step 23: else read deposit\_amt  
Step 24: transact\_acc.amt+=deposit\_amt  
Step 25: write the structure in time.bin  
Step 26: else goto step 25  
Step 27: close time.bin and record.bin files  
Step 28: open record.bin in write mode and tie.bin in read mode  
Step 29: write all structures of time.bin into record.bin  
Step 30: remove record.bin  
Step 31: read i  
Step 32: if i=1  
 call main\_menu()  
Step 33: else if i=0  
 Call out()  
Step 34: else  
 goto step 6  
Step 35 Stop

After the algorithm the problem was implemented into codes.

## **CODES**

#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<math.h>

#include<stdlib.h>

int test=0, i;

/\*to calculate interest for saving account\*/

float simple\_interest(float amt, float time, float rate)

{

float interest;

interest=(amt\*time\*rate)/100;

return(interest);

}

/\*to calculate amount for fixed account and loan annually\*/

float CA\_forF(float amt, float time, float rate)

{

float CA;

CA=amt\*pow((1+rate), time);

return(CA);

}

/\*to delay the screen\*/

void fdelay(int a)

{ int i,k;

for(i=0;i<a;i++)

k=i;

}

/\*for main menu of the system\*/

void main\_menu();

/\*to create new account in the bank\*/

void new\_account();

/\*to update the informations of the account holder\*/

void update();

/\*to check the detail of the existing account holder\*/

void check();

/\*for transaction of money\*/

void transact();

/\*to delete the account of the account holder\*/

void delete\_account();

/\*to view the list of the account holders with their account number\*/

void view\_list();

/\*to deal loan\*/

void loan();

/\*to exit the system\*/

void out();

/\*to store date\*/

struct date

{

int y, m, d;

};

/\*for the detail of the loanee\*/

struct loanee

{

int loan\_id;

char first\_name[20], last\_name[20];

struct date dob;

struct date loan\_taken;

char loan\_deposit[30];

char address[20], citizenship[20];

float loan\_amt;

double phone;

};

/\*to store information of the account holder\*/

struct data

{

float amt;

int acc\_no;

char first\_name[20], last\_name[20], address[20], acc\_type[3];

struct date dob;

char citizenship[20];

struct date deposit;

double phone;

};

void main()

{

char username[10], password[20];

int opt;

system("Color F0");

/\*for login into the system\*/

login:

system("cls");

printf("\n\n\n\t\t\t\t\tWELCOME TO BANK MANAGEMENT SYSTEM!!!\n\t\t\t\t\tENTER THE USERNAME AND PASSWORD TO ACCESS THE SYSTEM:");

printf("\n\t\t\t\t\tUSERNAME:");

scanf("%s", username);

printf("\t\t\t\t\tPASSWORD:");

scanf("%s", password);

if((strcmp(username,"sanimxd")==0&&strcmp(password,"sanimxd")==0)

||(strcmp(username,"snds")==0&&strcmp(password,"snds")==0)

||(strcmp(username,"suvash")==0&&strcmp(password,"suvash")==0))

{

printf("\n\nLOGIN SUCCESSFUL!\nLOADING");

for(i=0;i<6;i++)

{

fdelay(100000000);

printf(".");

}

system("cls");

main\_menu();

}

else

{

printf("\t\t\t\t\tWRONG USERNAME OR PASSWORD!");

printf("\n\tDo you want to retry(1) or exit?(Enter 1 to retry and any other number to exit):");

scanf("%d", &opt);

if(opt==1)

{

goto login;

}

else

{

out();

}

}

}

void main\_menu()

{

int action;

system("cls");

printf("\n\n\n\n\n\t\t\t\t\tWELCOME TO THE SYSTEM!!\n\t\t\t\t\tCHOOSE YOUR ACTION:\n\t\t\t\t\t1.Create new account\n\t\t\t\t\t2.Update information of existing account\n\t\t\t\t\t3.Check details of existing account\n\t\t\t\t\t4.Transact\n\t\t\t\t\t5.Remove existing account\n\t\t\t\t\t6.View customer's list\n\t\t\t\t\t7.LOAN\n\t\t\t\t\t8.EXIT\n\t\t\t\t\tENTER YOUR CHOICE:");

scanf("%d", &action);

switch(action)

{

case 1:

new\_account();

break;

case 2:

update();

break;

case 3:

check();

break;

case 4:

transact();

break;

case 5:

delete\_account();

break;

case 6:

view\_list();

break;

case 7:

loan();

break;

case 8:

out();

break;

default:

printf("ERROR ENTER A NUMBER TO CHOOSE THE ACTION!!!");;

for(i=0;i<15;i++)

{

fdelay(100000000);

printf(".");

}

system("cls");

main\_menu();

}

}

void new\_account()

{

int check\_acc\_no;

FILE\*add;

struct data test\_acc, new\_acc;

accno\_same:

add=fopen("record.txt", "a+");

system("cls");

printf("\n\n\n\t\t\t\t\tEnter the account number:");

scanf("%d", &check\_acc\_no);

while(fread(&test\_acc, sizeof(test\_acc), 1, add))

{

if(check\_acc\_no==test\_acc.acc\_no)

{

printf("\n\t\t\t\t\tThe Account number already exists!\n\t\t\t\t\tEnter Another Account number!!");

printf("Loading");

for(i=0;i<6;i++)

{

fdelay(100000000);

printf(".");

}

goto accno\_same;

}

}

new\_acc.acc\_no=check\_acc\_no;

printf("\n\n\nEnter the first name:");

scanf("%s", new\_acc.first\_name);

printf("\nEnter the last name:");

scanf("%s", new\_acc.last\_name);

printf("\nDATE OF BIRTH(yyyy/mm/dd):");

scanf("%d/%d/%d", &new\_acc.dob.y, &new\_acc.dob.m, &new\_acc.dob.d);

printf("\nAddress:");

scanf("%s", new\_acc.address);

printf("\nContact number:");

scanf("%lf", &new\_acc.phone);

printf("\nDate of deposit:(yyyy/mm/dd)");

scanf("%d/%d/%d", &new\_acc.deposit.y, &new\_acc.deposit.m, &new\_acc.deposit.d);

printf("\nCITIZENSHIP NO.:");

scanf("%s", new\_acc.citizenship);

printf("\nAccount type(C/F1/F2/F5/S):");

scanf("%s", new\_acc.acc\_type);

printf("\nEnter the amount to deposit(Rs.):");

scanf("%f", &new\_acc.amt);

fwrite(&new\_acc, sizeof(new\_acc), 1, add);

fclose(add);

printf("\n\n\t\t\tLOADING");

for(i=0;i<5;i++)

{

fdelay(100000000);

printf(".");

}

system("cls");

printf("\n\n\n\n\n\t\t\t\t\tACCOUNT CREATED!");

again:

printf("\n\t\t\t\t\tEnter 1 to return to MAIN MENU and 0 to EXIT:");

scanf("%d", &test);

if(test==1)

{

main\_menu();

}

else if(test==0)

{

out();

}

else

{

printf("INVALID INPUT");

goto again;

}

}

void update()

{

int j,k,re;

FILE\*real,\*upd;

real=fopen("record.txt","r");

upd=fopen("newr.txt","w");

struct data up;

ret:

system("cls");

printf("\n\t\t\t\t\tEnter the Account no. to update information:");

scanf("%d", &j);

while(fread(&up,sizeof(up), 1, real))

{

if(j==up.acc\_no)

{

test++;

printf("\t\t\t\tAccount found!\nName:%s %s\nAddress:%s\nDOB:%d/%d/%d\nAccount No.:%d\nCitizenship no:%s\nContact number:%.0lf\n", up.first\_name, up.last\_name, up.address, up.dob.y, up.dob.m, up.dob.d, up.acc\_no, up.citizenship, up.phone);

printf("Which information you want to edit?\n1.Address\n2.Name\n3.Citizenship no.\n4.Contact number\n");

scanf("%d", &k);

if(k==1)

{

printf("Enter new Address:");

scanf("%s", up.address);

fwrite(&up,sizeof(up), 1, upd);

printf("\nEdited!!");

}

else if(k==2)

{

printf("Enter new Name(first name(space)last name):");

scanf("%s %s", up.first\_name, up.last\_name);

fwrite(&up, sizeof(up), 1, upd);

printf("\nEdited!!");

}

else if(k==3)

{

printf("Enter new citizenship no:");

scanf("%s", up.citizenship);

fwrite(&up, sizeof(up), 1, upd);

printf("\nEdited!!");

}

else if(k==4)

{

printf("Enter new contact number:");

scanf("%f", &up.phone);

fwrite(&up, sizeof(up),1, upd);

printf("\nEdited!!");

}

else

{

elsere:

printf("\nWrong input!\nEnter 1 to return to main menu, 0 to exit and 2 to retry:");

scanf("%d", &re);

if(re==1)

{

main\_menu();

}

else if(re==0)

{

out();

}

else if(re==2)

{

goto ret;

}

else

{

goto elsere;

}

}

}

else

{

fwrite(&up, sizeof(up), 1 , upd);

}

}

fclose(upd);

fclose(real);

real=fopen("record.txt", "w");

upd=fopen("newr.txt", "r");

while(fread(&up,sizeof(up), 1, upd))

{

fwrite(&up, sizeof(up), 1, real);

}

fclose(real);

fclose(upd);

remove("newr.txt");

if(test==0)

{

printf("\nNORECORD FOUND!!\nEnter 1 to return to main menu, 0 to exit and any number to retry:");

scanf("%d", &re);

if(re==1)

{

main\_menu();

}

else if(re==0)

{

out();

}

else

{

goto ret;

}

}

updagain:

printf("\n\t\t\t\t\tEnter 1 to return to MAIN MENU and 0 to EXIT:");

scanf("%d", &test);

if(test==1)

{

main\_menu();

}

else if(test==0)

{

out();

}

else

{

printf("INVALID INPUT\n");

goto updagain;

}

}

void check()

{

struct date current\_date;

struct data check\_acc;

char check\_ac[20], check\_accitizen[20];

int check\_acno, check\_test=0;

float total\_amount;

FILE\*chec;

chec=fopen("record.txt", "r");

reenter:

system("cls");

printf("\n\t\t\tSearh account by:\n\t\t\t1.account holder's name\n\t\t\t2.account number\n\t\t\t3.Citizenship no.\n");

scanf("%d", &test);

if(test==1)

{

printf("\t\tEnter the Account holder's first name:");

scanf("%s", check\_ac);

while(fread(&check\_acc, sizeof(check\_acc), 1, chec))

{

if(strcmp(check\_acc.first\_name,check\_ac)==0)

{

check\_test++;

printf("\n\t\t\t\t\tAccount no:%d\n\t\t\t\t\tName:%s %s\n\t\t\t\t\tDOB:%d/%d/%d\n\t\t\t\t\tCitizenship No:%s\n\t\t\t\t\tAddress:%s\n\t\t\t\t\tBalance:%f\n\t\t\t\t\tPhone number:%.0lf\n", check\_acc.acc\_no,check\_acc.first\_name, check\_acc.last\_name, check\_acc.dob.y, check\_acc.dob.m, check\_acc.dob.d, check\_acc.citizenship, check\_acc.address, check\_acc.amt, check\_acc.phone);

if(strcmp(check\_acc.acc\_type,"C")==0)

{

printf("\t\t\t\t\tYou get no INTEREST.");

}

else if(strcmp(check\_acc.acc\_type,"F1")==0)

{

total\_amount=CA\_forF(check\_acc.amt, 1, 0.085);

printf("\t\t\t\t\tYOU WILL RECEIVE Rs.%f AS TOTAL AMOUNT ON %d/%d%d", total\_amount, check\_acc.deposit.y+1, check\_acc.deposit.m, check\_acc.deposit.d);

}

else if(strcmp(check\_acc.acc\_type,"F2")==0)

{

total\_amount=CA\_forF(check\_acc.amt, 2, 0.1);

printf("\t\t\t\t\tYOU WILL RECEIVE Rs.%f AS TOTAL AMOUNT ON %d/%d%d", total\_amount, check\_acc.deposit.y+2, check\_acc.deposit.m, check\_acc.deposit.d);

}

else if(strcmp(check\_acc.acc\_type,"F5")==0)

{

total\_amount=CA\_forF(check\_acc.amt, 5, 0.13);

printf("\t\t\t\t\tYOU WILL RECEIVE Rs.%f AS TOTAL AMOUNT ON %d/%d%d", total\_amount, check\_acc.deposit.y+5, check\_acc.deposit.m, check\_acc.deposit.d);

}

else if(strcmp(check\_acc.acc\_type,"S")==0)

{

total\_amount=simple\_interest(check\_acc.amt, 0.0833, 8);

printf("\t\t\t\t\tYOU WILL RECEIVE Rs.%f AS INTEREST ON %d OF EVERY MONTH", total\_amount, check\_acc.deposit.d);

}

}

}

if(check\_test==0)

{

printf("\t\tNo record!\nEnter 2 to try again, 1 to return to main menu and 0 to exit:");

scanf("%d", &i);

if(i==2)

{

check();

}

else if(i==1)

{

main\_menu();

}

else if(i==0)

{

out();

}

}

}

else if(test==2)

{

printf("\t\tEnter the Account no.:");

scanf("%d", &check\_acno);

while(fread(&check\_acc, sizeof(check\_acc), 1, chec))

{

if(check\_acno==check\_acc.acc\_no)

{

check\_test++;

printf("\n\t\t\t\t\tAccount no:%d\n\t\t\t\t\tName:%s %s\n\t\t\t\t\tDOB:%d/%d/%d\n\t\t\t\t\tCitizenship No:%s\n\t\t\t\t\tAddress:%s\n\t\t\t\t\tBalance:%f\n\t\t\t\t\tPhone number:%.0lf\n", check\_acc.acc\_no,check\_acc.first\_name, check\_acc.last\_name, check\_acc.dob.y, check\_acc.dob.m, check\_acc.dob.d, check\_acc.citizenship, check\_acc.address, check\_acc.amt, check\_acc.phone);

if(strcmp(check\_acc.acc\_type,"C")==0)

{

printf("\t\t\t\t\tYou get no INTEREST.");

}

else if(strcmp(check\_acc.acc\_type,"F1")==0)

{

total\_amount=CA\_forF(check\_acc.amt, 1, 0.085);

printf("\t\t\t\t\tYOU WILL RECEIVE Rs.%f AS TOTAL AMOUNT ON %d/%d%d", total\_amount, check\_acc.deposit.y+1, check\_acc.deposit.m, check\_acc.deposit.d);

}

else if(strcmp(check\_acc.acc\_type,"F2")==0)

{

total\_amount=CA\_forF(check\_acc.amt, 2, 0.1);

printf("\t\t\t\t\tYOU WILL RECEIVE Rs.%f AS TOTAL AMOUNT ON %d/%d%d", total\_amount, check\_acc.deposit.y+2, check\_acc.deposit.m, check\_acc.deposit.d);

}

else if(strcmp(check\_acc.acc\_type,"F5")==0)

{

total\_amount=CA\_forF(check\_acc.amt, 5, 0.13);

printf("\t\t\t\t\tYOU WILL RECEIVE Rs.%f AS TOTAL AMOUNT ON %d/%d%d", total\_amount, check\_acc.deposit.y+5, check\_acc.deposit.m, check\_acc.deposit.d);

}

else if(strcmp(check\_acc.acc\_type,"S")==0)

{

total\_amount=simple\_interest(check\_acc.amt, 0.0833, 8);

printf("\t\t\t\t\tYOU WILL RECEIVE Rs.%f AS INTEREST ON %d OF EVERY MONTH", total\_amount, check\_acc.deposit.d);

}

}

}

if(check\_test==0)

{

printf("\t\tNo record!\nEnter 2 to try again, 1 to return to main menu and 0 to exit:");

scanf("%d", &i);

if(i==2)

{

check();

}

else if(i==1)

{

main\_menu();

}

else if(i==0)

{

out();

}

}

}

else if(test==3)

{

printf("\t\tEnter the citizenship no. of the Account holder:");

scanf("%s", check\_accitizen);

while(fread(&check\_acc, sizeof(check\_acc), 1, chec))

{

if(strcmp(check\_accitizen,check\_acc.citizenship)==0)

{

check\_test++;

printf("\n\t\t\t\t\tAccount no:%d\n\t\t\t\t\tName:%s %s\n\t\t\t\t\tDOB:%d/%d/%d\n\t\t\t\t\tCitizenship No:%s\n\t\t\t\t\tAddress:%s\n\t\t\t\t\tBalance:%f\n\t\t\t\t\tPhone number:%.0lf\n", check\_acc.acc\_no,check\_acc.first\_name, check\_acc.last\_name, check\_acc.dob.y, check\_acc.dob.m, check\_acc.dob.d, check\_acc.citizenship, check\_acc.address, check\_acc.amt, check\_acc.phone);

if(strcmp(check\_acc.acc\_type,"C")==0)

{

printf("\t\t\t\t\tYou get no INTEREST.");

}

else if(strcmp(check\_acc.acc\_type,"F1")==0)

{

total\_amount=CA\_forF(check\_acc.amt, 1, 0.085);

printf("\t\t\t\t\tYOU WILL RECEIVE Rs.%f AS TOTAL AMOUNT ON %d/%d%d", total\_amount, check\_acc.deposit.y+1, check\_acc.deposit.m, check\_acc.deposit.d);

}

else if(strcmp(check\_acc.acc\_type,"F2")==0)

{

total\_amount=CA\_forF(check\_acc.amt, 2, 0.1);

printf("\t\t\t\t\tYOU WILL RECEIVE Rs.%f AS TOTAL AMOUNT ON %d/%d%d", total\_amount, check\_acc.deposit.y+2, check\_acc.deposit.m, check\_acc.deposit.d);

}

else if(strcmp(check\_acc.acc\_type,"F5")==0)

{

total\_amount=CA\_forF(check\_acc.amt, 5, 0.13);

printf("\t\t\t\t\tYOU WILL RECEIVE Rs.%f AS TOTAL AMOUNT ON %d/%d%d", total\_amount, check\_acc.deposit.y+5, check\_acc.deposit.m, check\_acc.deposit.d);

}

else if(strcmp(check\_acc.acc\_type,"S")==0)

{

total\_amount=simple\_interest(check\_acc.amt, 0.0833, 8);

printf("\t\t\t\t\tYOU WILL RECEIVE Rs.%f AS INTEREST ON %d OF EVERY MONTH", total\_amount, check\_acc.deposit.d);

}

}

}

if(check\_test==0)

{

printf("\t\tNo record!\nEnter 2 to try again, 1 to return to main menu and 0 to exit:");

scanf("%d", &i);

if(i==2)

{

check();

}

else if(i==1)

{

main\_menu();

}

else if(i==0)

{

out();

}

}

}

else

{

printf("Invalid input, enter again:");

goto reenter;

}

fclose(chec);

again:

printf("\n\t\t\t\t\tEnter 1 to return to MAIN MENU and 0 to EXIT:");

scanf("%d", &test);

if(test==1)

{

main\_menu();

}

else if(test==0)

{

out();

}

else

{

printf("\n\t\tINVALID INPUT\n");

goto again;

}

}

void delete\_account()

{

system("cls");

int delac;

struct data del\_acc;

FILE\*del;

FILE\*delt;

del=fopen("record.txt", "r");

delt=fopen("newrec.txt", "w");

printf("\n\n\n\n\t\tEnter the account number to be deleted:");

scanf("%d", &delac);

while(fread(&del\_acc, sizeof(del\_acc), 1, del))

{

if(delac!=del\_acc.acc\_no)

{

fwrite(&del\_acc, sizeof(del\_acc), 1, delt);

}

}

fclose(del);

fclose(delt);

del=fopen("record.txt", "w");

delt=fopen("newrec.txt", "r");

while(fread(&del\_acc, sizeof(del\_acc), 1, delt))

{

fwrite(&del\_acc, sizeof(del\_acc), 1, del);

}

fclose(del);

fclose(delt);

remove("newrec.txt");

printf("\n\t\t\t\t\tDeleted successfully!!");

again:

printf("\n\t\t\t\t\tEnter 1 to return to MAIN MENU and 0 to EXIT:");

scanf("%d", &test);

if(test==1)

{

main\_menu();

}

else if(test==0)

{

out();

}

else

{

printf("INVALID INPUT");

goto again;

}

}

void view\_list()

{

int pc, retry;

struct data view;

struct loanee viewloan;

FILE\*fview,\*fviewloan;

fview=fopen("record.txt", "r");

fviewloan=fopen("loan.txt", "r");

system("cls");

printf("\n\t\t\t\t\tENTER THE PASSCODE TO ACCESS THE LIST:");

scanf("%d", &pc);

if(pc!=1234)

{

printf("\t\t\t\t\tWRONG PASSCODE!\n\t\t\t\t\tEnter 1 to return to main menu, 0 to exit and any other number to retry:");

scanf("%d", &retry);

if(retry==1)

{

main\_menu();

}

else if(retry==0)

{

out();

}

else

{

view\_list();

}

}

printf("\nPASSWORD MATCH!\nLOADING");

for(i=0;i<6;i++)

{

fdelay(100000000);

printf(".");

}

system("cls");

printf("\t\t\t\tThe list of account holders are:\n");

while(fread(&view, sizeof(view), 1, fview))

{

printf("\t\t\t\t\t%d\t%s %s\n", view.acc\_no, view.first\_name, view.last\_name);

}

printf("\t\t\t\tThe record of loanees are:\n");

while(fread(&viewloan, sizeof(viewloan),1, fviewloan))

{

printf("\t\t\t\t\t%d\t%s %s\n", viewloan.loan\_id, viewloan.first\_name, viewloan.last\_name);

}

again:

printf("\t\t\tEnter 1 to return to main menu or 0 to exit:");

scanf("%d", &i);

if(i==1)

{

main\_menu();

}

else if(i==0)

{

out();

}

else

{

printf("INVALID INPUT\n");

goto again;

}

}

void out()

{

system("cls");

printf("\n\n\n\n\t\t\t\t\tYOU HAVE EXITED THE BANK MANAGEMENT SYSTEM!!!");

}

void loan()

{

system("cls");

struct loanee a;

printf("\t\t\t\t\tLOAN\n");

printf("\t\t\t\t\tChoose an action:\n\t\t\t\t\t1.Taking loan\n\t\t\t\t\t2.Details of loan and loanee\n\t\t\t\t\t3.Clearing loan/debt");

scanf("%d", &i);

if(i==1)

{

system("cls");

FILE\*lon;

lon=fopen("loan.txt", "a");

printf("First name:");

scanf("%s", a.first\_name);

printf("Last name:");

scanf("%s", a.last\_name);

printf("DOB(yyyy/mm/dd):");

scanf("%d/%d/%d", &a.dob.y, &a.dob.m, &a.dob.d);

printf("Citizenship no:");

scanf("%s", a.citizenship);

printf("Address:");

scanf("%s", a.address);

printf("Contact number:");

scanf("%lf", &a.phone);

printf("Loan deposit:");

scanf("%s", a.loan\_deposit);

printf("Loan amount:");

scanf("%f", &a.loan\_amt);

printf("Date of loan granted(yyyy/mm/dd):");

scanf("%d/%d/%d", &a.loan\_taken.y, &a.loan\_taken.m, &a.loan\_taken.d);

printf("Assign loan id:");

scanf("%d", &a.loan\_id);

fwrite(&a, sizeof(a), 1, lon);

printf("successful!!");

fclose(lon);

printf("1 to main 0 to exit:");

scanf("%d", &i);

if(i==1)

{

main\_menu();

}

else if(i==0)

{

out();

}

}

else if(i==2)

{

int test=0;

float loan\_interest;

char check\_name[20];

system("cls");

FILE\*londet;

londet=fopen("loan.txt", "r");

retry:

printf("search by first name:");

scanf("%s", check\_name);

while(fread(&a, sizeof(a), 1, londet))

{

if(strcmp(check\_name,a.first\_name)==0)

{

test++;

printf("\t\t\tLoan id:%d\n\t\t\tName:%s %s\n\t\t\tAddress:%s\n\t\t\tPhone number:%.0lf\n\t\t\tLoan amount:%f\n\t\t\tLoan deposit:%s", a.loan\_id, a.first\_name, a.last\_name, a.address, a.phone, a.loan\_amt, a.loan\_deposit);

loan\_interest=simple\_interest(a.loan\_amt, 0.8333, 4);

printf("\nYou need to pay Rs.%f as loan interest on %d of every month.", loan\_interest, a.loan\_taken.d);

}

}

if(test==0)

{

printf("\t\tNO record!!\nenter 0 to exit, 1 to go to main menu and any other number to try again:");

scanf("%d", &i);

if(i==1)

{

main\_menu();

}

else if(i==0)

{

out();

}

else

{

goto retry;

}

}

fclose(londet);

term:

printf("\n\t\t\tEnter 1 to retur to main menu and 0 to exit:");

scanf("%d", &i);

if(i==1)

{

main\_menu();

}

else if(i==0)

{

out();

}

else

{

printf("\n\t\t\tInvalid input!Enter again!!");

goto term;

}

}

else if(i==3)

{

int acc\_d;

system("cls");

FILE\*londelete;

FILE\*tempfile;

londelete=fopen("loan.txt", "r");

tempfile=fopen("tempfile.txt", "w");

printf("\n\n\n\n\t\t\tEnter the loan id:");

scanf("%d", &acc\_d);

while(fread(&a, sizeof(a), 1, londelete))

{

if(acc\_d!=a.loan\_id)

{

fwrite(&a, sizeof(a), 1, tempfile);

}

}

fclose(londelete);

fclose(tempfile);

londelete=fopen("loan.txt","w");

tempfile=fopen("tempfile.txt","r");

while(fread(&a, sizeof(a), 1, tempfile))

{

fwrite(&a, sizeof(a), 1, londelete);

}

remove("tempfile.txt");

printf("\n\t\t\t\tDeleted successfully!!!\n\t\tEnter 1 to main and any other number to exit:");

scanf("%d", &i);

if(i==1)

{

main\_menu();

}

else

{

out();

}

}

}

void transact()

{

int acc, test=0;

float withdraw\_amt, deposit\_amt;

struct data transact\_acc;

FILE\*fptr,\*temp;

fptr=fopen("record.txt", "r");

temp=fopen("time.txt", "w");

transact\_again:

system("cls");

printf("\n\t\t\tCHOOSE THE ACTION:\n\t\t\t\t\t1.Withdraw\n\t\t\t\t\t2.Deposit\n");

scanf("%d", &i);

if(i==1)

{

printf("\t\tEnter the Account number of the account to withdraw from:");

scanf("%d", &acc);

while(fread(&transact\_acc, sizeof(transact\_acc), 1, fptr))

{

if(transact\_acc.acc\_no==acc)

{

test++;

if(strcmp(transact\_acc.acc\_type,"F1")==0

||strcmp(transact\_acc.acc\_type,"F2")==0

||strcmp(transact\_acc.acc\_type,"F5")==0)

{

printf("\t\tMoney can not be withdrawal as it is fixed account!");

fwrite(&transact\_acc, sizeof(transact\_acc), 1, temp);

}

else

{

printf("\tRs.%f is available.\nEnter the amount of money to withdraw:", transact\_acc.amt);

scanf("%f", &withdraw\_amt);

transact\_acc.amt-=withdraw\_amt;

fwrite(&transact\_acc, sizeof(transact\_acc), 1, temp);

printf("\tSuccessful!Now there is Rs.%f in the Account no.%d\n", transact\_acc.amt, transact\_acc.acc\_no);

}

}

else

{

fwrite(&transact\_acc, sizeof(transact\_acc), 1, temp);

}

}

fclose(fptr);

fclose(temp);

fptr=fopen("record.txt","w");

temp=fopen("time.txt","r");

while(fread(&transact\_acc, sizeof(transact\_acc), 1, temp))

{

fwrite(&transact\_acc, sizeof(transact\_acc),1, fptr);

}

fclose(fptr);

fclose(temp);

remove("time.txt");

if(test==0)

{

printf("\n\t\tNo record!\nEnter 2 to try again, 1 to return to main menu and 0 to exit:");

scanf("%d", &i);

if(i==2)

{

goto transact\_again;

}

else if(i==1)

{

main\_menu();

}

else if(i==0)

{

out();

}

}

else

{

printf("\n\t\tTransaction Completed \nEnter 1 to return to main menu and 0 to exit:");

scanf("%d", &i);

if(i==2)

{

goto transact\_again;

}

else if(i==1)

{

main\_menu();

}

else if(i==0)

{

out();

}

}

test=0;

}

else if(i==2)

{

printf("\t\tEnter the Account number of the account to deposit to:");

scanf("%d", &acc);

while(fread(&transact\_acc, sizeof(transact\_acc), 1, fptr))

{

if(acc==transact\_acc.acc\_no)

{

test++;

if(strcmp(transact\_acc.acc\_type,"F1")==0

||strcmp(transact\_acc.acc\_type,"F2")==0

||strcmp(transact\_acc.acc\_type,"F5")==0)

{

printf("\tMoney can not be deposit as account is fixed!!");

fwrite(&transact\_acc, sizeof(transact\_acc), 1, temp);

}

else

{

printf("\tRs.%f is available.\nEnter the amount of money to deposit:", transact\_acc.amt);

scanf("%f", &deposit\_amt);

transact\_acc.amt+=deposit\_amt;

fwrite(&transact\_acc, sizeof(transact\_acc), 1, temp);

printf("\tSuccessful!Now there is Rs.%f in the Account no.%d\n", transact\_acc.amt, transact\_acc.acc\_no);

}

}

else

{

fwrite(&transact\_acc, sizeof(transact\_acc), 1, temp);

}

}

fclose(fptr);

fclose(temp);

fptr=fopen("record.txt","w");

temp=fopen("time.txt","r");

while(fread(&transact\_acc, sizeof(transact\_acc), 1, temp))

{

fwrite(&transact\_acc, sizeof(transact\_acc),1, fptr);

}

fclose(fptr);

fclose(temp);

remove("time.txt");

if(test==0)

{

printf("\n\t\tNo record!\nEnter 1 to return to main menu, 0 to exit and any other number to retry:");

scanf("%d", &i);

if(i==1)

{

main\_menu();

}

else if(i==0)

{

out();

}

else

{

goto transact\_again;

}

}

else

{

printf("\t\tTransaction Completed \n Enter 1 to return to main menu and 0 to exit:");

scanf("%d", &i);

if(i==2)

{

goto transact\_again;

}

else if(i==1)

{

main\_menu();

}

else if(i==0)

{

out();

}

}

test=0;

}

else

{

printf("Error!!\n\tEnter 1 to return to main menu and 0 to exit:");

scanf("%d", &i);

if(i==2)

{

goto transact\_again;

}

else if(i==1)

{

main\_menu();

}

else if(i==0)

{

out();

}

}

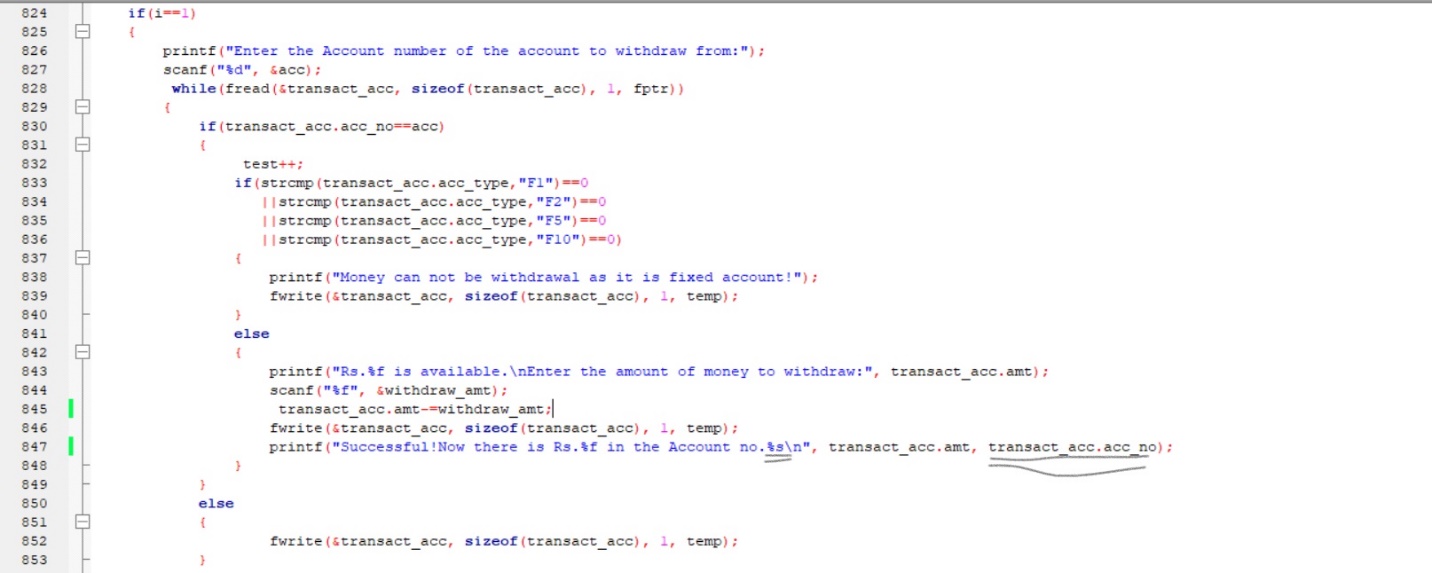
}

## **TESTING AND DEBUGGING**

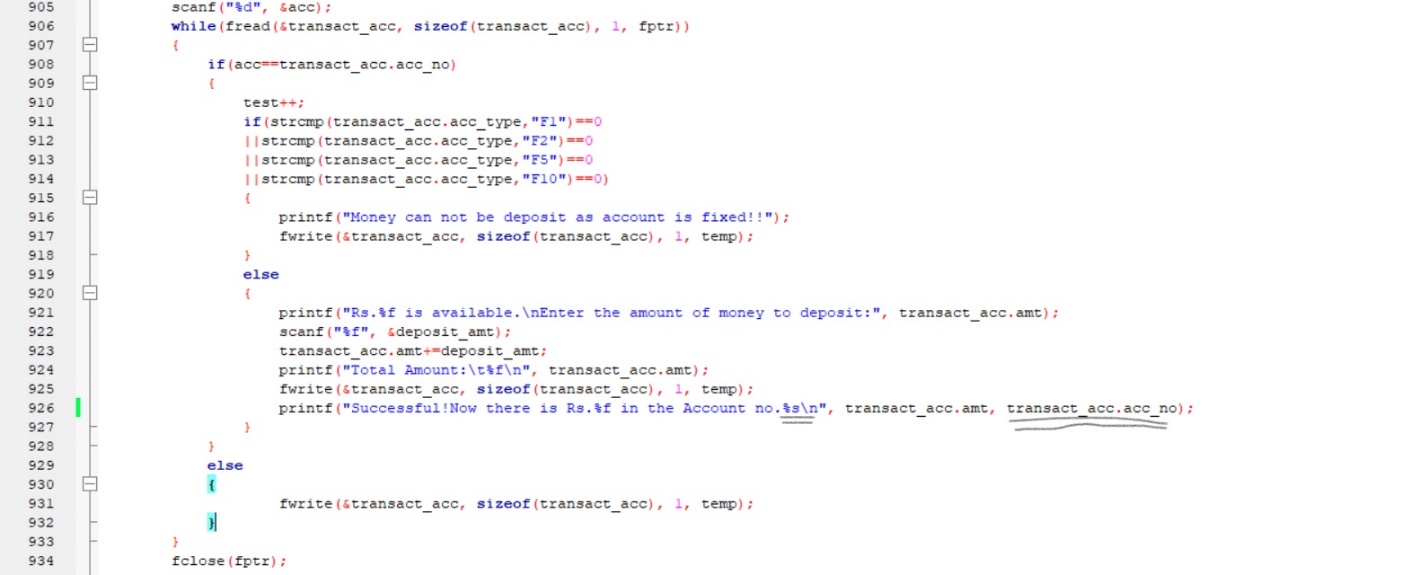
Each and every codes were tested before making it a program. The following table shows the testing sequence of the codes:

|  |  |  |
| --- | --- | --- |
| Function | Test | Remarks |
| main() | Login codes work as desired | Test positive |
| fdelay(int a) | The fdelay works as its use freezes screen for a while | Test positive |
| simple\_interest(amt, time, rate) | The function works as it calculates the simple interest as desired | Teat positive |
| CA\_forF(amt, time, rate) | The compound amount is given by this function as desired. | Test positive |
| main\_menu() | The switch statement works properly | Test positive |
| new\_account() | The entered data is successfully stored in record.txt. | Test positive` |
| out() | The function exits the program | Test positive |
| view\_list() | The function lists the clients as desired | Test positive |
| check() | The entered information shows the detail of the account holder. | Test positive |
| delete\_account() | The output is as desired after using the check() function the deleted account was no more in the file. The temporary file gets removed after the completion of the function as desired. | Test positive |
| update() | After using update function the same account was checked through check function and the data shown was updated information. The temporary file gets deleted as desired after the completion of the function. | Test positive |
| loan() | The loan function works as desired, the grant of new account, checking new account and clearing loan all work as desired. The temporary file gets removed after the completion of the function as desired. | Test positive |
| transact() | The transact function does not work as desired, it freezes while depositing or withdrawing money from current and saving account, and the temporary file does not get removed after the transact function is closed. While there is no error if the account number entered doesn’t exist or if the account is fixed. | Error, test is negative |

The above tests were run and there was problem in the transact function, it took a while to pick out the error. The transact function wasn’t working properly due to syntax error, the acc\_no was being represented by wrong conversion character so it was holding back the function from running properly.



Figure



Figure

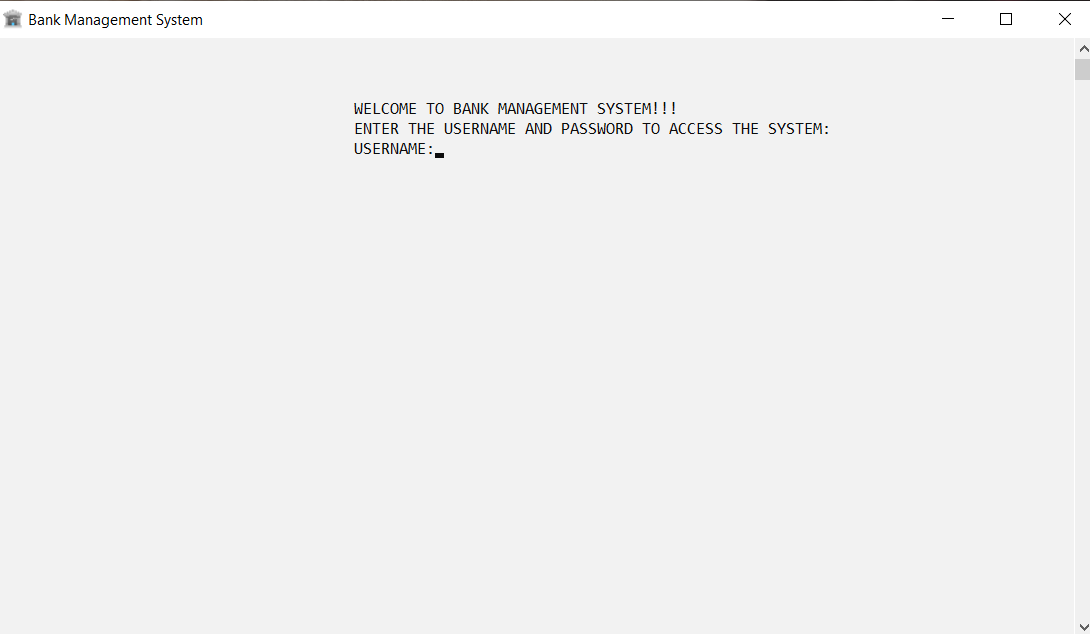
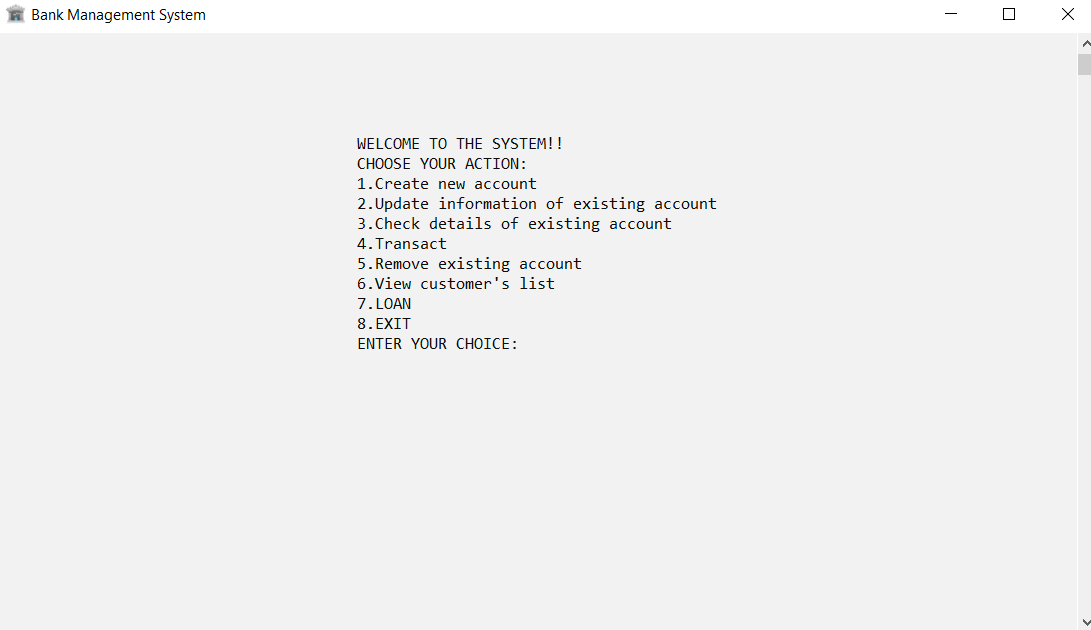
The above pictures (screenshots) show the error in the code, transact\_acc.acc\_no is integer data type but instead of using "d" as conversion character we accidently wrote "s" as its conversion character which is conversion character of string. Due to this error the program did not run after this and the temporary file did not get removed.   
This error was debugged by replacing conversion character "s" with "d".

# **DETAILS ABOUT THE MANAGEMENT SYSTEM**

In this bank management system we stored some data for demo propose in record.txt and loan.txt. Also there are some details to be mentioned:

acc\_type

|  |  |  |
| --- | --- | --- |
| Acc\_type | Interest rate | Time |
| C (Current Account) | - | - |
| S (Saving Account) | 8% p.a. simple interest | Monthly |
| F1 (Fixed account for a year) | 8.5% p.a. compound interest | 1 year |
| F2 (Fixed Account for 2 years) | 10% p.a. compound interest | 2 years |
| F3 (Fixed Account for 5 years) | 13% p.a. compound interest | 5 years |

The terminal;  
Figure 

Figure

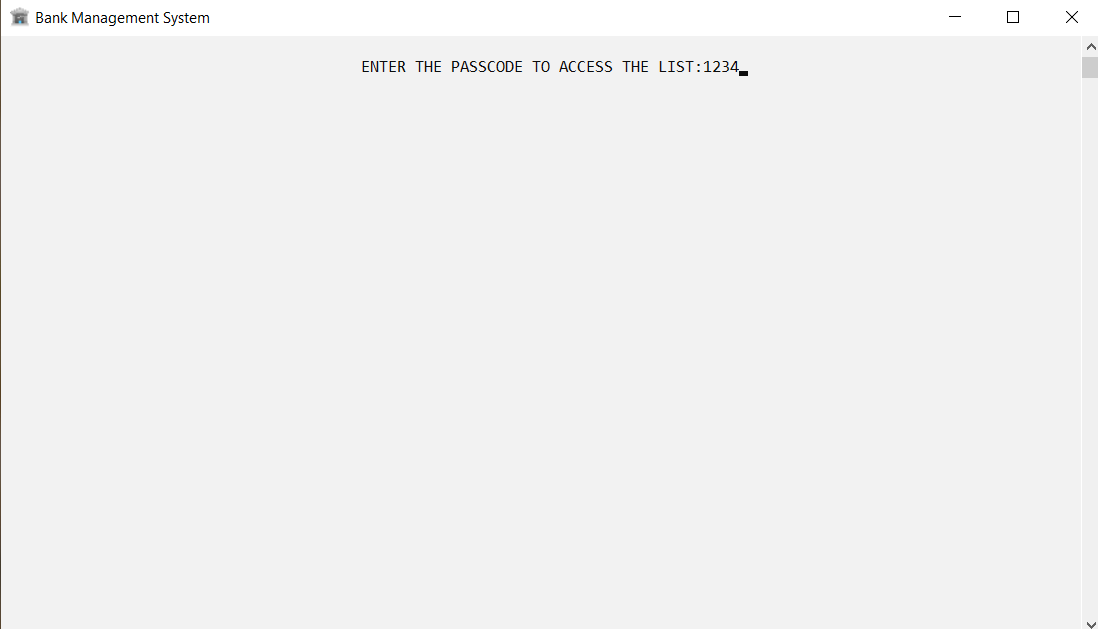
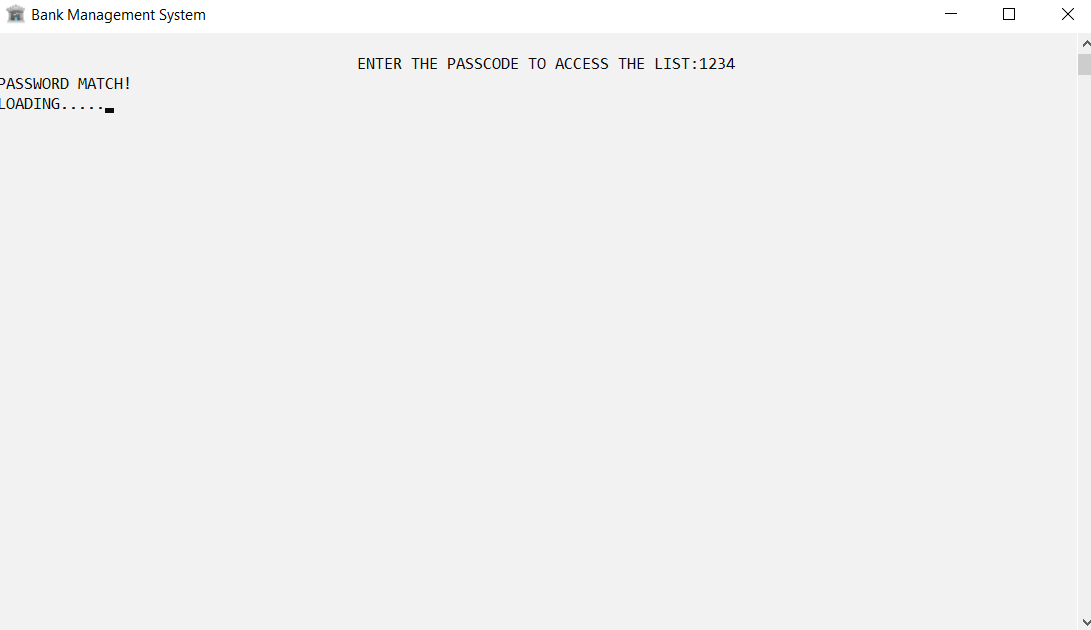
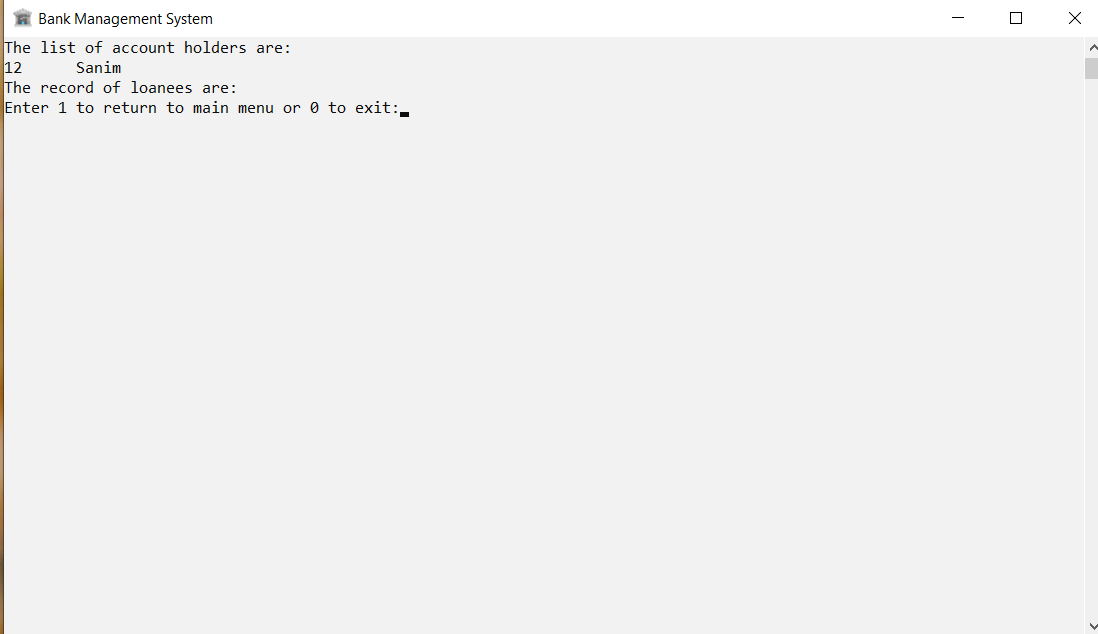


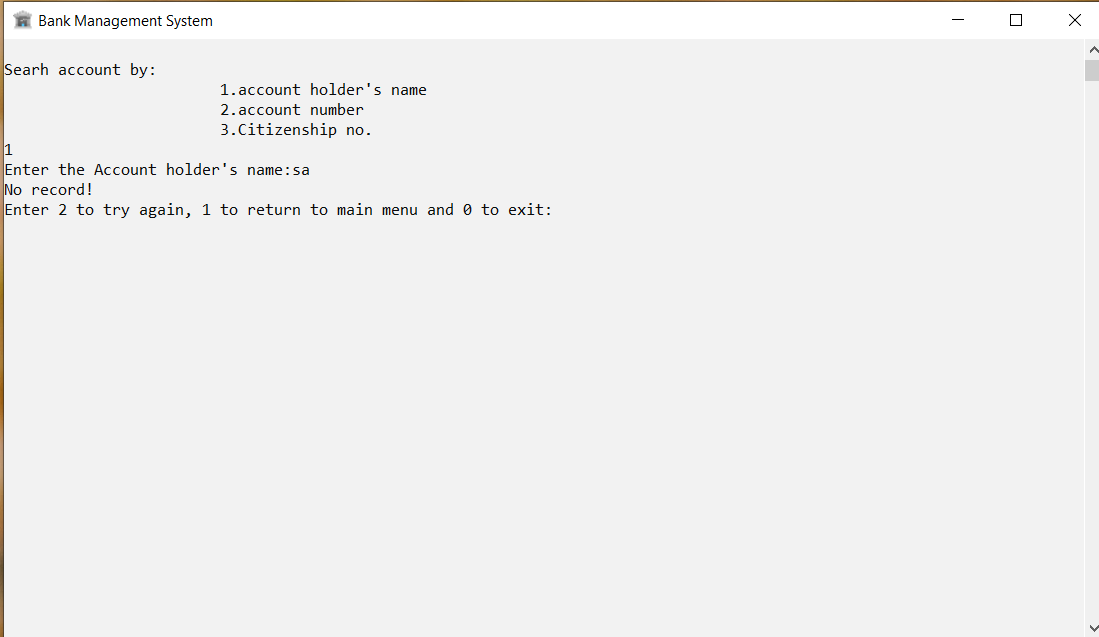
Figure 

Figure

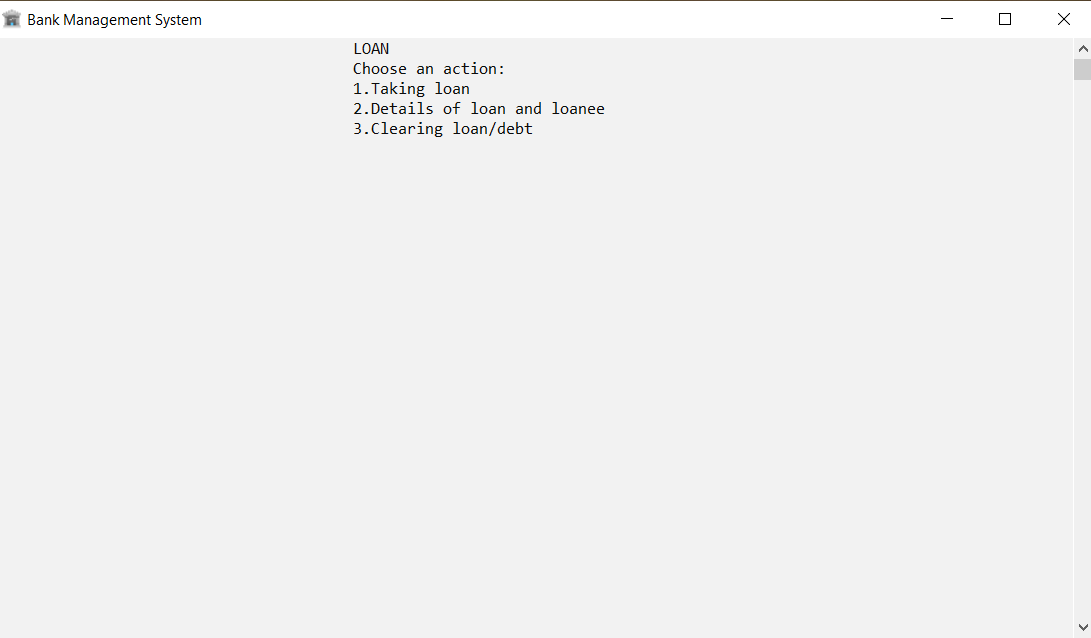


Figure 

Figure



Figure



Figure

# **DISCUSSION AND CONCLUSION**

This mini project had mainly following objectives:

1. Understanding the concept of C programming language/ programming language as a whole.
2. Problem analysis and solving.
3. Use of the C programming language to solve the problem.

These objectives were fulfilled during this project and thus a "Bank Management System" was created in C programming language after analyzing and solving the problems. We used GCC compiler to compile this program. Although this mini project is a basic management system and only covers vital areas of the bank but it however doesn’t totally describe all the functions of bank. Also being written in C programming language without use of graphics it makes the interface quite unattractive.