

**A PROJECT REPORT**  
**ON**  
**BANKING AUTOMATION SYSTEM**  
**OF**



SUBMITTED BY

**DURGESH KUMAR KAUSHIK**  
**ROLL NO : 2120140080058**

In partial fulfillment for the award of the degree of  
**BACHELOR OF COMPUTER APPLICATIONS**



Under the regular guidance of  
**Mr. PRAMOD KUMAR SINGH**

## **ACKNOWLEDGEMENT**

In this fragment of my project. I would like to remember those entire people who are part and parcel of this project. First of all, I am thankful to my project guide Mr. Pramod kumar singh for his divine help and encouragement. Whenever I found myself in problem, honourable sir proved himself a deified embodiment. I am again thankful to my sir for sacrificing his knowledge repertory and precious moments on me. Secondly I am thankful to all staffs of bank, which provided me every vital information whenever I needed and helped me very carefully listening to me. I can't forget to thank to all my family members who ever delighted me in the situation of perplexity. I am also thankful to all my seniors, colleagues, who directly or indirectly helped me in the completion of my project and testing it, to fix bugs.

I owe great debt to all of them who misspent this period of busy schedule.

DURGESH KUMAR KAUSHIK

## DECLARATION

We hereby declare that this project on “Banking Automation System ” have been designed and done by us.

In completing this project we had to undergo a lot of research and a lot of thought was put into it before completing this work. We had searched various books, web-sites in order to complete this assignment. We had asked many persons who were related in this field for their generous help.

We have tried our level best to make it free of any possible errors. If, any there is any error then it is just a human error and we are solely responsible for it and are sorry for it .

Thanking you,

DURGESH KUMAR KAUSHIK

## ABSTRACT

Software may be applied in any situation for which pre-specified set of procedural steps has been defined. It is a means for optimal utilisation of time, space and above all there is more of customers satisfaction the efficiency and arithmetical accuracy of system is increased and hence less errors.

Banking in general plays a major role in day to day of human life, previously before the usage of computers in the banking sector, maintaining the records of customers was a time taking task and there was a risk of error full manipulation of records. From the advent of computers into the banking sector, it has become an easy job to maintain the records and the day to day transactions of the bank.

We have considered only saving account for the “**BANK OF BARODA**, REGIONAL OFFICE, S.K.PURI, PATNA ”.

Any customer can open his/her account, he can see his balance, the day book and balance sheet is updated regularly and the customer will get an interest at the rate of 4% per annum.

This project is being developed by keeping in mind the requirement of a bank, the types of transactions, which take place in a bank for saving account. This project provides to a certain

extent the functions that are performed in a bank. The automation of bank i.e. proposed system plays vital role in development of any economy.

This project can perform the below mentioned functions:-

1. Creating a saving bank account
2. Closing a saving bank account
3. Debiting from a saving bank account
4. Crediting from a saving bank account
5. Modifying the saving bank account
6. Generating various types of reports like list of accounts  
    , individual account, monthly reports
7. Maintaining the records of saving bank account

## TABLE OF CONTENTS

<u>Sl no.</u>	<u>Topic</u>	<u>Page no.</u>
1.	INTRODUCTION	6
2.	BRANCH COMPUTERISATION	8
3.	TOOLS/PLATFORM,LANGUAGE TO BE USED	30
4.	WHY USE C++	31
5.	DATA DICTIONARY	34
6.	DATA FLOW DIAGRAM(DFD)	35
7.	INPUT AND OUTPUT SCREEN	38
8.	CODING	57
9.	CONCLUSION	110
10.	BIBLIOGRAPHY	111

## INTRODUCTION

Banking transaction system for saving account plays a vital role in the development of any economy. Today this bank has come a long way from the “money charges” to the modern day finance managers. They have become essential requirement for an individual whether in business, in service, household or even a labourer. Today it has become proved itself as a backbone of economical development of our country. A mass part of our nation regarding the growth, progress of human financial structure. There is nothing alternative thing except banking organisation. On the point of education, there are a number of banking organisation, which financially support the student who have really talent to achieve higher education inside the country or abroad.

The growth of Indian bank network since nationalisation in 1969 is he from 8000 in 1969 48000 in 1993. The transition from classes to mass banking resulted in huge rise in no. of transactions that called for mechanization various commissions and committees were set up to study the process of atomisation in banks they recommended adaptation of computer and information technology at 3 levels.

1. Head office
2. Regional & Zonal office
3. Branch level

*My project considers only automation at branch level*

Computer industry has witnessed changes through the process of continuous innovation and technical up gradation of systems, both for hardware and software where as banking is concerned the financial service industry in India is now facing tremendous competition from foreign banks and other private banks and financial institution who appeared in areas equipped with improved information technology .The scope of mechanisation of banking is vast since majority of banking activity involves data/information capturing and routine processing .

To remain ahead in run the industry has to improved customer service, housekeeping to procure higher magnitude of business volume introduction of **TOTAL BRANCH**

**AUTO-MATION IN BANKING** is considered to be a right approach.



## **BRANCH COMPUTERISATION**

This branch is the scene of action in bank 'structure' where major activity is no interface with customer, provide improved Service to them mobilise resources and generate profit by raising productivity. It is challenging task to working in automated environment by introducing suitable computerisation at branch level.

### **Benefit's to clients**

1. Passbook updated immediately
2. Single window service
3. Withdrawal of cash immediately

4. Round 'O' clock access through remote terminals
5. Quality customer service

### **BENEFITS TO BANK MANAGEMENT**

1. Ease in handling more business in same premises
2. Effective automation of branch
3. Increased portability
4. Raising productivity of staff and effective utilisation of manpower
5. Safeguard against fraud and forgery

6. Backup of records
7. Crisis management
8. Facility to know balance sheet and P & L statement.

### **BENEFITS TO EMPLOYEES**

1. Eliminating drudgery of repetitive work.
2. Faster information retrieval.

3. Complete satisfaction of employees and congenially work.

### **HIGHLIGHTS OF THE SYSTEM**

1. Independently implementable system
2. Remote customer terminal
3. Configurable security
4. Full backup system
5. Less paperwork
6. Risk monitoring system
7. Bank office automated

## **SECURITY FEATURES**

The G.B.A. is on line branch computerisation. Any security lapse will prove very costly to the bank. Accordingly the system must be fault tolerant and should not fail during office hours. Encrypted password for users and validated for any function with multiple levels of security in addition to Unix entry validation by password. The fault tolerant features are designed as follows.

1. Password
2. Access restriction
3. Audit trials
4. Check sums

## **OUTLOOK OF THE BANKING SYSTEM**

The major components of banking system are:

1. Accounts
2. Cashbook
3. Ledger
4. Administration
5. Reports

## ACCOUNTS

1.1 Saving Account

1.2 Current Account

1.3 Reccuring Account

1.4 Fixed Account

1.5 Interest Calculation

## **SAVING BANK ACCOUNTS**

Saving bank account is opened with the name of individuals singly or jointly with other individuals. The types of accounts opened by saving bank accounts are:

1. Personal Accounts
2. Minor Account
3. Joint Account

An opening form should be written. An introduction from a person known to the bank is necessary. When an account is opened in joint names of guardian and minor a survivorship, form should be signed by the guardian. A declaration for the date of birth of a minor is necessary then an account is opened in his name. Nomination can also be obtained if so, enter



like same in the nomination register obtained the initials of the manager is account opening form. Have the account holders signature recorded on the specimen signature sheet request the depositor to tender initial deposit and cash. Given account no. Enter in appropriate ledger maintained. Index the account in alphabetical index maintains separate ledger and S.NO.

**For accounts with cheque facility and without it.**

Complete the ledger having from account opening form and other connected forms enter cheque issue, in ledger. When account is opened in joint names indicate operation "E" on "S" or "jointly" or "severely". If nomination is available enter same in nomination register. Issue of passbook with all particulars after making proper entries in saving bank passbook register. Issue chequebook, in case of amount with

cheque facility write the account no. on account opening form and the specimen signature card.

### **PARAMETERS FOR SAVINGS ACCOUNTS**

1. Number of Months in which Interest is paid [06]
2. Debit Interest to be Charged for Overdrawing in SB A/C [14%]

- |  |          |
|--|----------|
| 3. Minimum Credit Interest Rate                | [04%]    |
| 4. Maximum Credit Interest Rate                | [05%]    |
| 5. Interest Rounding-off Option                | [H 1.00] |
| 6. Minimum Service Charges                     | [Rs.20]  |
| 7. Minimum Balance to be kept in Staff Account | [Rs.250] |
| 8. Minimum Balance for Cheque Book Account     | [Rs.500] |
| 9. Minimum Balance for Non-cheque Book Account | [Rs.500] |
| 10. Minimum Balance for Locker Account         | [Rs.500] |
| 11. Withdrawal slips to be permitted (y/n)     | [Y]      |

### **ISSUE OF CHEQUE BOOK :**

Book required cheque register receive requisition from for a chequebook. Refer to ledger; see that requisition in from the current cheque book. Issue chequebook after writing account no., ledger no. All leaves, and date of issue, and name of depositor, on requisition from within the new chequebook. Record particulars of chequebook issued in the ledger enter the particular of cheque book register on delivery of chequebook to customer obtain the sign on chequebook register. When an account is transferred to another office the constituent may be advised to use the same chequebook by altering the name of office and also the ledger no. By authorisation under his full signature.

## **PAYMENT OF CHEQUES**

Payment of cheques in current cash credit and saving account can be described by the following procedure: book required: ledger, cheques repaired and returned register.

Receive cheque and see that it is drawn on your office and pertain to your ledger.

Scrutinize the cheque and ensured the following:

A: Account no

B: Cheque series

And update the cash book for daily transactions.

### **WITHDRAWAL/OTHER THEN CHEQUES :**

#### **Book required : Ledger**

Receive completed withdrawal form along with passbook, refer the withdrawal to the manager if account is not operated for more than one year. See that a proper date is mentioned. Account no. must be mentioned correctly, amount should be in whole round rupee only. Refer to the ledger account and find out that the amount to be withdrawal is withdrawal in the ledger. And complete the passbook entries.

### **STANDING INSTRUCTIONS AND STOP PAMENTS :**

Book required: current account /cash credit/ saving bank ledger/ monthly/ quarterly/ term deposit interest payment register daily list, stop cheque register, standing instruction binder.

Obtain standing instructions from part. File standing instructions sheet in the binder according to the date of month to which instruction is to be carried out. Make suitable entries in the index charges to be recovered / exchange postage commission etc. should be indicated.

### **STOP PAYMENTS OF CHEQUES:**

Record the particulars of stopped cheques in stopped cheque register with date and time of receipt of letter. Record in red ink ledger sheet the particulars of stopped cheque, write to below the last transaction. Enter the particulars of the stopped cheques in list of the stopped cheque, in case of dated cheques cancel stop when it becomes out of data. Loss of bank cheque forms should be noted in the last document register and in ledger in stop list.



### **CASH BOOK:**

At the end of the day the cash book are written accordingly. Collect and sort vouchers ledger wise, folio with write the account in folio no., name of account holder in the representative column and the amount of vouchers under debit and credit. A separate day book is maintained for current and saving account.

### **BALANCING OF LEDGERS:**

Balancing of ledgers is by far very important part of day to day banking and it accomplishes as follows:

Balance books are maintained in set of two volumes A and B. Two balancing exercises should be of same volume take out such volume of balancing book set not used for last exercise write the day and day balancing on folio.

Taking the concerned correct ledger of down balances from accounts on same day ensure that the day vouchers have been accounted properly. Delete the account closed and added new account open after last exercise in volume total each page summarise page totals, check the summary total the progressive balance of ledger. If the total agrees in out each page total and summary figure. If the balance does not agree recheck totals otherwise verify the balances allotted with those in ledgers.

## **OBJECTIVE OF THE PROJECT**

The main objective of this project is to develop a system to effectively monitor and manage the internal financial structure of " BANK OF BARODA ,S.K.Puri Branch,PATNA ". Efforts will be made to cover each and every aspects of "Banking Transaction System" process for saving account. The software will be able to manage set –wise procedure ranging.

- Keep track of all account holders.
- To develop a software or transaction system for saving account, recording day-to-day transactions, updating and interest calculation.
- Effectiveness of present working system.
- Reduces involvement of manpower.
- To full-fill the partial requirement of BCA(final) M.U.Bodhgaya

## **SOFTWARE DEVELOPMENT LIFE CYCLE**

Software systems pass through two principal phases during their life cycle.

1. The development phase.
2. The operation and maintenance phase.

The development phase begins when the need for the product is identified; it ends when the implemented product is tested and delivered for operation. Operation and maintenance include all activities during the operation of the software, such as fixing bugs discovered during operation, making performance enhancements, adapting the systems to its environment, adding minor features, etc.

#### ► **PROBLEM DEFINITION:-**

The first stage in the development process understands the problem in question and its requirements. At this point, the managers and software specialists decide whether it is feasible to build the system.

#### ► **ANALYSIS:-**

Analysis phase delivers requirements specification. Specification must be

carefully checked for suitability, omission, inconsistencies and ambiguities. In this project the analysis is done with the help of the Branch Manager and staffs after considering the needs of the bank i.e. what is the basic need of the bank and what it is requiring for the automation of the bank. After going through the entire requirement we have come to the next phase of the Software Life Cycle.

#### ► **DESIGN: -**

Design is the process of mapping systems requirements defined during analysis to an abstract representation of a specific-system implementation, meeting the cost and performance constraints. The purpose of design phase is to specify a particular software system that will meet the stated requirements. This project is divided into modules and their interactions. The modules may then be further decomposed into sub-modules and procedures until each module can be implemented easily.

#### ► **CODING/IMPLEMENTATION :-**

Once the specification and the design of the software are over, the choice of a programming language remains as one of the most critical aspect in producing reliable software. Implementation involves the actual production of code. Although it is one of the important phases, it takes only 20% of the total development time. In this project the generation of program is done with the use of appropriate programming language i.e.

C++.

### ► **TESTING:-**

Testing is the process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies the specified requirements. Normally, most of the testing and debugging is done after the system has been implemented. A large percentage of errors are discovered during testing originates in the requirement and design phase.

The techniques that have been purpose for unit testing include the following: -

1. Path testing: - Each possible path from input to output is traversed once.

2. Branch testing: - Each path must be traversed at least once.
3. Functional testing: - Each functional decomposition is tested at least once.
4. Special value testing: -Testing for all values assumed to cause problems.

All the modules that have been developed before and tested individually are put together – integrated – in this phase and tested as a whole system. After doing this in the project the project will be almost ready to be delivered.

#### ► **MAINTENANCE: -**

Once the system passes all the tests, it is delivered to the customer and enters the maintenance phase. Any modification made to the system after initial deliveries are usually attributed to this phase. Basically in this phase the software will be maintained and this is

done prior to release of software and I have reviewed the software and assured that all the documentation is available.

### ► **CHANGE MANAGEMENT:-**

Changes to systems are bound to happen many times either during the system design, or after complete implementation of the system, or during system operation. Hence, it is very essential to define change management process.

A change is an alteration to the project scope, deliverables or milestones that would affect the project cost, schedule, or quality. Change is inevitable and occurs during the course of project as shown in figure. The project manager is responsible for change control. Different categories of change exist: mandatory, critical, and nice to have. The project manager can accept the change request, or reject the change request, or return the change request for further investigation or clarification.



Some of the factors causing changes in a project are the following:-

1. Customer misunderstanding.
2. Inadequate specification.
3. New customer request.
4. Organization changes.
5. Government regulation.

## **Tools / Platform, Languages to be used**

### **HARDWARE**

Processor: Pentium (500 MHz)

Main memory: -256 MB

HDD : 80GB,

FDD : 1.44MB,

SVGA Color monitor,

Multimedia Kit (DVD Combo Drive),

Sound card,

Speaker

Microphone

INTEX mouse, Multimedia Keyboard

## SOFTWARE

Operating system: - windows-XP (professional)

Programming language: - C++

Documentation: - MS-WORD2003

### Why use C++?

C++ is a superset of c with features designed to support oops. OOP is a new way of organizing code and data that promises increased control over the complexity of the software development process.

Yet, C being a structured programming language offered the ease of software development but failed to support maintenance of large code. With C++, it is much easier to build and maintain a really big code. Object – Oriented Programming is cantered around new concepts such as Objects, Classes, Polymorphism, Inheritance, etc.

It is well-suited paradigm for the following: -

Modelling the real-world problem as close as possible to the user's perspective.

Interacting easily with computational environment using familiar metaphors. Constructing reusable software components and easily extendable libraries. Easily modifying and extending implementations of components without having to recode every thing from scratch.

A language's quality is judged by twelve important criteria. They are a well defined syntactic and semantic structure, reliability, fast translation, efficient object code, orthogonally, machine independence, provability, generality, and consistency with commonly used notations, subsets, uniformity, and extensibility. The various constructs of OOP languages (such as c++) are designed to achieve these with ease.

Unlike, the traditional methodology (Function-Oriented Programming), objectoriented programming emphasizes on the data rather than the algorithm. In oops, data is compartmentalized or encapsulated with the associated functions and this compartment or capsule is called an object.

In the OOP approach, the problem is divided into objects , whereas in functionoriented programming the problem is divided into functions. Although, both approaches adopt the same philosophy of divide and conquer, OOP conquers a bigger region, while function-oriented programming is content with conquering a smaller region. ObjectOriented Programming contains function-oriented programming and so OOP can be referred to as the superset of function-oriented programming and hence, it can be concluded that OOP has an edge over function-oriented programming.

Some of the most prominent features of C++ are classes, operator and function overloading, free store management, constant types, references, inline and friend function, inheritance, virtual functions, streams for console and file manipulations, templates and exception handling.

**Advantage:-**

Information hiding and data abstraction increase reliability and help decouple the procedural and representational specification from its implementation.

Dynamic binding increases flexibility. Inheritance coupled with dynamic binding enhances the reusability of code, thus increasing the productivity of a programmer. Code reuse is possible in conventional language as well, but object-oriented language greatly enhances the possibility of reuse.

Object –Orientation provides many other advantages in the production and maintenance of software; shorter development time, high degree of code sharing and malleability.

**DATA DICTIONARY**

In this software two files will be created with given record structure.

(1) **INITIAL.DAT**

<b><u>ATTRIBUTE</u></b>	<b><u>DATA TYPE</u></b>	<b><u>LENGTH</u></b>
ACCNO	Numeric	10
NAME	Character	30
ADDRESS	Character	30

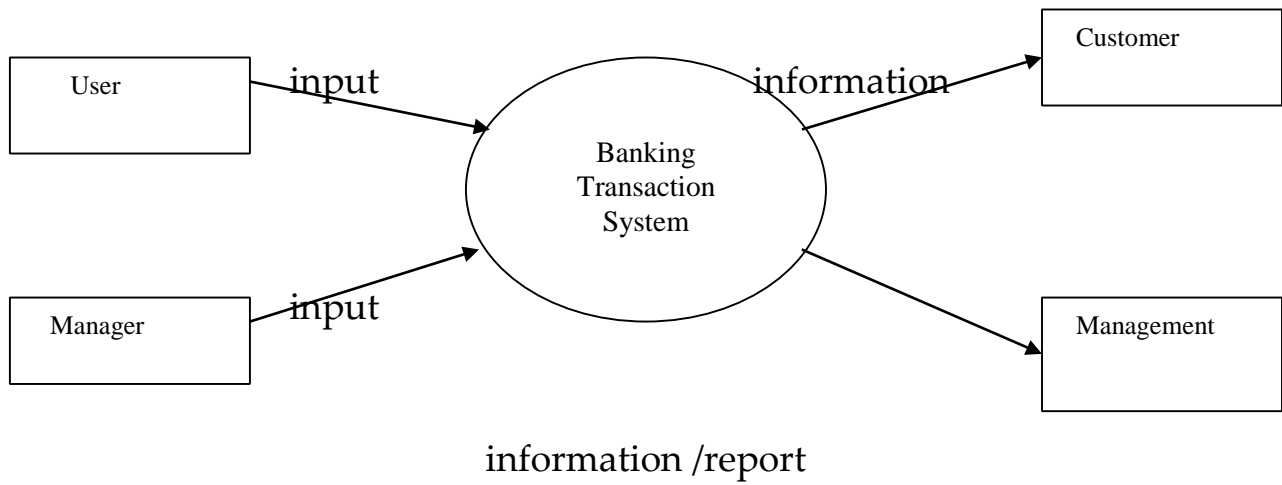
DEPOSIT	Numeric	10
BALANCE	Numeric	10

(2) **LEDGER.DAT**

<b><u>ATTRIBUTE</u></b>	<b><u>DATA TYPE</u></b>	<b><u>LENGTH</u></b>
ACCNO	Numeric	10
DD/MM/YY	Numeric	8
TRANS_TYPE	Character	10
T_TRANS	Character	1
DEPOSIT	Numeric	10
INTEREST	Numeric	10
BALANCE	Numeric	10
TOT_BALANCE	Numeric	30

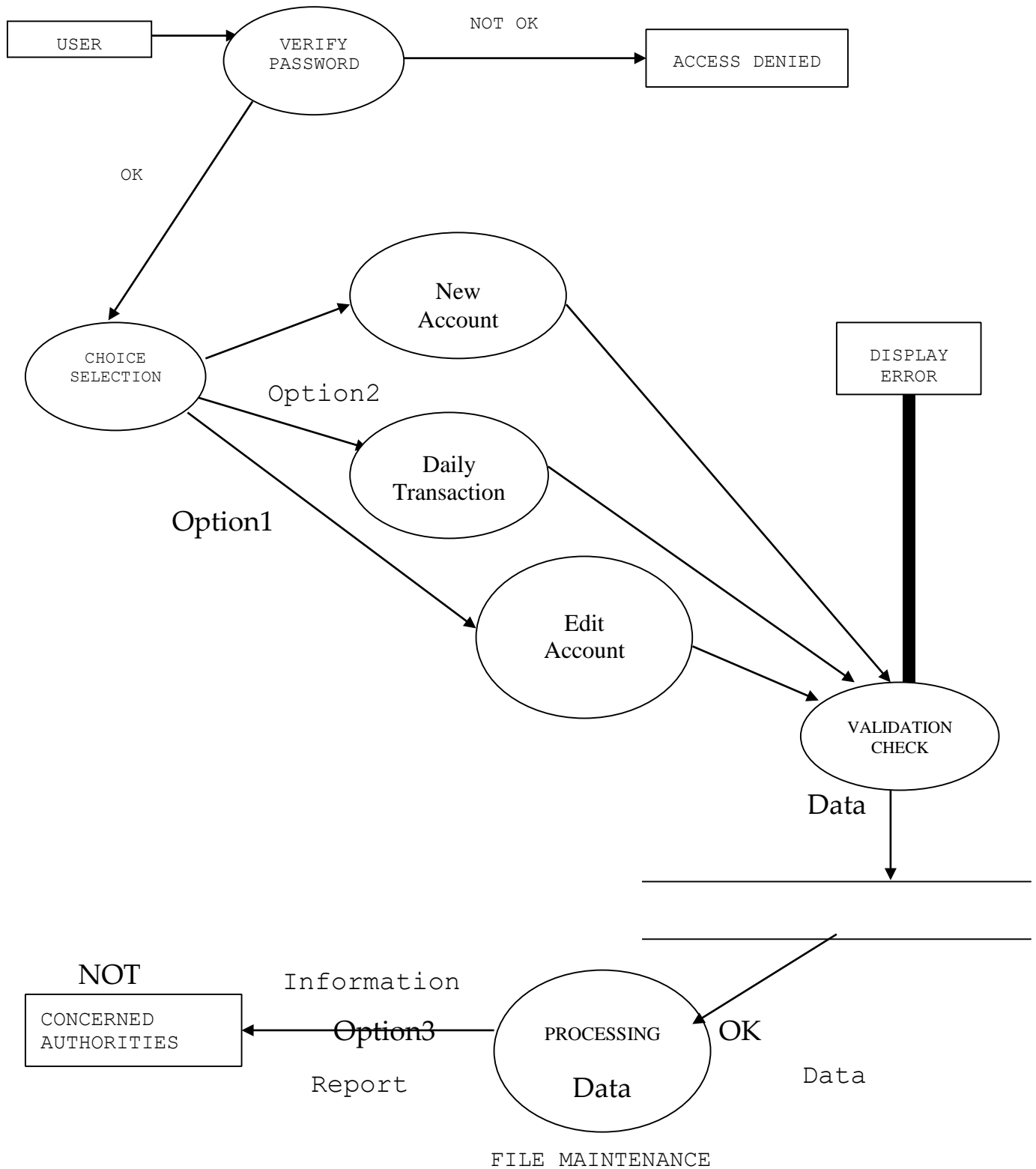
► Note: - Key fields are Bold.

### Data Flow Diagram(DFD)



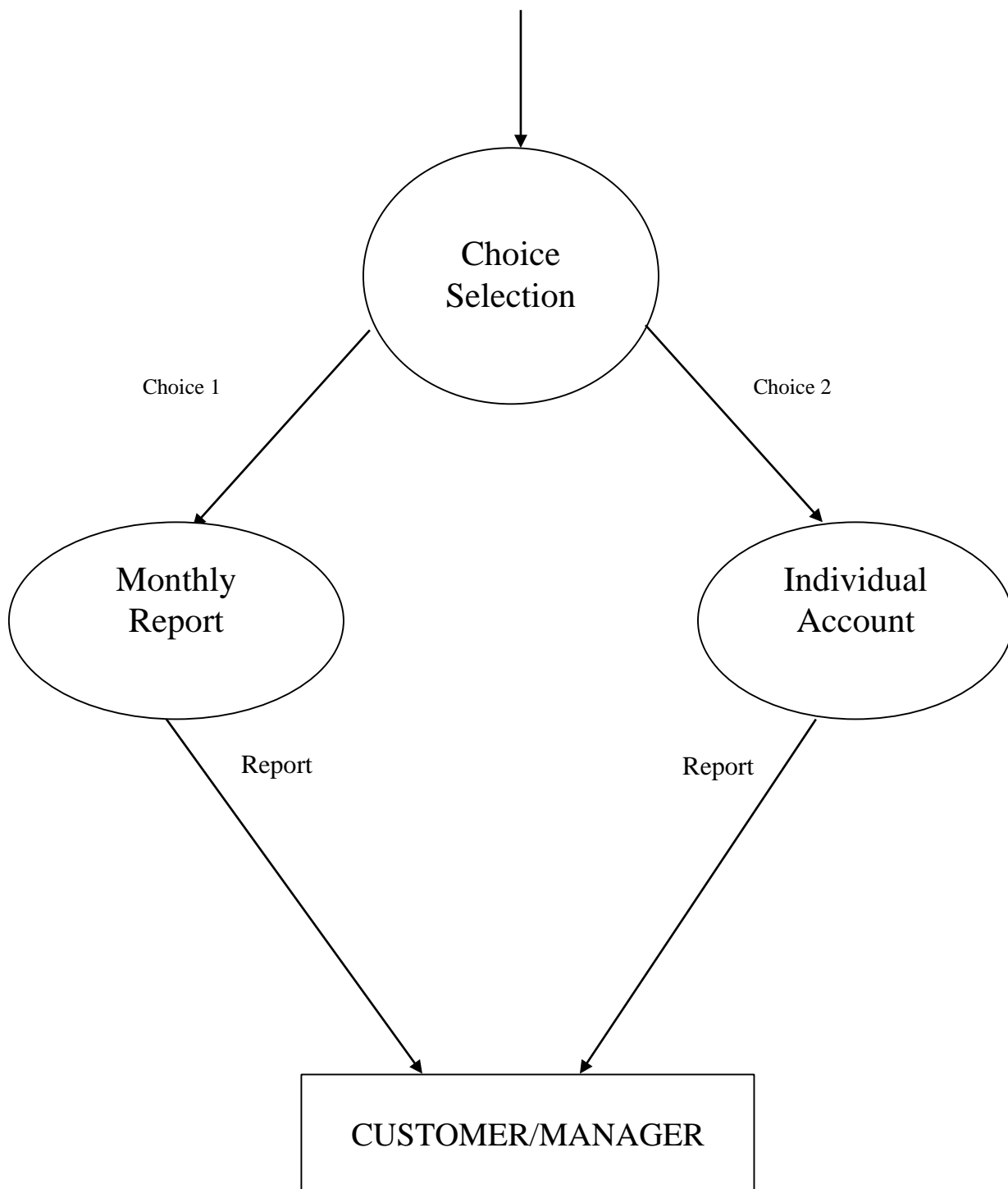
### 0 level DFD





FILE MAINTENANCE

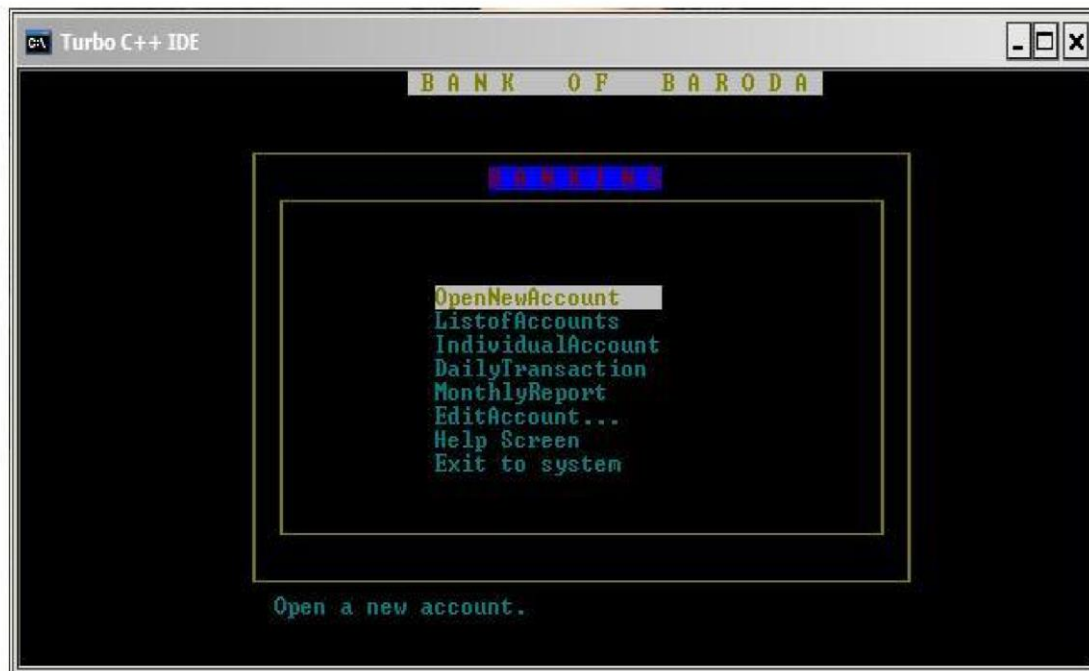
**LEVEL 1 DFD**



**LEVEL 2 DFD OF REPORT GENERATION**

## INPUT AND OUTPUT SCREEN

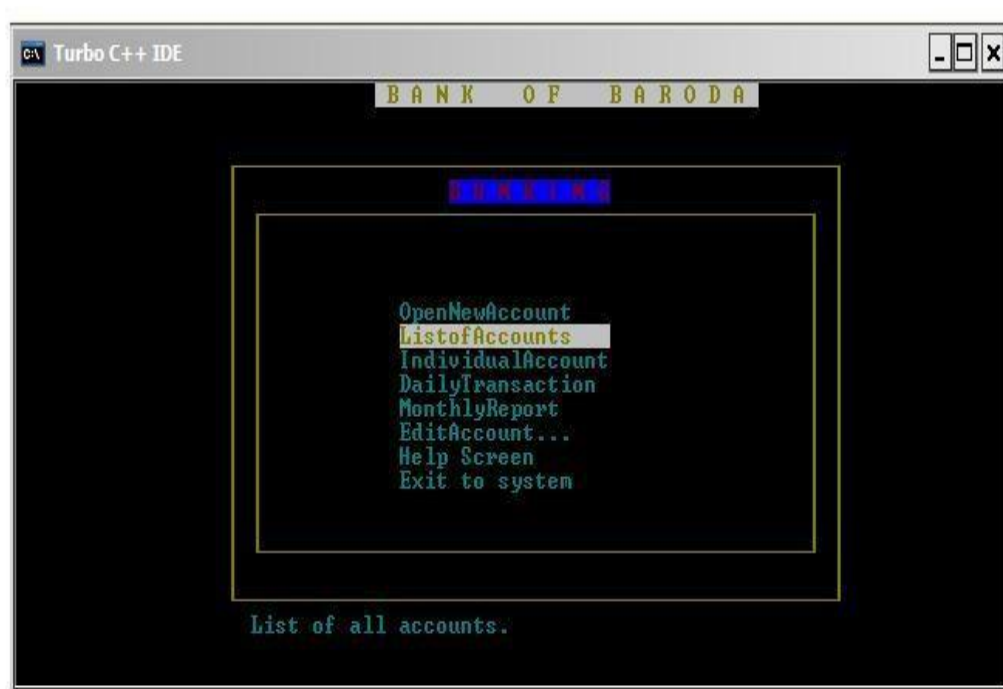
### Main menu for banking system:-



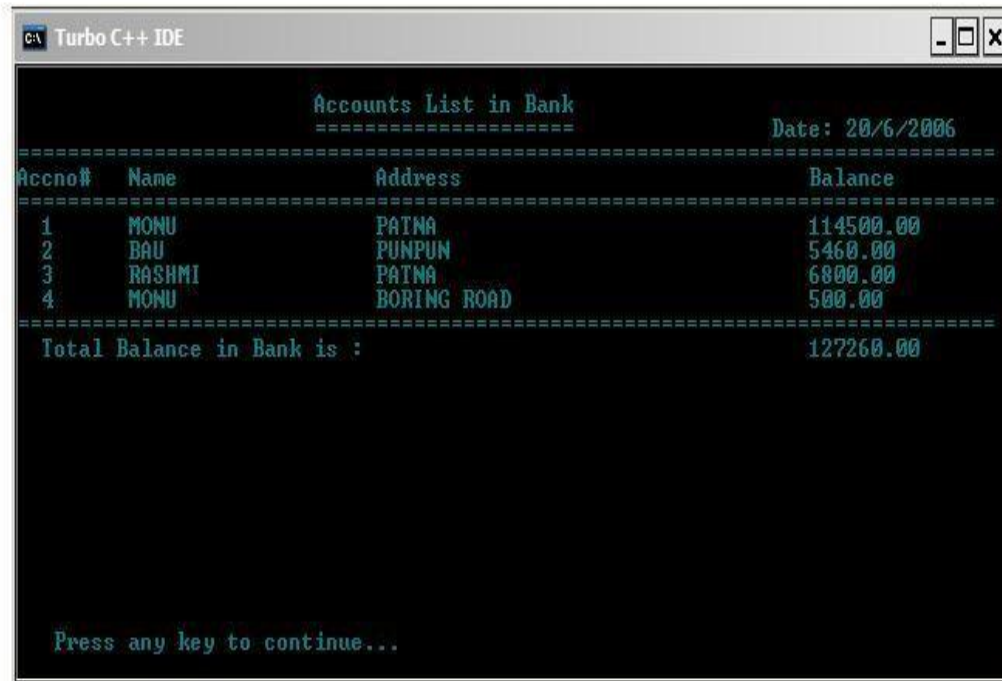
### Menu for Opening a new Account:-



## List Account:-



## Output of List Accounts in the bank:-



The screenshot shows a Turbo C++ IDE window with a black background and green text. The program output displays a table of bank accounts. The title 'Accounts List in Bank' is centered at the top, followed by a date 'Date: 20/6/2006'. The table has four columns: 'Accno#', 'Name', 'Address', and 'Balance'. It lists four accounts with their respective details. At the bottom, it shows the 'Total Balance in Bank is : 127260.00'. A prompt 'Press any key to continue...' is at the very bottom.

```
Accounts List in Bank
=====
Date: 20/6/2006
=====
Accno#  Name      Address      Balance
=====
1       MONU      PATNA        114500.00
2       BAU       PUNPUN       5460.00
3       RASHMI    PATNA        6800.00
4       MONU      BORING ROAD   500.00
=====
Total Balance in Bank is :      127260.00

Press any key to continue...
```

### Individual Account:-



### Output of Individual Account:-



Turbo C++ IDE

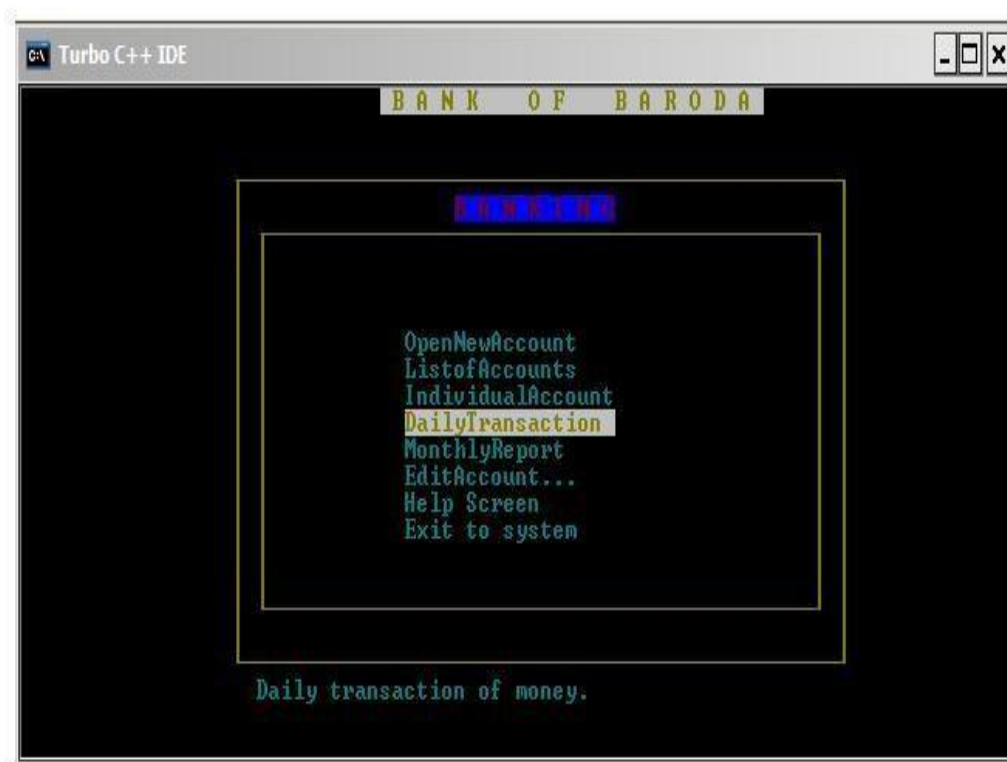
Account No. 1      MONU  
                         PATNA      Date: 20/6/2006

Global Report of Account

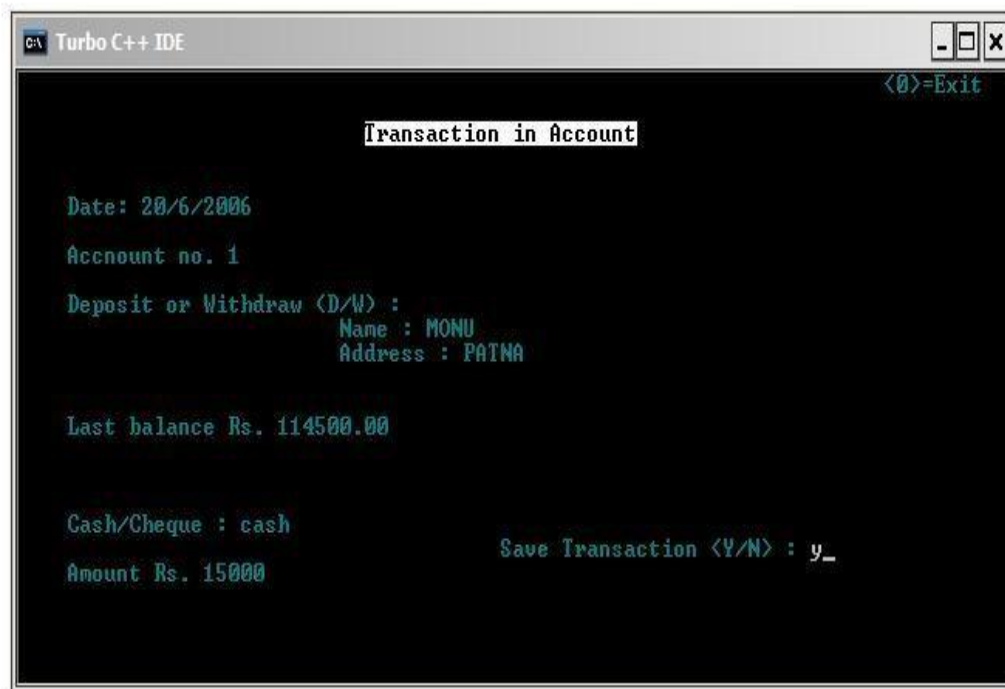
Date	Particular	Deposit	Withdraw	Balance
17-6-2006	INITIAL	1000.00		1000.00
17-6-2006	CASH		1000.00	0.00
17-6-2006	CHEQUE	5000.00		5000.00
17-6-2006	CASH		500.00	4500.00
19-6-2006	CASH	10000.00		14500.00
19-6-2006	CASH	100000.00		114500.00
Total-->:		-15072	1500	-16572

Press any key to continue...\_

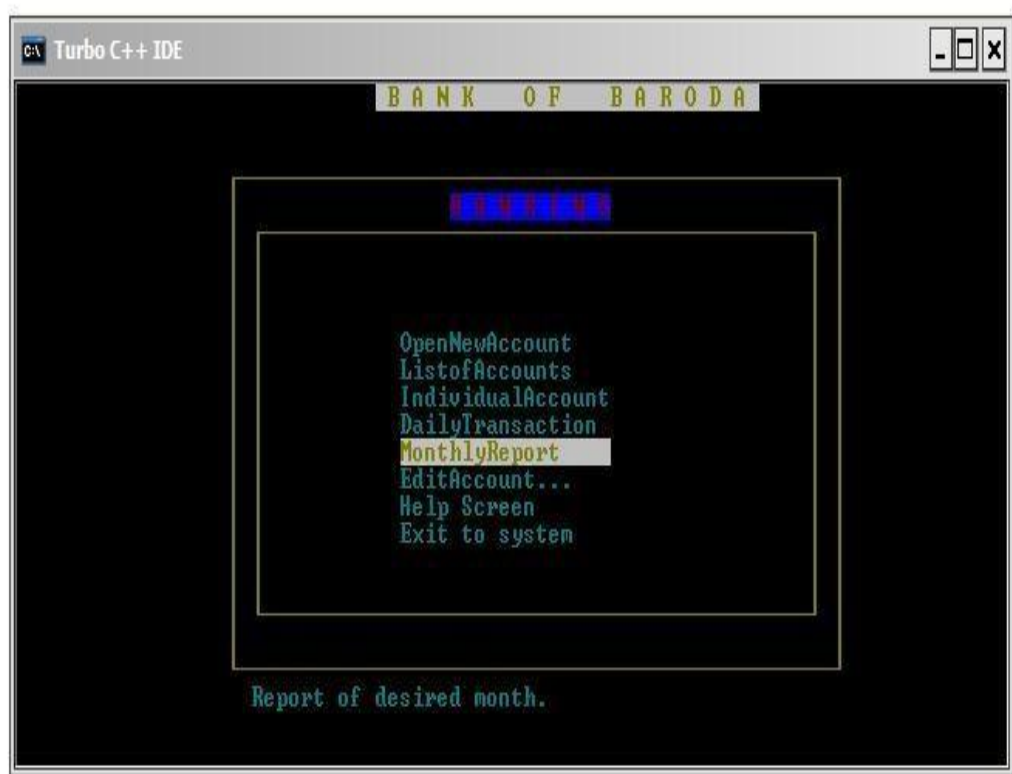
### Daily Transaction:-



### Output of Daily Transaction:-



## Monthly report



Output of Monthly report :-

Turbo C++ IDE

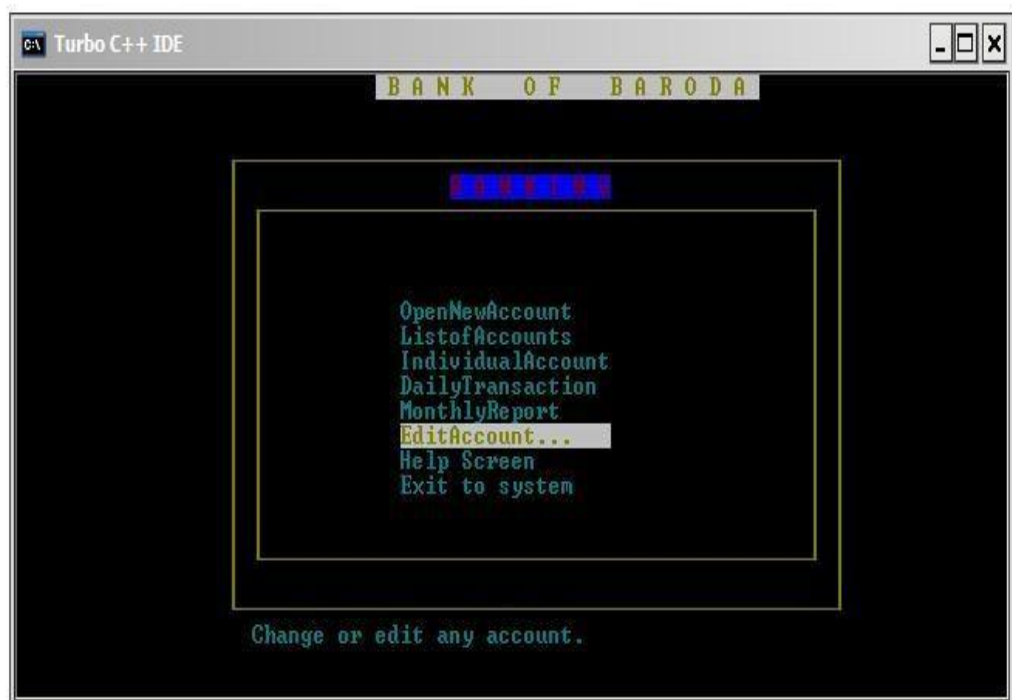
Account No. 2      BAU  
PUMPUN      Date: 20/6/2006

Statement Month: 20/6/6t

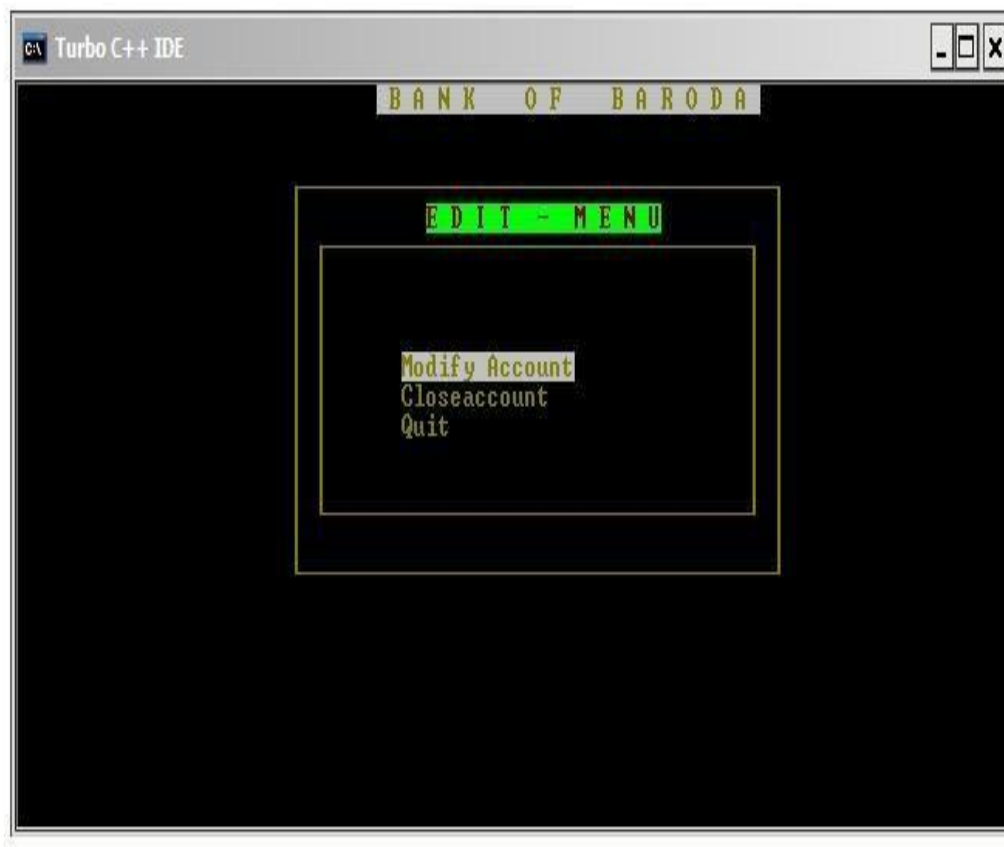
Date	Particular	Deposit	Withdraw	Balance
				B/F .... 0.00
17-6-2006	INITIAL	1000.00		1000.00
18-6-2006	CASH		1000.00	0.00
18-6-2006	CASH	5460.00		5460.00
Total-->		6460	1000	5460

Press any key to continue...\_

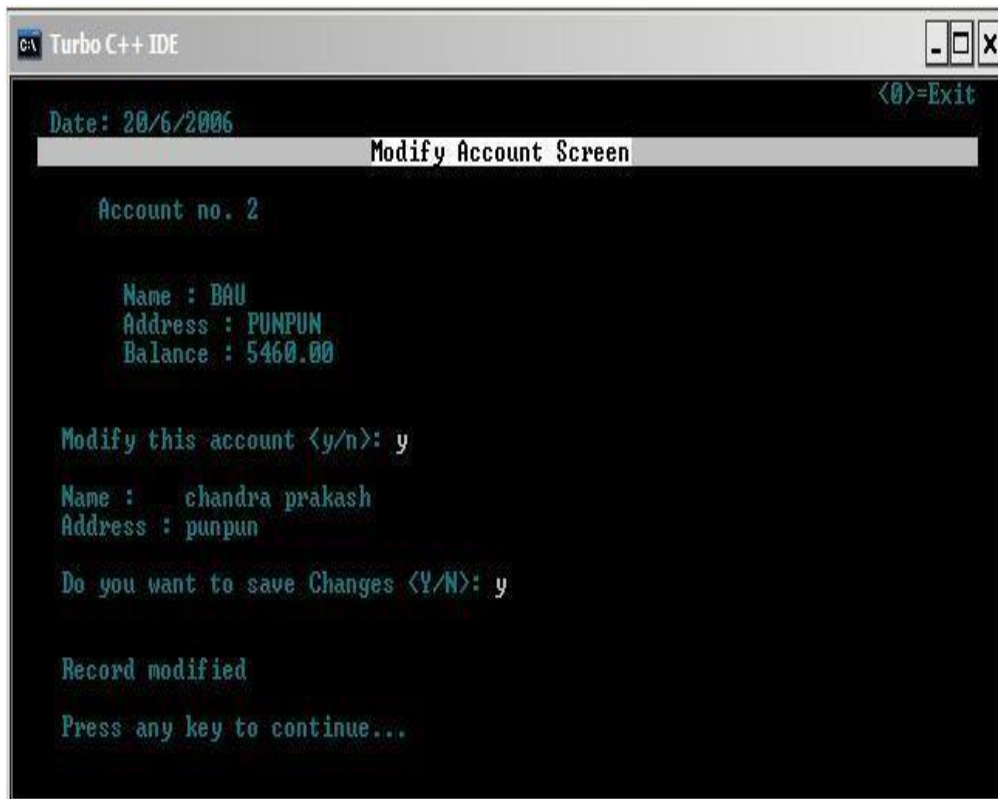
## Edit account



Edit menu start to modify account



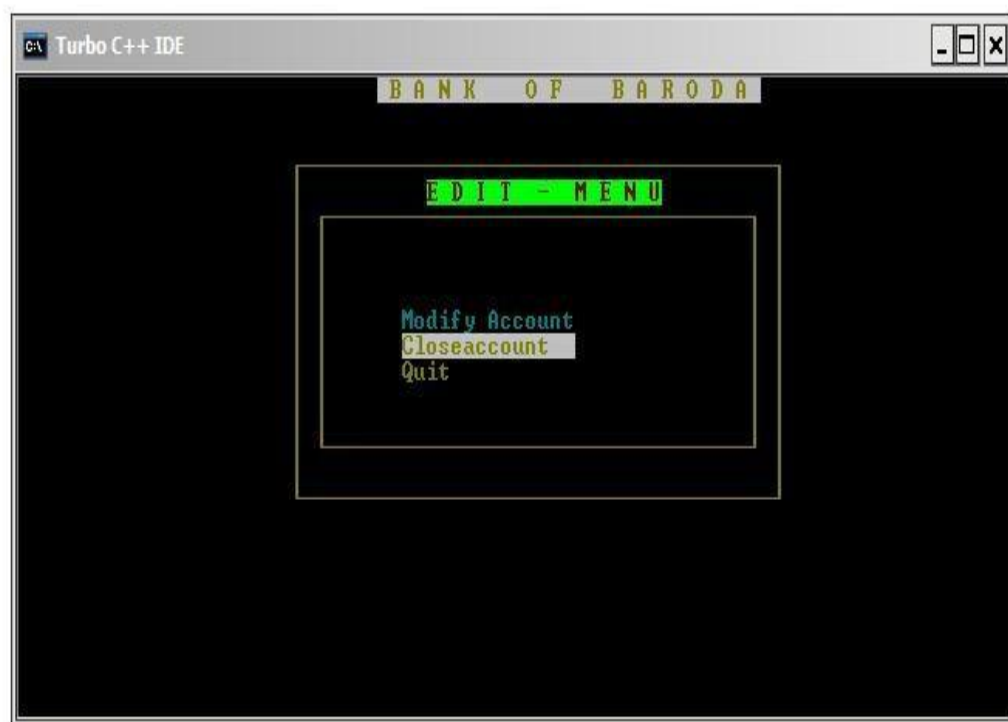
### Output of Modify account Screen



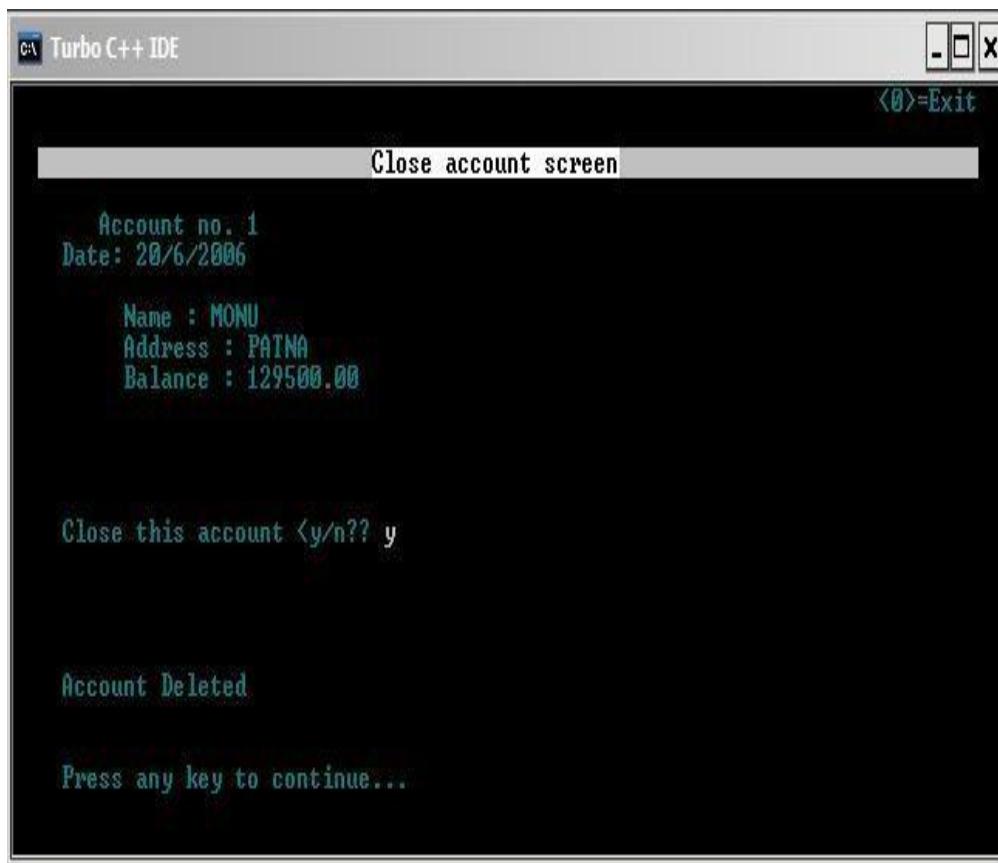
```
C:\ Turbo C++ IDE
Date: 20/6/2006
Modify Account Screen
Account no. 2
Name : BAU
Address : PUNPUN
Balance : 5460.00
Modify this account <y/n>: y
Name : chandra prakash
Address : punpun
Do you want to save Changes <Y/N>: y
Record modified
Press any key to continue...
```



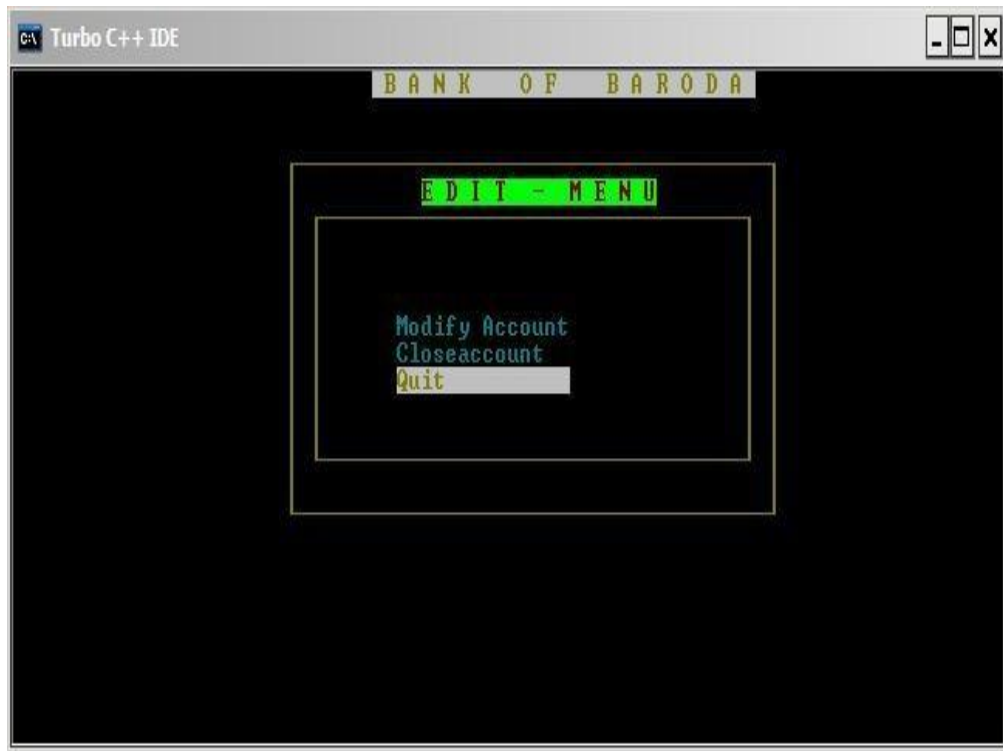
## Close account



### **Output of Close account screen**

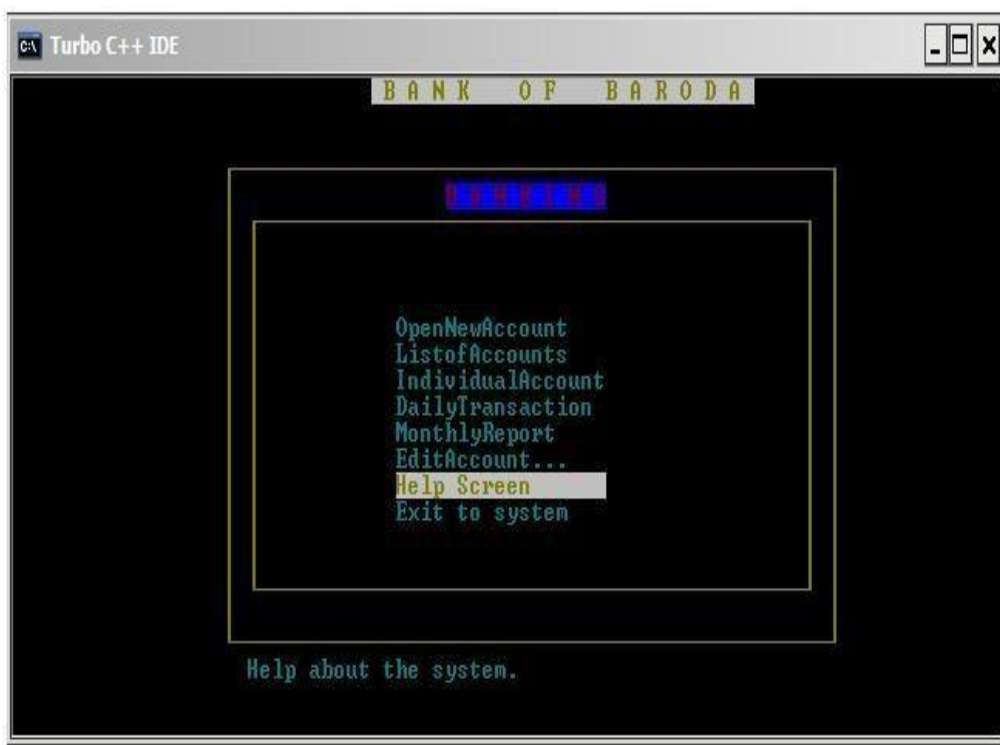


## Quit system





## Help Screen



## Exit to system



### **Scope For Future Application of Project:-**

The application /project “banking transaction system” for saving account has various types of future application:-

The proposed software has been made to automate the internal bank transaction of saving account. It can be further the confined to use for various other financial work of the bank and other financial institution.

This project provides the flexibility so that it will be portable to any other similar organization.

This system will work as base for development of more advanced future application of organization.

In the future this project is a base to networking of all the branches of rural post office of this region through its head post office, S.K.Puri, ,Patna.



## CODING

### BANKING AUTOMATION SYSTEM

```
/******<<<<Project Work on Banking System>>>>******/
```

```
// Declaration of header files
```

```
#include <iostream.h>
```

```
#include <fstream.h> #include  
<process.h>
```

```
#include <string.h>
```

```
#include <stdio.h>
```

```
#include <ctype.h>
```

```
#include <conio.h>
```

```
#include <dos.h>
```

```
#include <stdlib.h>
```

---

```
#include <iomanip.h>
```

```
#include <graphics.h>
```

```
typedef char option[20]; typedef  
char message[50];
```

```
const int ROW = 10, COL = 10;
```

```
int scan; // To hold the special characters for moving the prompt in menu  
int  
ascii;
```

```
// To display the main menu options
```

```
option a[] = {
```

```
    "OpenNewAccount  ",
```

```
    "ListofAccounts  ",
```

```
    "IndividualAccount",
```

```
    "DailyTransaction ",
```

```
"MonthlyReport  ",
```

```
    "EditAccount...  ",
```

```
    "Help Screen    ",
```

```
    "Exit to system  "
```

```
};
```

```
//messages for options
```

```
message m[]={
```

```
    "Open a new account.      ",
```

```
    "List of all accounts.    ",
```

```
    "Information of individual account.",
```

```
    "Daily transaction of money.  ",
```

```
    "Report of desired month.    ",
```

```
    "Change or edit any account.  ",
```

```
        "Help about the system.    ",
        "Exit..                    "
    };
```

```
// Displays the modify menu options
```

```
option b[] = {
    "Modify Account",
    "Closeaccount ",
    "Quit          "
};
```

```
// Function used to do screening class
```

```
main_menu
{
    int i,done;

    public:
        void normalvideo(int x,int y,char *str);
        void reversevideo(int x,int y,char *str);
        void box(int x1,int y1,int x2,int y2);
        char menu();
        void control_menu();
        char e_menu();        void
        edit_menu();
        void help(void);
};
```

---

---

```
/* Class member functions for drawing boxes */
```

```
class shape
```

```
{  
    public:  
        void line_hor(int, int, int, char);  
void line_ver(int, int, int, char);  
        void box(int, int, int, int, char);  
};
```

```
// Class contains the initial deposit of customers class
```

```
initial  
{  
    void modify_account(int, char t_name[30], char t_address[30]); // Function to  
modify the customer account  
    int accno;  
char name[30];        char  
address[30];  
    float balance;  
  
    public:  
        void add_to_file(int, char t_name[30],  
char t_address[30], float); // For initial deposits in customers account  
void display_list(void); // Displaying customers account list        void  
delete_account(int); // Deleting customers account  
        void update_balance(int, char t_name[30], char t_address[30], float); // For  
updating the customer account  
        void modify(void); // To modify the customer account information  
int last_accno(void); // To know the last account number  
        int found_account(int); // To found the account is in "INITIAL.dat" or not
```

```

        char *return_name(int); // Function for validation entry of customer name
        char *return_address(int); // Function for validation entry of customer address
        float give_balance(int); // To print the balance amount of a particular customer      int
        recordno(int);

        void display(int); // To display the customer account

};

// Class contains the customers daily transaction entry
class account
{
    void add_to_file(int, int, int, int, char, char t_type[10], float, float, float);
    // Function to add transaction records
    void delete_account(int); // Function to delete a transaction record
    int no_of_days(int, int, int, int, int, int); // Function to find the total days
    float calculate_interest(int, float); // Function for calculating interest of an
    account

    void display(int); // Function to display a transaction account
    void box_for_display(int); // Function for displaying box      int
    accno;

    char type[10];    // Account type as Cheque or Cash
    int dd, mm, yy; // To store the system date/ Enter date

    char tran; // As the account type is Deposit or Withdraw    float interest;    float
    amount;

    float balance;

public:
    void new_account(void); // Function to create a new account
    void close_account(void); // Function to close an account      void
    display_account(void); // Function to display the accounts      void

```

```
transaction(void); // To display the transaction process        void
clear(int, int); // Function to perform a clear screen function

        void month_report(void); // Function to list monthWise transaction report

};

// Function to displays all the menu prompt messages from the pointer array of option a[]

void main_menu::normalvideo(int x,int y,char *str)
{
    gotoxy(x,y);
    cprintf("%s",str);
}

// Function to move the cursor on the menu prompt with a reverse video color
void main_menu::reversevideo(int x,int y,char *str)
{
    textcolor(5+143);
    textbackground(WHITE);
    gotoxy(x,y);    cprintf("%s",str);
    textcolor(GREEN);
    textbackground(BLACK);
}

void main_menu::box(int x1,int y1,int x2,int y2)
{
    for(int col=x1;col<x2;col++)
    {
        gotoxy(col,y1);
        cprintf("%c",196);        gotoxy(col,y2);
        cprintf("%c",196);
    }
}
```

```

    for(int row=y1;row<y2;row++)
    {
        gotoxy(x1,row);
        cprintf("%c",179);          gotoxy(x2,row);
        cprintf("%c",179);
    }
    gotoxy(x1,y1);
    cprintf("%c",218);          gotoxy(x1,y2);
    cprintf("%c",192);          gotoxy(x2,y1);
    cprintf("%c",191);          gotoxy(x2,y2);
    cprintf("%c",217);
}

```

```

char main_menu::menu()
{ clrscr();
  textcolo
  r(22);
  box(20,
  6, 65,
  20);
  box(18,
  4, 67,
  22);
  gotoxy(30,1); textbackground(WHITE);
  textcolor(22);
  cprintf(" B A N K   O F   B A R O D A ");
  textcolor(5+143);
  gotoxy(36, 5);
  textbackground(BLUE);
  cprintf("B A N K I N G");
  textbackground(WHITE);

```

```
    textcolor(20);
    for(i = 1; i < 8; i++)
normalvideo(32, i+10, a[i]);
reversevideo(32, 10, a[0]);
normalvideo(20,23,m[0]);    i =
done = 0;
    _setcursortype(_NOCURSOR);
do
{
    int key = getch();
    switch (key)
    {
        case 00:
            key = getch();
            switch (key)
            {
                case 72:
normalvideo(32, i+10, a[i]);

                    i--;
                    if (i == -1)
                        i = 7;
                    reversevideo(32,i+10,a[i]);
normalvideo(20,23,m[i]);
                    break;
                case 80:
                    normalvideo(32, i+10, a[i]);
                    i++;
                    if (i == 8)
                        i = 0;
                    reversevideo(32, i+10, a[i]);
normalvideo(20,23,m[i]);
                    break;
```



```
        }
        break;

    case 13:
done = 1;
        }
    }
    while (!done);
    _setcursortype(_NOCURSOR);
return(i+49);
}
```

// The function main\_menu() is used to display the main menu of banking system.

```
void main_menu::control_menu()
{

    char choice;
    account a;

    do
    {
        choice = menu();
        clrscr();
switch (choice)
    {
        case '1':
            _setcursortype(_NORMALCURSOR);
            box(3, 1, 75, 24);
            box(5, 2,
73, 23);
```

```
        a.new_account(); // New account member function
        break; case
'2':
        box(3, 1, 75, 24);
        box(5, 2, 73, 23);
        initial ini;
        ini.display_list(); // Glogal list of account function
        break;
        case '3':
            box(3, 1, 75, 24);
box(5, 2, 73, 23);

        _setcursortype(_NORMALCURSOR);
        a.display_account(); // Displaying individual accounts all
        transactions
        break;
case '4':

        box(3, 1, 75, 24);
box(5, 2, 73, 23);

        account a;
        _setcursortype(_NORMALCURSOR);
        a.transaction(); // Daily transaction for individual account
        break;
case '5':

        box(3, 1, 75, 24);
box(5, 2, 73, 23);

        _setcursortype(_NORMALCURSOR);
        a.month_report(); // Monthly report for any account
```

```
                break;
            case '6':
                box(3, 1, 75, 24);
            box(5, 2, 73, 23);
                gotoxy(10,10);
edit_menu(); // Sub menu for modifying or deleting any account
                break;
            case '7' :
                clrscr();
                box(3, 1, 75, 24);
            box(5, 2, 73, 23);
                help();
                break; case '8'
                :exit(0);
        }
    } while (choice != 7);
}
```

/\* This function is used to return the cursor position to the edit menu function where the menu prompt will valid \*/

```
char main_menu::e_menu()
{
    clrscr();

    textcolor(22);
    box(25,6,60,15);  box(23,4,62,17);
        gotoxy(30,1);
    textbackground(WHITE);
    textcolor(22);
```

```
cprintf(" B A N K   O F   B A R O D A ");

textcolor(5+143);

gotoxy(34,5);
textbackground(GREEN);
cprintf("E D I T - M E N U");
textcolor(22);
textbackground(BLACK);
for (i = 1;i < 3; i++)
    normalvideo(32, i+10, b[i]);

reversevideo(32, 10, b[0]);
i = done = 0;

_setcursortype(_NOCURSOR);
do
{
    int key = getch();
    switch (key)
    {
        case 00:
            key = getch();
            switch (key)
            {
                case 72:
                    normalvideo(32, i+10, b[i]);
                    i--;
                    if (i == -1)
                        i = 2;
```

```

                                reversevideo(32, i+10, b[i]);
                                break;
                        case 80:
                                normalvideo(32, i+10, b[i]);
                                i++;
                                if (i == 3)
                                        i=0;
                                reversevideo(32, i+10, b[i]);
                                break;
                }
                break;
        case 13:
                done = 1;
                }
        }
        while (!done);
        _setcursortype(_NOCURSOR);
return(i+49);
}

```

/\* Function for edit menu with account modification and close

This is the one of the submenu which manages two basic operations as:

- Editing any account
- Deleting any account \*/

```

void main_menu::edit_menu()
{

```

```

        char        choice;
        account a; do

```

```
{
    choice = e_menu();
    clrscr();
    switch
(choice)
    {
        case '1':
            box(3, 1, 75, 24);
            box(5, 2, 73, 23);
initial ini;
            _setcursortype(_NORMALCURSOR);
            ini.modify();
            break;

        case '2':
            box(3, 1, 75, 24);
            box(5, 2, 73, 23);
account a;
            _setcursortype(_NORMALCURSOR);
            a.close_account();
            break;
        case '3':
            return;
    }
} while (choice != 6);
}
```

/\* Function to draw horizontal line

This public function draws one horizontal line at a time \*/

```
void shape::line_hor(int column1, int column2, int row, char c)
{
    for (column1; column1 <= column2; column1++)
    {
        gotoxy(column1, row);
        cout << c;
    }
}
```

/\* Function to draw vertical line

This public function draws one vertical line at a time \*/

```
void shape::line_ver(int row1, int row2, int column, char c)
{
    for (row1; row1 <= row2; row1++)
    {
        gotoxy(column, row1);
        cout << c;
    }
}
```

/\* Function for drawing box

This function draws a box for menus \*/

```
void shape::box(int column1, int row1, int column2, int row2, char c)
{
    char ch = 218;    char
c1, c2, c3, c4;    char l1 =
196, l2 = 179;    if (c == ch)
    {
```

```
        c1 = 218;
        c2 = 191;
c3 = 217;        c4 =
217;        l1 = 196;
        l2 = 179;
    }
    else
    {
        c1 = c;
        c2 = c;        c3
= c;    c4 = c;
        l1 = c;        c2
= c;
    }
    gotoxy(column1, row1);
    cout << c1;
    gotoxy(column2, row1);
    cout << c2;
    gotoxy(column1, row2);
    cout << c3; gotoxy(column2,
row2);    cout << c4;
    column1++;
        column2--;
        line_hor(column1, column2, row1, l1); //Horizontal line
    line_hor(column1, column2, row2, l1);
        column1--;
    column2++;    row1++;
        row2--;
        line_ver(row1, row2, column1, l2); // Vertical line
    line_ver(row1, row2, column2, l2);
}
```



---

```
/* Function to display help about this project
```

```
This help function is the first screen output display to know about the menu options and about the
```

```
banking system project */
```

```
void main_menu::help(void)
```

```
{
```

```
    clrscr();
```

```
    int gdriver = DETECT, gmode, errorcode,i,j;
```

```
    initgraph(&gdriver, &gmode, ""); //Initialize graphics mode
```

```
    setbkcolor(15);
```

```
    setcolor(7);
```

```
    rectangle(10,10,632,418); rectangle(10,10,632,98);
```

```
    rectangle(55,98,590,418);
```

```
    rectangle(11,11,631,417);
```

```
    rectangle(11,11,631,97);
```

```
    rectangle(56,99,589,417);
```

```
    rectangle(12,12,630,416);
```

```
    rectangle(12,12,630,96);
```

```
    rectangle(57,100,588,416);
```

```
    setcolor(5);
```

```
    settextstyle(7,VERT_DIR,4);
```

```
    outtextxy(18,130," BANKING ");
```

```
    outtextxy(590,130," SYSTEM ");
```

```
    settextstyle(7,HORIZ_DIR,5);    setcolor(22);
```

```
    outtextxy(25,20," BANK OF BARODA");
```

```
    settextstyle(2,HORIZ_DIR,5);    setcolor(50);
```

```
    outtextxy(60,100, "This project can keep record of daily banking transaction");
```

```
    delay(2);
```

```
        outtextxy(60,130, "This program is capable of holding any no. of account.");
delay(2);
        outtextxy(60,160, "-In first option you can open new account");
delay(2);
        outtextxy(60,190, "-In second option you can see the list of all the accounts");
delay(2);
        outtextxy(60,220, "-In third option you can see all the transaction of individual account");
        delay(2);
        outtextxy(60,250, "-Through fourth option you can do banking transactions");
delay(2);
        outtextxy(60,280, "(Deposit/Withdraw)");
        delay(2);
        outtextxy(60,310, "-In fifth option you can take monthWise individual account report");
        delay(2);
        outtextxy(60,340, "-In sixth option you can modify or delete any account");
delay(2);
        outtextxy(60,370, "Note-: Opening amount should not less than Rs. 500/-"); delay(2);
        outtextxy(60,400, "-And last option is Quit (Exit to Window)"); delay(2);
        settextstyle(2,0,4);
        //outtextxy(250,460, "[PASSWORD : ANC]");
        settextstyle(7,HORIZ_DIR,4);
        setcolor(20);
        outtextxy(80,420, "Press any key to continue. . ."); getch();
closegraph();
}

/* Function for modifying the existing accounts */

void initial::modify(void)
{
```

```
        clrscr();    int j; char t_acc[10];
int t, t_accno;    gotoxy(17, 1);
cout << "<0>=Exit";    gotoxy(5,5);
cout << "Enter the account no. ";

    gets(t_acc);
t = atoi(t_acc);
t_accno = t;

    if (t_accno == 0)
        return;
    clrscr();
    if (!found_account(t_accno))
    {
        gotoxy(5, 5);    cout <<
"\7Account not found";    getch();
        return;
    }
    gotoxy(71, 1);    cout <<
"<0>=Exit";
textbackground(WHITE);
gotoxy(3, 3);
    for (j = 1; j<= 76; j++)
        cprintf(" ");
    textbackground(BLACK);
    textcolor(BLACK+BLINK);
    textbackground(WHITE); gotoxy(30,
3);
    cprintf("Modify Account Screen");
    textcolor(LIGHTGRAY);
    textbackground(BLACK);
int d1, m1, y1;
    struct date d; // For extracting system date
```

```
        getdate(&d);
d1 = d.da_day;    m1 =
d.da_mon;        y1 =
d.da_year;

        gotoxy(4, 2);
        cout << "Date: " << d1 << "/" << m1 << "/" << y1;
        char ch;
display(t_accno);
        account a;
        do
        {
                a.clear(5, 13);
                gotoxy(5, 13);
                cout << "Modify this account <y/n>: ";
                ch = getche();
                if (ch == '0')
                        return;
                ch = toupper(ch);
        }while (ch != 'N' && ch != 'Y');
        if (ch == 'N')
return;

        int modified = 0, valid;
char t_name[30], t_address[30];
        gotoxy(5, 15);  cout <<
"Name: "; gotoxy(5, 16);
        cout << "Address : ";

        do
        {
                a.clear(15, 15);
```

```
        a.clear(5, 23);
        gotoxy(5, 23);
        cout << "Enter Name or Press Enter for No Change";
        valid = 1;
        gotoxy(15, 15);
        gets(t_name);         strupr(t_name);
        if (t_name[0] == '0')
            return;
        if (strlen(t_name) > 25)
        {
            valid = 0;
            gotoxy(5, 23);
            printf("\7Name should not greater than 25");
            getch();
        }
    } while (!valid);

do
{
    a.clear(15, 16);
    a.clear(5, 23);
    gotoxy(5, 23);
    cout << "Enter Address or press enter for no Change";
    valid = 1;
    gotoxy(15, 16);
    gets(t_address);
    strupr(t_address);        if
    (t_address[0] == '0')
        return;
    if (strlen(t_address) > 25)
```

```
{
    valid = 0;
    gotoxy(5, 23);
    cprintf("\7Address should not greater than 25");
    getch();
}
}while (!valid);
if (strlen(t_address) > 0)
    modified = 1;
if (!modified)
    return;
// clears the screen at 23rd row and from 5th column
a.clear(5,23);

do
{
    a.clear(5, 23);
    gotoxy(5, 18);
    cout << "Do you want to save Changes <Y/N>: ";
    ch = getche();
    if (ch == '0')
        return;
    ch = toupper(ch);
}while (ch != 'N' && ch != 'Y');

if (ch == 'N')
    return;

// Passes the parameter to add in data file
modify_account(t_accno, t_name, t_address);
```

```
    gotoxy(5, 21);
    cout << "\7Record modified";
gotoxy(5, 23);
    cout << "Press any key to continue...";
    getch();
}
```

/\* Function for displaying an account when modified

This display() function is used to display all the account holders account no, name, address, and

balance amount at screen. \*/

```
void initial::display(int t_accno)
```

```
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
```

```
// Displays the record contents matching with t_accno from INITIAL.dat data file while
(file.read((char *)this, sizeof(initial)))
```

```
{
    if (t_accno == accno)
    {
        gotoxy(8, 5);
        cout << "Account no. " << accno;
        gotoxy(10, 8);          cout <<
        "Name : ";              puts(name);
        gotoxy(10, 9);          cout << "Address : ";
        puts(address);
        gotoxy(10, 10);
        cout << "Balance : "
```

```

        << setw(15) // setwidth
        << setprecision(2) // set position of decimal point
        << setiosflags(ios::left) // set left justified output
    << setiosflags(ios::showpoint) // always show decimal point
        << setiosflags(ios::fixed) // set fixed notation for display
        << balance;

    break;
}
}
file.close();
}

```

/\* Function for updating the modified account into INITIAL.dat file

This function modify\_account() receives number parameters such as, account number, name, address etc, and

will overWrite at the existing place in the "INITIAL.dat" data file. \*/

```

void initial::modify_account(int t_accno, char t_name[30], char t_address[30])
{
    int recno;
    recno = recordno(t_accno); fstream
    file;    file.open("INITIAL.dat",
    ios::out|ios::ate);

    strcpy(name, t_name); strcpy(address,
    t_address);
    int location;
    // finds the position in data file

    location = (recno-1) * sizeof(initial);

```



```
        file.seekp(location);

        // Overwrites the modified record into INITIAL.dat data file
file.write((char *)this, sizeof(initial));
        file.close();
return;
}
```

/\* Function to find the last account number \*/

```
int initial::last_accno(void)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
file.seekg(0, ios::beg);
    int count = 0;

    // Finds the last account no.
    while (file.read((char *)this, sizeof(initial)))
        count = accno;
    file.close();
    return count;
}
```

/\* This function add\_to\_file() is used to create new/fresh record in the data file, and the parameters

is the actual value which will be write into the "INITIAL.dat" file. Every time when any account

makes transaction for either Withdraw (W) or Deposit (D) the balance amount will modified in the data file\*/

```
void initial::add_to_file(int t_accno, char t_name[30], char t_address[30], float t_balance) {  
    accno = t_accno;  
    strcpy(name, t_name);  
    strcpy(address, t_address);  
    balance = t_balance;  
    fstream file;  
  
    // Appends new account record with the balance into INITIAL.dat data file  
    file.open("INITIAL.dat", ios::out | ios::app);  
    file.write((char *)this, sizeof(initial));  
    file.close();  
}
```

```
// Function for deleting a account from INITIAL.dat file  
// This function is used to delete any account from data file. By this function void  
initial::delete_account(int t_accno)
```

```
{  
    fstream file;  
    file.open("INITIAL.dat", ios::in);  
    fstream temp;  
    temp.open("TEMP.dat", ios::out);  
    file.seekg(0, ios::beg);  
  
    // Uses a copy method to delete the account from INITIAL.dat data file  
    while (!file.eof())  
    {  
        file.read((char *)this, sizeof(initial));  
        if (file.eof())  
            break;
```

```
        if (accno != t_accno)
            temp.write((char *)this, sizeof(initial));
    }
    file.close();
    temp.close();
    file.open("INITIAL.dat", ios::out);
    temp.open("TEMP.dat", ios::in);
    temp.seekg(0, ios::beg);

    // Copy the TEMP.dat contents into INTITAL.dat data file
    while (!temp.eof())
    {
        temp.read((char *)this, sizeof(initial));
        if (temp.eof())
            break;
        if (accno != t_accno)
            file.write((char *)this, sizeof(initial));
    }
    file.close();
    temp.close();
}
```

/\* Function for add an account details of daily tranaction into BANKING.dat file. By using the parameters the information of daily transaction appended into "BANKING.dat"

data file including amount transaction and the current balance value. \*/

```
void account::add_to_file(int t_accno, int d1, int m1, int y1, char t_tran, char t_type[10], float t_interest, float t_amount, float t_balance)
```

```
{
    fstream file;
```

```
file.open("BANKING.dat", ios::app);  
accno = t_accno;  
getch(); dd = d1;  
mm = m1; yy = y1;  
tran = t_tran;  
strcpy(type, t_type);  
interest = t_interest;  
amount = t_amount;  
balance = t_balance;
```

```
// Appends the transaction record into BANKING.dat data file  
file.write((char *)this, sizeof(account));  
file.close();  
}
```

/\* Function for deleting an account from BANKING.dat file. This is a copy method like, when the user will input any account no, the same account no. will checked in the "BANKING.dat" file, if the account no. matched in your data file then, it will remain in your data file otherwise other remaining records will transferred into "TEMP.dat" data file. So, your new "TEMP.dat" file holds all the records execept inputed account no. On the same way make copy of all the "TEMP.dat" records back into your "BANKING.dat" file for restoring that your data file deletes the matched account no. \*/

```
void account::delete_account(int t_accno)  
{  
    fstream file;  
    file.open("BANKING.dat", ios::in); // Open to read records  
    fstream temp;  
    temp.open("TEMP.dat", ios::out); // Open to write records  
    file.seekg(0, ios::beg); // Positioned from begining of the file
```

```
// Uses the copy method for deleting the transaction record from BANKING.dat data file
while (!file.eof())
{
    file.read((char *)this, sizeof(account));
    if (file.eof())
break;
    if (accno != t_accno)
        temp.write((char *)this, sizeof(account));
}
file.close();
temp.close();
file.open("BANKING.dat", ios::out);
temp.open("TEMP.dat", ios::in);    temp.seekg(0,
ios::beg);

// Uses copy method to transfer the record from TEMP.dat file to BANKING.dat data
file while (!temp.eof())
{
    temp.read((char *)this, sizeof(account));
    if (temp.eof())
        break;
    if (accno != t_accno)
        file.write((char *)this, sizeof(account));
}
file.close();
temp.close();
}
```

/\* Function for displaying an account from "INITIAL.dat". This function shows all account holders

data records and their respective current balance amount at screen. Thus, the bank will be a position to know

that there is a particular sum is available in bank. Thus, the total balance in bank can easily realised or status

of the bank can known in the management \*/ void

initial::display\_list(void)

```
{
    clrscr();    int
flag; float t_bal = 0.0;
fstream file;

    gotoxy(25,2);

    cout << "Accounts List in Bank";
    gotoxy(25, 3);
    cout << "=====";

int d1, m1, y1;

    struct date d;        // For extracting system date
getdate(&d);    d1 = d.da_day;    m1 = d.da_mon; y1 =
d.da_year;

    gotoxy(62, 3);
cout << "Date: " << d1 << "/" << m1 << "/" << y1; gotoxy(1, 4);
    for (int j = 1; j <= 79; j++)
        cout << "=";

    gotoxy(1, 5); cout
<<    "Accno#";
    gotoxy(10,5); cout
<< "Name";
```

```
        gotoxy(30,5);
cout << "Address";
gotoxy(65,5);    cout <<
"Balance"; gotoxy(1, 6);
        for (j = 1; j <= 79; j++)
            cout << "=";

        file.open("INITIAL.dat", ios::in);
file.seekg(0,ios::beg);
        int row = 7;

// Reads all the records to display on the screen
        while (file.read((char *)this, sizeof(initial)))
        {
            flag = 0;
delay(2);        gotoxy(3,
row);          cout <<
accno;         gotoxy(10,
row);
puts(name);
gotoxy(30, row);
puts(address);
            gotoxy(65, row);

            cout << setw(15)
<< setprecision(2)
                << setiosflags(ios::left)
                << setiosflags(ios::showpoint)
                << setiosflags(ios::fixed)
                << balance;
            t_bal = t_bal + balance;
            row++;
        }
```

```
        if (row > 23)
        {
            flag = 1;
            row = 6;
            gotoxy(4, 24);
            cout << "Press any key to continue.... ";
            getch();
            clrscr();
        }
    }
```

```
gotoxy(1, row);
for (j = 1; j <= 79; j++)
    cout << "=";
row++;
gotoxy(3, row);
cout << "Total Balance in Bank is : ";
gotoxy(65, row);
```

```
cout << setw(15)
    << setprecision(2)
    << setiosflags(ios::left)
    << setiosflags(ios::showpoint)
    << setiosflags(ios::fixed)
    << t_bal;
```

```
file.close();
if (!flag)
{
```



```
        gotoxy(4, 24);
        cout << "Press any key to continue...";
        getch();
    }
}
```

/\* Function for clearing specified row and column. By using this function you can clear from a specified row and column

from your screen. Here the function utilizes two parameter for clearing the row and column.  
\*/

```
void account::clear(int col, int row)
{
    for (int j = col; j <= 79; j++)
    {
        gotoxy(j, row);
        cout << " ";
    }
}
```

/\* Function to found an account for display account function. This function is used to found any account

in the "INITIAL.dat" data file, where the file is searched from the beginning position and search

whether the entered account exist or not. If exist then the found variable will return a value 1 or return 0 as the parameter \*/

```
int initial::found_account(int t_accno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
```

```
int found = 0;

// Searches the specified record in INITIAL.dat data file
while (file.read((char *)this, sizeof(initial)))
{
    if (accno == t_accno)
    {
        found = 1;
        break;
    }
}
file.close();
return found;
}
```

/\* Function for return name of the account holder from INITIAL.dat. This function basically used to return only name of the account holder if the account in the "INITIAL.dat" data file. When the name will returned it may simply display / modify at other functions \*/

```
char *initial::return_name(int t_accno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg); char
    t_name[30];

    // Return the name to display at report screen if found
    while (file.read((char *)this, sizeof(initial)))
    {
```

```
        if (accno == t_accno)
        {
            strcpy(t_name, name);
            break;
        }
    }
    file.close();
    return t_name;
}
```

/\* Function for return address of the account holder from INITIAL.dat. This function basically used to return only address of the account holder if the account in the "INITIAL.dat" data file. When the address will returned it may simply display / modify at other functions \*/

```
char *initial::return_address(int t_accno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    char t_address[30];

    // Return the address to display at report screen if found
    while (file.read((char *)this, sizeof(initial)))
    {
        if (accno == t_accno)
        {
            strcpy(t_address, address);
            break;
        }
    }
}
```

```
    }  
    file.close();  
    return t_address;  
}
```

/\* Function for display account details as: This function displays the heading and the account no. name, address and current date

on the screen with other function like display\_account(), month\_report() etc.. Also the same function returns the account holders

name and address by using two pointer type function like return\_name() and return\_address(). \*/

```
void account::box_for_display(int t_accno)  
{  
    int d1, m1, y1;  
    struct date d;  
    getdate(&d);    d1 =  
    d.da_day; m1 =  
    d.da_mon;    y1 =  
    d.da_year;  
    gotoxy(63, 2);  
    cout << "Date: " << d1 << "/" << m1 << "/" << y1; gotoxy(4, 2);  
    cout << "Account No. " << t_accno;  
  
    initial ini;  
  
    char t_name[30];  
    strcpy(t_name, ini.return_name(t_accno));  
  
    char t_address[30];  
    strcpy(t_address, ini.return_address(t_accno));
```

```

        gotoxy(25, 2);
cout << t_name;
gotoxy(25, 3);    cout
<< t_address;
        gotoxy(4, 5);
        cout << "Global Report of Account";
        textbackground(WHITE); textcolor(BLACK);
        textbackground(WHITE); gotoxy(1, 6);
        for (int i = 1; i <=79; i++)
            cout << "=";
        gotoxy(4, 7);
        cprintf("Date    Particular  Deposit    Withdraw        Balance");
        gotoxy(1, 8);
        for (i = 1; i <=79; i++)
            cout << "=";
        textcolor(LIGHTGRAY);
        textbackground(BLACK);
    }

```

/\* Function for display an account from BANKING.dat file. This is a function who displays all the transaction of any account on screen. And the function show the account no. name, and address

through the function box\_for\_display(). Also through this function it shows the total deposit, total withdraw

and current balance amount globally. \*/

```
void account::display_account(void)
```

```

{
    clrscr();
    char t_acc[10];    int
j;

```

```
    int tamt = 0, damt = 0, wamt = 0;
int t, t_accno;    gotoxy(71, 1);
cout << "<0>=Exit";    gotoxy(5, 5);
    cout << "Enter account no. ";
    gets(t_acc);
t = atoi(t_acc);
t_accno = t;    if
(t_accno == 0)
return; clrscr();
    initial ini;
    if (!ini.found_account(t_accno))
    {
        gotoxy(5, 5);    cout <<
"\7Account not found";
        getch();
        return;
    }

    // Display the heading from this function
box_for_display(t_accno);

    int row = 9, flag;
fstream file;
    file.open("BANKING.dat", ios::in);
    while (file.read((char *)this, sizeof(account)))
    {
        if (accno == t_accno)
        {
            flag = 0;
            delay(2);
            gotoxy(4, row);
```

```
        cout << dd << "-" << mm << "-" << yy;
    gotoxy(16, row);
puts(type);

    if (tran == 'D')
    {
        damt = damt + amount;
        tamt = tamt + amount;
        gotoxy(30, row);
    }
    else
    {
        wamt = wamt + amount;
        tamt = tamt - amount;
        gotoxy(42, row);
    }

    cout << setw(15)
    << setprecision(2)
        << setiosflags(ios::left)
        << setiosflags(ios::showpoint)
setiosflags(ios::fixed)
        << amount;

    gotoxy(66, row);
    cout << setw(15)
        << setprecision(2)
        << setiosflags(ios::left)
        << setiosflags(ios::showpoint)
        << setiosflags(ios::fixed)
        << balance;
```

```
        row++;
        if (row > 23)
        {
            flag = 1;
            row = 7;
            gotoxy(4, 24);
            cout << "Press any key to continue";
            getch();
            clrscr();
            box_for_display(t_accno);
        }
    }
    file.close();
    gotoxy(1, row);
    for (j = 1; j <= 79; j++)
        cout << "=";
    row++;
    gotoxy(4, row);
    cout << "Total-->:";
    gotoxy(30, row);
    cout << setw(15)
        << setprecision(2)
        << setiosflags(ios::left)
        << setiosflags(ios::showpoint)
        << setiosflags(ios::fixed)
        << damt;
    gotoxy(42, row);
    cout << setw(15)
```



```
        << setprecision(2)
        << setiosflags(ios::left)
        << setiosflags(ios::showpoint)
        << setiosflags(ios::fixed)
        << wamt;
gotoxy(66, row);
cout << setw(15)
        << setprecision(2)
        << setiosflags(ios::left)
        << setiosflags(ios::showpoint)
        << setiosflags(ios::fixed)
        << tamt;
if (!flag)
{
    gotoxy(4, 24);
    cout << "Press any key to continue...";
    getch();
}
}

/* Function to list monthWise transaction report.*/
void account::month_report(void)
{

    int dd1, mm1, yy1;
    clrscr();    fflush(stdin);
    gotoxy(10, 5);
    cout << "Enter any date of a month ";
```

```
        gotoxy(38, 5);
cin >> dd1;
gotoxy(40, 5); cout <<
"-"; gotoxy(41, 5); cin
>> mm1; gotoxy(43, 5);
cout << "-"; gotoxy(44,
5); cin >> yy1; clrscr();
char t_acc[10];

    int j;

    int tamt = 0, damt = 0, wamt = 0;
int t, t_accno;    gotoxy(71, 1);
cout << "<0>=Exit";    gotoxy(5, 5);

    cout << "Enter account no. ";

    gets(t_acc);
t = atoi(t_acc);
t_accno = t;

    if (t_accno == 0)
        return;

    clrscr();
initial ini;

    if (!ini.found_account(t_accno))
    {
        gotoxy(5, 5);
        cout << "\7Account not found";
getch();

        return;
    }

    box_for_display(t_accno);
gotoxy(4, 5);

    cout << "Statement Month: " << dd1 << "/" << mm1 << "/" << yy1;
getch();
```

```
int row = 9, flag;
fstream file;

file.open("BANKING.dat", ios::in);
float pre_balance = 0.0; // Previous balance amount

// The loop finds the last months balance while
(file.read((char *)this, sizeof(account)))
{
    // Checks the account no. and till the previous month and till current year
    if ((accno == t_accno) && ((mm < mm1 && yy <= yy1) || (mm1 < mm && yy < yy1)))
    {
        pre_balance = balance;
    }
}

file.close();
file.open("BANKING.dat", ios::in);
gotoxy(54, row);

cout << "B/F .... " << setw(15)
      << setprecision(2)
      << setiosflags(ios::left)
      << setiosflags(ios::showpoint)
      << setiosflags(ios::fixed)
      << pre_balance;

row++;

// The loop displays the current months transaction after previous month
while (file.read((char *)this, sizeof(account)))
```

```
{
    if ((accno == t_accno) && (mm1 == mm && yy1 <= yy))
    {
        flag = 0;
    delay(2);
        gotoxy(4, row);
        cout << dd << "-" << mm << "-" << yy;
        gotoxy(16, row);
    puts(type);
        if (tran == 'D')
        {
            damt = damt + amount;
            tamt = tamt + amount;
            gotoxy(30, row);
        }
        else
        {
            wamt = wamt + amount;
            tamt = tamt - amount;
        gotoxy(42, row);
        }

        cout << setw(15)
    << setprecision(2)
            << setiosflags(ios::left)
            << setiosflags(ios::showpoint)
            << setiosflags(ios::fixed)
            << amount;
```

```
        gotoxy(66, row);
    cout << setw(15)
<< setprecision(2)
        << setiosflags(ios::left)
        << setiosflags(ios::showpoint)
        << setiosflags(ios::fixed)
        << balance;

    row++;

// If row increases 23 then the next screen continues
    if (row > 23)
    {
        flag = 1;
        row = 7;
        gotoxy(4, 24);
        cout << "Press any key to continue";
        getch();
        clrscr();
        box_for_display(t_accno);
    }
}

file.close();
gotoxy(1, row);
for (j = 1; j <= 79; j++)
    cout << "=";

row++; gotoxy(4,
row);
```

```
cout << "Total-->:"; gotoxy(30,
row);

// Deposited amount cout <<
setw(15) // setwidth
    << setprecision(2) // set position of decimal point
    << setiosflags(ios::left) // set left justified output
<< setiosflags(ios::showpoint) // always show decimal point
    << setiosflags(ios::fixed) // set fixed notation for display
    << damt;

gotoxy(42, row); //
Withdraw amount cout <<
setw(15)
    << setprecision(2)
    << setiosflags(ios::left)
    << setiosflags(ios::showpoint)
    << setiosflags(ios::fixed)
    << wamt;

gotoxy(66, row);
tamt = tamt + pre_balance; //
Balance amount cout <<
setw(15)
    << setprecision(2)
    << setiosflags(ios::left)
    << setiosflags(ios::showpoint)
    << setiosflags(ios::fixed)
    << tamt;
```

```
        if (!flag)
        {
            gotoxy(4, 24);
            cout << "Press any key to continue...";
            getch();
        }

    }

/* Function for creating new account for new customer. */ void
account::new_account(void)
{
    char ch;
    int i, valid;
    clrscr();
    initial ini;

    shape s;
    s.box(2, 1, 79, 25, 218);
    s.box(25, 2, 54, 4, 219);

    gotoxy(65, 2);
    cout << "<0>=Exit";
    gotoxy(3,3);
    for (i = 1; i<= 76; i++)
        cprintf(" ");
    textbackground(BLACK);
    textcolor(BLACK+BLINK);
    textbackground(WHITE);
    gotoxy(30, 3);
```

```
cprintf(" Open New Account ");
textcolor(LIGHTGRAY);    textbackground(BLACK);
int d1, m1, y1;
struct date d;           // For extracting system date
getdate(&d);    d1 = d.da_day;   m1 = d.da_mon; y1 =
d.da_year; int t_accno;
t_accno = ini.last_accno();
t_accno++;

// Appends and deletes a false record to create primary position in data files if
(t_accno == 1)
{
    ini.add_to_file(t_accno, "abc", "xyz", 1.1);
    ini.delete_account(t_accno); add_to_file(t_accno, 1, 1, 1997, 'D',
"INITIAL", 1.1, 1.1, 1.1);
    delete_account(t_accno);
}
char t_name[30], t[10], t_address[30];
float t_bal = 0.0, t_balance = 0.0;
gotoxy(5, 6);
cout << "Date: " << d1 << '/' << m1 << '/' << y1; gotoxy(5, 8);
cout << "Account No # " << t_accno;
gotoxy(5, 10);
cout << "Name : ";
gotoxy(5, 11);    cout <<
"Address : ";
gotoxy(5, 12);
cout << "Name of verifying Person : ";
gotoxy(5, 14);
cout << "Initial Deposit : ";
```



```
do
{
    clear(15, 10);
clear(5, 23);
gotoxy(5, 23);
    cout << "Enter Name of the Person";
    valid = 1;        gotoxy(15, 10);
gets(t_name);       strupr(t_name);        if
(t_name[0] == '0')
    return;
    if (strlen(t_name) == 0 || strlen(t_name) > 25)
    {
        valid = 0;
        gotoxy(5, 23);
        cprintf("\7Name should not greater than 25");
        getch();
    }
}while (!valid);
do
{
    clear(25, 15);
    clear(5, 23);
    gotoxy(5, 23);
    cout << "Enter Address of the Person ";
    valid = 1;        gotoxy(15, 11);
gets(t_address);   strupr(t_address);
if (t_address[0] == '0')
    return;
    if (strlen(t_address) == 0 || strlen(t_address) > 25)
    {
        valid = 0;
```

```
        gotoxy(5, 23);
    cprintf("\7Address should not greater than 25");
        getch();
    }
}while (!valid);

do
{
    char vari[30];
clear(32, 12);        clear(5,
23);

    gotoxy(5, 23);
    cout << "Enter name of the varifying Person ";
    valid = 1;
gotoxy(32, 12);
gets(vari);       strupr(vari);
    if (vari[0] == '0')
        return;
    if (strlen(vari) == 0 || strlen(vari) > 25)
    {
        valid = 0;
        gotoxy(5, 23);
        cprintf("Should not blank or greater than 25");
        getch();
    }
}while (!valid);

do
{
```

```
        clear(23, 14);
clear(5, 23);
gotoxy(5, 23);
        textcolor(GREEN);
        cout << "Enter initial amount to be deposit ";
        valid = 1;
gotoxy(23, 14);        gets(t);
        t_bal = atof(t);
t_balance = t_bal;        if
(t[0] == '0')
    { if (strlen(t[0]) <=500)
        {
            valid = 0;
            clear (5, 23);
gotoxy(5, 23);
                textcolor(RED+BLINK);
                cprintf("\7Should not less than 500");
                getch();
            }
        }
    }while (!valid);
clear(5, 23);
do
{
    clear(5, 17);
valid = 1;
    gotoxy(5, 17);
    cout << "Do you want to save the record <Y/N>: ";
    ch = getche();
    if (ch == '0')
```

```
        return;

        ch = toupper(ch);
    }while (ch != 'N' && ch != 'Y');

    if (ch == 'N')
        return;

    float t_amount, t_interest;
    t_amount = t_balance;
    t_interest = 0.0;  char t_tran,
    t_type[10];
    t_tran = 'D';
    strcpy(t_type, "INITIAL");

    // Appends the records contents into both INITIAL.dat and BANKING.dat data files
    ini.add_to_file(t_accno, t_name, t_address, t_balance);
    add_to_file(t_accno, d1, m1, y1, t_tran, t_type, t_interest, t_amount, t_balance);
}

/* Function for returning balance amount of an account. This function returns the balance
amount of any account
to know the current balance from "INITIAL.dat" data file. */

float initial::give_balance(int t_accno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    float t_balance;
```

```
// Gives the last balance of an individual account
while (file.read((char *)this, sizeof(initial)))
{
    if (accno == t_accno)
    {
        t_balance = balance;
        break;
    }
}
file.close();
return t_balance;
}

/* Function for returning the record no. for updating balance
This function check the position of the account number for updating new balance amount
into either "INITIAL.dat" or "BANKING.dat" data files. */ int initial::recordno(int t_accno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);  int count = 0;

    // Finds the record position in INITIAL.dat data file
    while (file.read((char *)this, sizeof(initial)))
    {
        count++;
        if (t_accno == accno)
            break;
    }
    file.close();
    return count;
}
```

---

```
/* Function for updating the balance for the given account no.*/
```

```
void initial::update_balance(int t_accno, char t_name[30], char t_address[30], float t_balance)
{
    int recno;
    recno = recordno(t_accno); fstream file;
    file.open("INITIAL.dat", ios::out|ios::ate);
    strcpy(name, t_name);
    strcpy(address, t_address); balance =
    t_balance;
    int location;
    location = (recno-1) * sizeof(initial); // Find the location in file file.seekp(location);
    // Searches the insertion position in data file

    // Updates the balance amount in INITIAL.dat data file
    file.write((char *)this, sizeof(initial));
    file.close();
}
```

```
/* Function to return no. days between two dates.
```

The function calculates total number of days between two date. And the function passes parameters as related to date (day, month, year). \*/

```
int account::no_of_days(int d1, int m1, int y1, int d2, int m2, int y2)
{
    static int month[] = {31, 28, 31, 30, 31, 30, 31, 30, 31, 30, 31, 30};
    int days = 0;
    while (d1 != d2 || m1 != m2 || y1 != y2)
    {
        days++;
        d1++;
```

```
        if (d1 > month[m1-1])
        {
            d1 = 1;
m1++;
        }
        if (m1 > m2)
        {
            m1 = 1;
y1++;
        }
    }
    return days;
}
```

/\* Function for calculates interest

This function calculate the interest of any account accouding to the account no. and balance from "BANKING.data" data file

Before calculate the interest, the funcation also finds total number of days and then find the interest. \*/

```
float account::calculate_interest(int t_accno, float t_balance)
```

```
{
    fstream file;
    file.open("BANKING.dat", ios::in);
    file.seekg(0, ios::beg); int
    d1, m1, y1, days;
    while (file.read((char *)this, sizeof(account)))
    {
        if (accno == t_accno)
        {
```

```

        d1 = dd;
        m1 = mm;
        y1 = yy;
        break;
    }
}

int d2, m2, y2;
struct date d;
getdate(&d);    d2 =
d.da_day;  m2 =
d.da_mon;
y2 = d.da_year;
float t_interest = 0.0;
if ((y2 < y1) || (y2 == y1 && m2 < m1) || (y2 == y1 && m2 == m1) && (d2 < d1))
return t_interest;
days = no_of_days(d1, m1, y1, d2, m2, y2);
int months = 0;
if (days > 30)
{
    months = days / 30;
    t_interest = ((t_balance*2)/100 * months);
}
file.close();
return t_interest;
}

```

/\* Function for making daily transaction (Deposit 'D'/Withdraw 'W'. \*/ void  
account::transaction(void)

```

{ clrscr(); char t_acc[10];
  int t, t_accno, valid;
  gotoxy(71,1);

```



```

        cout << "<0>=Exit"; gotoxy(5,
5);
        cout << "Enter the account no. ";
        gets(t_acc); t = atoi(t_acc);
t_accno = t; if
(t_accno == 0)
        return;
        clrscr();
initial ini;
        if (!ini.found_account(t_accno))
        {
                gotoxy(5, 5);          cout <<
"\7Account not found";          getch();
                return;
        }
        gotoxy(71, 1);
cout << "<0>=Exit";
        gotoxy(3, 3);
        for (int i = 1; i <= 76; i++)
                cprintf(" ");
textbackground(BLACK);
textcolor(BLACK+BLINK);
textbackground(WHITE);    gotoxy(29, 3);
cprintf ("Transaction in Account");
textcolor(LIGHTGRAY);
textbackground(BLACK);    int d1, m1, y1;
struct date d;    getdate(&d);    d1 = d.da_day;
m1 = d.da_mon; y1 = d.da_year; gotoxy(5, 6); cout
<< "Date: " << d1 << "/" << m1 << "/" << y1; gotoxy(5,
8);

        cout << "Accnount no. " << t_accno;

```

```
char t_name[30]; char
t_address[30]; float
t_balance;

strcpy(t_name, ini.return_name(t_accno)); strcpy(t_address,
ini.return_address(t_accno));
t_balance = ini.give_balance(t_accno);
gotoxy(27, 11);  cout <<
"Name : " << t_name;  gotoxy(27,
12);

cout << "Address : " << t_address;

gotoxy(5, 15);
cout << "Last balance Rs. " << setw(15)
    << setprecision(2)
    << setiosflags(ios::left)
<< setiosflags(ios::showpoint)
    << setiosflags(ios::fixed)
    << t_balance;

char t_tran, t_type[10], tm[10];
float t_amount, t_amt;

do
{
    clear(5, 10);
    valid = 1;
    gotoxy(5, 10);
    cout << "Deposit or Withdraw (D/W) : ";
    t_tran = getch();
    if (t_tran == '0')
```

```
        return;

        t_tran = toupper(t_tran);
    }while (t_tran != 'D' && t_tran != 'W');

do
{
    clear(5, 19);
    clear(5, 23);
    gotoxy(5, 23);
    cout << "Enter
Transaction by Cash or
Cheque ";
    valid = 1; gotoxy(5, 19); cout <<
"Cash/Cheque : ";
    gets(t_type);
   strupr(t_type);
    if (t_type[0] == '0')
        return;
    if (strcmp(t_type, "CASH") && strcmp(t_type, "CHEQUE"))
    {
        valid = 0;
        gotoxy(5, 23);
        printf("\7Enter correctly");
        getch();
    }
}while (!valid);

do
{
```

```
        clear(5, 21);
clear(5, 23);
        gotoxy(5, 23);
        cout << "Enter Amount for Transaction ";
        valid = 1;        gotoxy(5, 21);
        cout << "Amount Rs. ";
gets(tm);
        t_amt = atof(tm);
t_amount = t_amt;
        if (tm[0] == '0')
                return;
if ((t_tran == 'W' && t_amount > t_balance) || (t_amount < 1))
{
        valid = 0;
        gotoxy(5, 23);
        cprintf("\7Invalid Data entered");
        getch();
}
}while (!valid);
char ch;
clear(5, 23); do
{
clear(20, 23);  valid = 1;
        gotoxy(40, 20);
        cout << "Save Transaction <Y/N> : ";
        ch = getche();
        if (ch == '0')
                return;
        ch = toupper(ch);
}while (ch != 'N' && ch != 'Y');
```

```
    if (ch == 'N')
        return;
    float t_interest;

    t_interest = calculate_interest(t_accno, t_balance);

    if (t_tran == 'D')
        t_balance = t_balance + t_amount + t_interest;
    else
        t_balance = (t_balance - t_amount) + t_interest;

    // Modified records are updated in data bases.
    ini.update_balance(t_accno, t_name, t_address, t_balance);
    add_to_file(t_accno, d1, m1, y1, t_tran, t_type, t_interest, t_amount, t_balance);
}

/* Function for closing any account after inputing account number. */ void
account::close_account(void)
{
    clrscr();
    char t_acc[10]; int
    t, t_accno;
    gotoxy(71, 1); cout
    << "<0>=Exit";
    gotoxy(5, 5);
    cout << "Enter the account no. ";
    gets(t_acc); t = atoi(t_acc);
    t_accno = t;
    if (t_accno == 0)
        return;
```

```
        clrscr();
initial ini;
        if (!ini.found_account(t_accno))
        {
                gotoxy(5, 5);          cout <<
"\7Account not found ";
                getch();
return;
        }
        gotoxy(71, 1);   cout <<
"<0>=Exit";          gotoxy(3, 3);
textbackground(WHITE);
        for (int i = 1; i <= 76; i++)
                cprintf(" ");
textbackground(BLACK);
textcolor(BLACK+BLINK);
textbackground(WHITE);   gotoxy(30, 3);
cprintf("Close account screen");
textcolor(LIGHTGRAY);
textbackground(BLACK);   int d1, m1, y1;
struct date d;   getdate(&d);   d1 = d.da_day;
m1 = d.da_mon; y1 = d.da_year; gotoxy(5, 6); cout
<< "Date: " << d1 << "/" << m1 << "/" << y1; char ch;
ini.display(t_accno); do
{
        clear(5, 15); gotoxy(5, 15);
                cout << "Close this account <y/n?? ";
                ch = getche();
                if (ch == '0')
                        return;
                ch = toupper(ch);
}while (ch != 'N' && ch != 'Y');
```

```
        if (ch == 'N')
return;

        // Function calls to delete the existing account no.
        ini.delete_account(t_accno);
delete_account(t_accno);
gotoxy(5, 20);    cout <<
"\7Account Deleted"; gotoxy(5,
23);

        cout << "Press any key to continue...";
getch();
}

// Graphics display WELCOME TO BANKING SYSTEM
void mainscreen()
{
    int d=DETECT,m=0,i,j=489,k;
    char *pwd; initgraph(&d,
&m,"");
    for(k=0;k<=450;k++)
    {
        setbkcolor(10);
setcolor(22);
        circle(320,240,k);
    }
cleardevice(); for(k=450;k>=0;k--)
{
```

```
    setbkcolor(30);
setcolor(20);
    circle(320,240,k);
}

cleardevice();
setbkcolor(15);
textcolor(5);
delay(1000);      int
ctr=0;
gotoxy(10,20);
textcolor(20);

    //printf("See Help for password");
    A:
    gotoxy(10,10);
fflush(stdin);
    textcolor(25);
    pwd=getpass("ENTER YOUR PASSWORD : ");
    if(strcmp(pwd,"monu")!=0)
    {
        settextstyle(3,0,4);
        outtextxy(250,250,"INVALID PASSWORD");
        delay(1000);    cleardevice();    ctr++;
        gotoxy(20,20);
        printf("TRY AGAIN : %d Chance left",3-ctr);
        if(ctr==3)      {
            cleardevice();
            outtextxy(100,100,"SORRY! YOU ARE NOT AUTHORISED");
            delay(1000);
            exit(0);
        }    goto A;
    }
```



```
        for(i=0;i<170;i++)
        {
            setbkcolor(20);
            setcolor(i+10);
            settextstyle(3,0,4);
            outtextxy(260,i,"WELCOME");
            setcolor(i+50);
            outtextxy(310,200,"TO");    setcolor(i+20);
            outtextxy(i+55,270,"BANKING");
            outtextxy(j,270,"SYSTEM");

            j--;
            delay(20);
            if(i==149)
            break;

            cleardevice();
        }
        setcolor(YELLOW);
        rectangle(202,138,451,352);
        setcolor(RED);
        rectangle(200,140,453,350);
        setcolor(BLUE);
        rectangle(198,142,455,348);
        settextstyle(3,0,2);
        outtextxy(250,400,"Press any key to continue...");
        getch();
        closegraph();
    }
```

// Main program logic which control the class members and member functions.

```
void main(void)
```

```
{  
    main_menu m_menu;  
    int gdriver = DETECT, gmode, errorcode,i,j;  
    initgraph(&gdriver, &gmode, ""); //Initialize graphics mode  
  
    m_menu.help();  
    mainscreen(); //display the main screen closegraph();  
    m_menu.control_menu();  
}
```

## CONCLUSION

This project application provides not all but some functions that are performed in a bank. the developed project is very efficient and reliable for the bank financing process like saving account. with the help of this project any banking organisation can automate its full banking process to save time and increase work efficiency. In order to make this project a success, the management of computerisation must be effective at branch and adequate trained man power must be provided.

This project applications so simple to work on that a one time instructions is enough for any person having some knowledge of computers. A variety of training modules should

be designed and specific training must be provided to senior management. All functions are performed on selecting of a specific choice/option assigned for a specific function.

I expect that, this project is to be our first step towards the software development profession.