**MINI PROJECT – BANK MANAGEMENT**

**INPUT**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

#include<iomanip>

using namespace std;

**static** **int** i = 0; *//keyword static used so that it cannot be modified again, it is allocated for a lifetime*

class account

{

public:

**int** acct\_no;

**char** name[40];

**int** size = 50;

string gender;

string acc\_type;

**int** balance;

*//default construtor for balance while opening of account*

account()

{

balance = 2000;

}

*//creting an account*

**void** create\_account()

{

cout << "Enter the account no :" << endl;

cin >> acct\_no;

cout << "Enter the account holders name :" << endl;

cin.ignore(); *//The cin. ignore() function is used which is used to ignore or clear one or more characters from the input buffer*

cin.getline(name, 50); *// For example, after entering into the cin statement, we need to input a character array or string.*

cout << "Enter the gender :" << endl; *// So we need to clear the input buffer, otherwise it will occupy the buffer of previous variable*

cin >> gender; *//The getline() is a standard library function that is used to read a string or a line*

cout << "\nEnter Type of The account (C/S) : " << endl; *//from an input stream. It is a part of the <string> header.*

cin >> acc\_type;

cout << endl;

}

};

*//personal data of the user*

class personal\_data :virtual public account

{

string ph\_no;

**char** address[100];

**int** size = 50;

string pan\_no;

**char** document[100];

public:

**void** get\_data()

{

cout << "Enter the account holders address:" << endl;

cin.ignore();

cin.getline(address, 50);

cout << "Enter the acount holders phone number :" << endl;

cin >> ph\_no;

cout << "Enter the account holders pan card no :" << endl;

cin >> pan\_no;

cout << "Enter the Document submitted by account holder :" << endl;

cin.ignore();

cin.getline(document, 50);

}

**void** display\_data()

{

cout << "\nThe account holders address is :" << address;

cout << "\nthe account holders phone number is:" << ph\_no;

cout << "\nThe documents are is :" << document << endl<<endl;

}

};

*//updating balance data*

class update\_data :virtual public account

{

public:

**void** deposit();

**void** withdraw();

**void** operator+(**int** x) { *//operator overloading*

balance = balance + x;

}

**void** operator-(**int** x) {

balance = balance - x; *//operator overloading*

}

};

*//application of loan*

class loan :public personal\_data, public update\_data

{

public:

**char** loan\_type[50];

**int** size = 60;

string mortagage;

**int** amount;

**void** get\_loan()

{

**if** (balance >= 5000)

{

cout << "\nyou can apply for the loan";

cout << "\nEnter the type of loan\nPersonal Loan:\nBuisness Loan:\nHome Loan:\nGold Loan:" << endl;

cin.ignore();

cin.getline(loan\_type,60);

cout << "\nEnter the loan amount" << endl;

cin >> amount;

}

**else**

{

cout << "Cannot apply for the loan" << endl;

exit;

}

**void** loan\_n(); *//for loan application*

}

**void** set\_loan()

{

cout << "Loan amount: " << amount;

cout << "\nTotal balance: " << balance;

cout << endl;

}

};

*//loan details*

class result : public loan

{

**double** roi, noy, t\_loan\_amt, m\_amt;

public:

**void** calculate()

{

cout << "Enter the interest rate: " << endl;

cin >> roi;

cout << endl;

cout << "The number of years: " << endl;

cin >> noy;

cout << endl;

t\_loan\_amt = (noy \* amount) + (noy \* amount \* (roi / 100.00));

}

**void** show\_account()

{

cout << "\nAccount No. : " << acct\_no;

cout << "\nAccount Holder Name : " << name;

cout << "\nType of Account : " << acc\_type;

cout << "\nBalance amount : " << balance;

cout << endl;

}

*//delete function*

**void** del() {

acct\_no = 0; *//when called will make the account no. 0*

cout<<" account has been closed "<<endl;

}

*//account details*

**int** retacno() {

**return** acct\_no;

}

**void** report()

{

cout << "Loan amount: " << amount << endl;

cout << "rate of interst: " << roi << endl;

cout << "Total amount with interst: " << t\_loan\_amt << endl;

}

~result() *//destructor*

{

}

}person[100]; *//globally decared variable*

*//function for creating account*

**void** write\_account()

{

person[i].create\_account();

person[i].get\_data();

i++;

}

*//function for depositing balance*

**void** deposit()

{

**int** x;

**int** actno;

cout << "Enter the account no" << endl;

cin >> actno;

**for** (**int** j = 0; j <= i; j++) {

**if** (actno == person[j].acct\_no) {

cout << "Enter the amount to be deposited" << endl;

cin >> x;

person[j] + x; *//used cause of operator overloading*

}

}

}

*//function to show balance details*

**void** show() {

**int** actno;

cout << "Enter the account no" << endl;

cin >> actno;

**for** (**int** j = 0; j <= i; j++) {

**if** (actno == person[j].acct\_no) {

person[j].show\_account();

}

}

}

*//function for withdrawing balance*

**void** withdraw() {

**int** x;

person[i];

**int** actno;

cout << "Enter the account no" << endl;

cin >> actno;

**for** (**int** j = 0; j <= i; j++) {

**if** (actno == person[j].acct\_no) {

cout << "Enter the amount to be withdrawn" << endl;

cin >> x;

person[j] - x; *//used cause of operator overloading*

}

}

}

*//function for modifying account and personal data*

**void** modify() {

**int** actno;

cout << "Enter the account no" << endl;

cin >> actno;

**for** (**int** j = 0; j <= i; j++) {

**if** (actno == person[j].acct\_no) {

cout << "Previous Data:" << endl;

person[j].show\_account();

person[j].display\_data();

cout << "updated data:" <<endl<<endl;

person[j].create\_account();

person[j].get\_data();

}

}

}

*//function for displaying account details*

**void** display\_acc\_detail()

{

**int** actno;

cout << "enter the account no" << endl;

cin >> actno;

**for** (**int** j = 0; j <= i; j++) {

**if** (actno == person[j].acct\_no)

{

cout << "account data" << endl;

person[j].show\_account();

cout << endl;

cout << "personal data" << endl;

person[j].display\_data();

cout << "loan data" << endl;

cout << endl;

person[j].report();

}

}

}

*//function for deleteing account*

**void** del\_acc() {

**int** actno;

cout << "Enter the account no:" << endl;

cin >> actno;

**for** (**int** j = 0; j <= i; j++) {

**if** (actno == person[j].acct\_no) {

person[j].del();

}

}

}

*//function for displaying details of all accounts*

**void** all\_acc() {

**for** (**int** j = 0; j < i; j++) {

**if** (person[j].acct\_no == 0) {

**continue**; *//when executed the loop will be skipped*

}

person[j].show\_account();

person[j].display\_data();

cout << endl;

}

}

*//function for loan application*

**void** loan\_n() {

**int** actno;

cout << "Enter the account no" << endl;

cin >> actno;

**for** (**int** j = 0; j <= i; j++) {

**if** (actno == person[j].acct\_no) {

person[j].get\_loan();

person[j].set\_loan();

person[j].calculate();

person[j].report();

}

}

}

**int** main()

{

**int** num;

**int** choice;

**while** (1)

{

cout.width(30);

cout << "\*\*\*\*\*\*\*BANK MANGEMENT\*\*\*\*\*\*\*"<<endl <<endl;

cout.width(22);

cout << "\*\*\*\*MAIN MENU\*\*\*\*" << endl <<endl;

cout << "1. CREATE NEW ACCOUNT" << endl;

cout << "2. DEPOSIT AMOUNT" << endl;

cout << "3. WITHDRAW AMOUNT" << endl;

cout << "4. BALANCE ENQUIRY" << endl;

cout << "5. ALL ACCOUNT HOLDER LIST" << endl;

cout << "6. CLOSE AN ACCOUNT" << endl;

cout << "7. MODIFY AN ACCOUNT" << endl;

cout << "8. DISPLAY ACCOUNT DETAIL" << endl;

cout << "9. LOAN APPLICATION" << endl;

cout << "10. EXIT " << endl;

cout << "Select Your Option (1-10) "<< endl;

cin >> choice;

**switch** (choice)

{

**case** 1:

write\_account();

**break**;

**case** 2:

deposit();

**break**;

**case** 3:

withdraw();

**break**;

**case** 4:

show();

**break**;

**case** 5:

all\_acc();

**break**;

**case** 6:

del\_acc();

**break**;

**case** 7:

modify();

**break**;

**case** 8:

display\_acc\_detail();

**break**;

**case** 9:

loan\_n();

**break**;

**case** 10:

exit(0);

**break**;

default:

cout << "\n\n\tThanks for using bank managemnt system";

**break**;

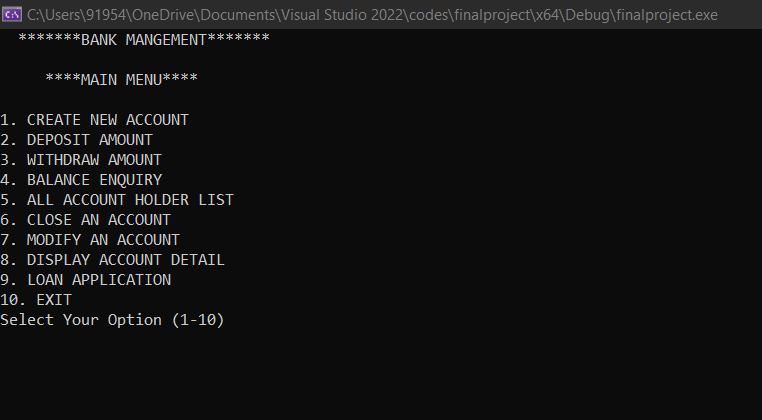
}

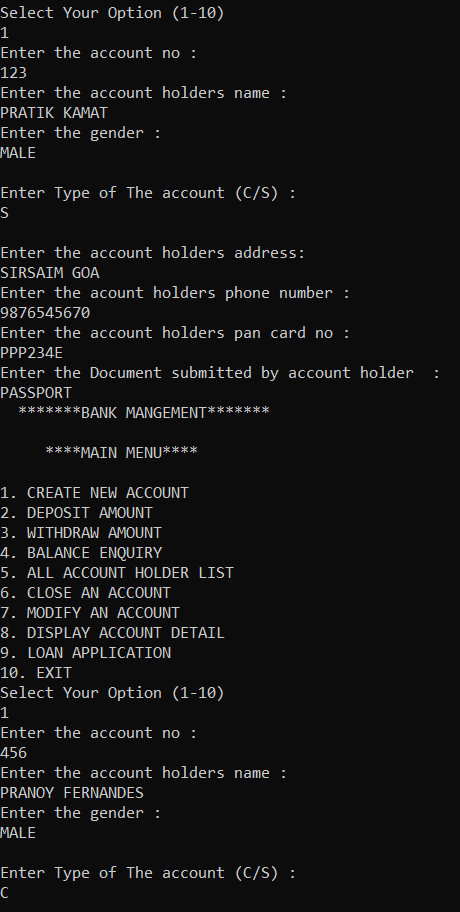
}

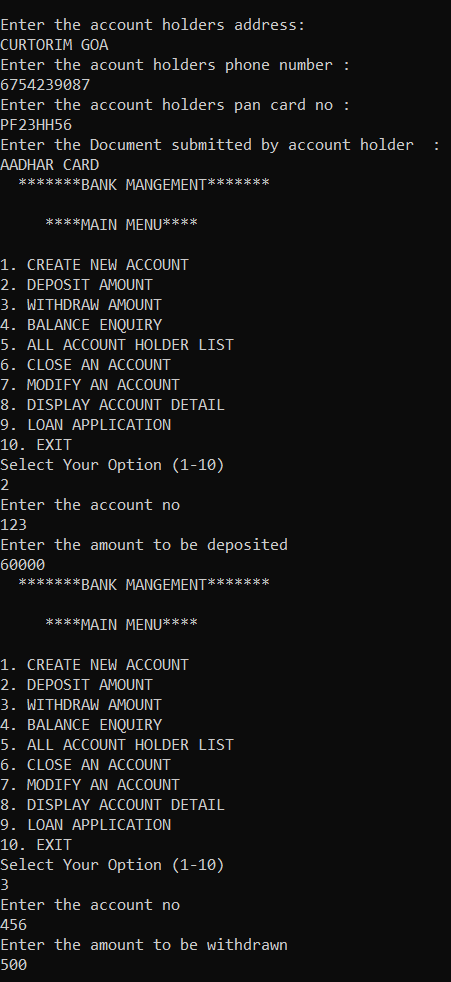
**return** 0;

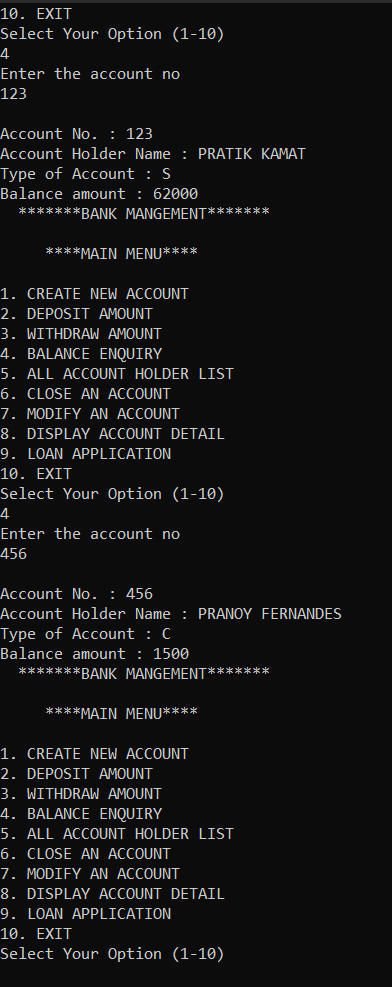
}

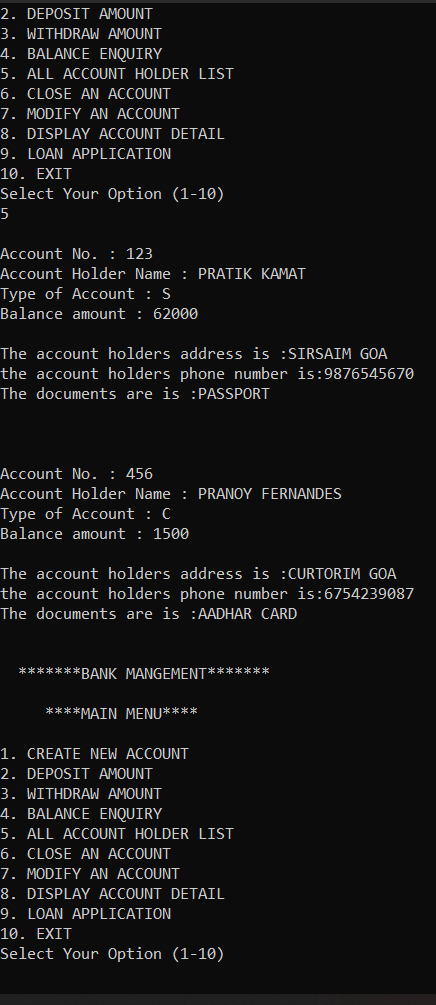
**OUTPUT**

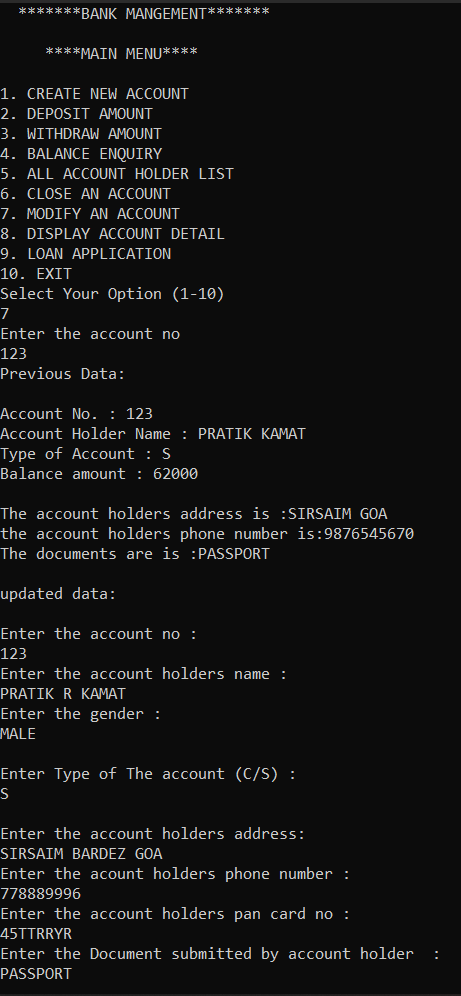
****

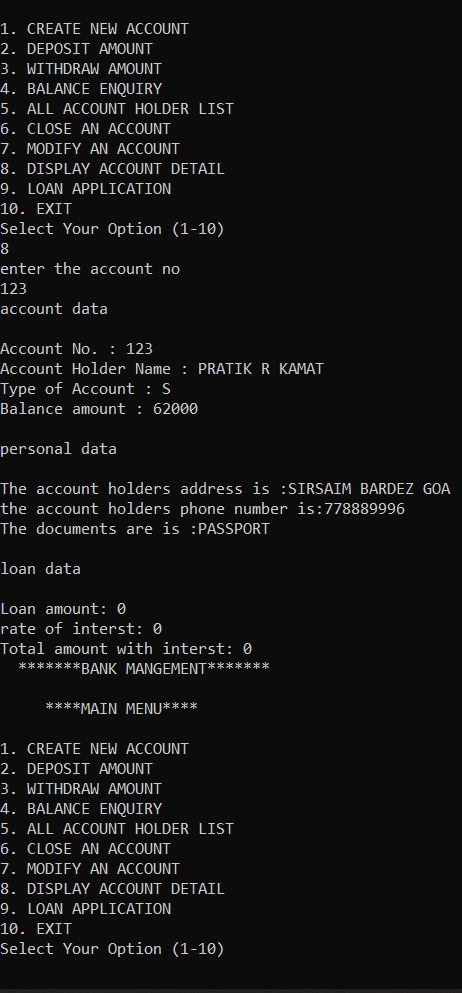
****

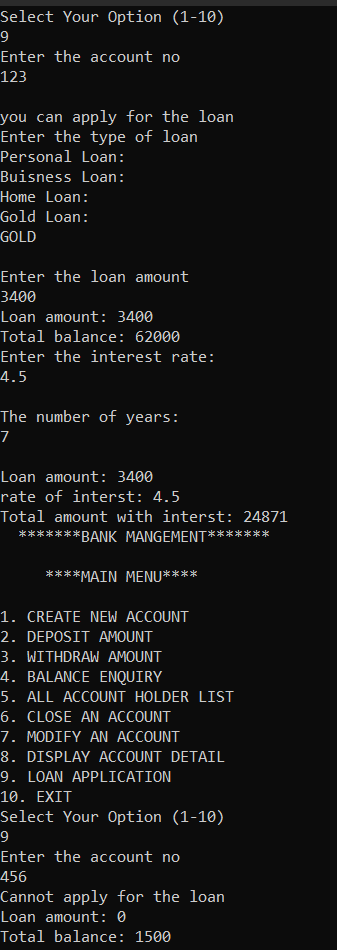
****

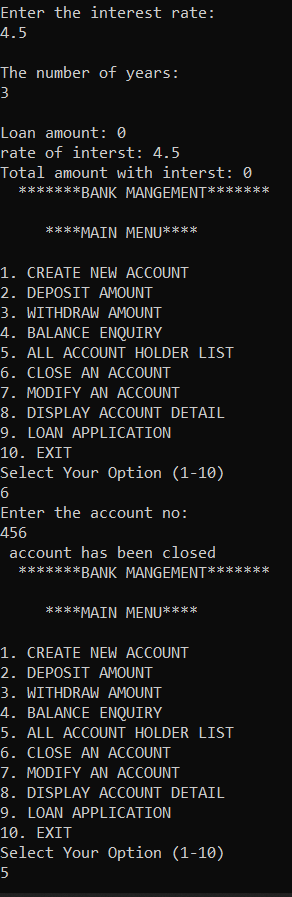
****

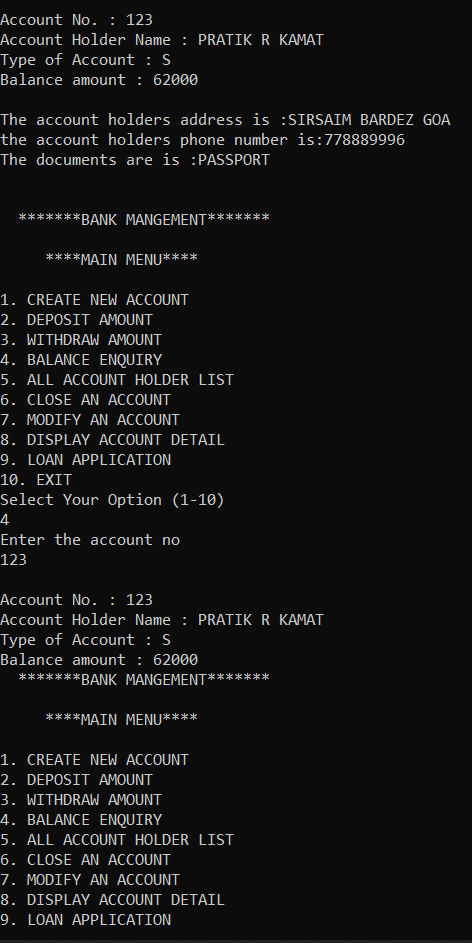
****

****

****

****

****

****

**THANK YOU**