جامعة المنوفية

| ختم الكنترول |
|---|

# استمارة تقييم بحث طلابي

**الفصل الدراسي الثاني للعام الجامعي ٢٠١٩/ ٢٠٢٠**

****يرجى التأكد من الا تزيد هذه الاستمارة بعد استكمال البيانات عن صفحة واحدة فقط**

أولا بيانات الطالب (تملأ بمعرفة الطالب

كلية : **الحاسبات والمعلومات**

**اسم الطالب: عمر فتح الله محمد حسن الجزارة**

**الرقم القومي : 30102241700315**

**البريد الإلكتروني الأكاديمي :** omarfath.9583@ci.menofia.edu.eg

**المستوى/ الفرقة**: الاولى

**ثانيا بيانات البحث (تملأ بمعرفة الطالب)**

**اسم المقرر:computer programming**

**اسم استاذ المقرر: ١-Dr/ handy m. mousa ٢-**

**عنوان موضوع البحث: **

**Bank Management System**

**ثالثا تقييم البحث**

**نسبة الاقتباس: هل البحث منقول ؟ ( فى حدود علم اللجنة)**

**نعم --** **لا__**

**في حالة ان الإجابة بنعم لا يتم تقييم البحث ويعتبر تقييم الطالب في البحث لم يجتاز**

| عناصر التقييم | مستوفى بدرجة | | |
|---|---|---|---|
| | ممتاز | جيدة | ضعيفة |
| | 3 | 2 | 1 |
| الشكل العام والصياغة اللغوية وجودة الكتابة | | | |
| شمول ملخص البحث. | | | |
| وضوح هدف البحث. | | | |
| وضوح مقدمة البحث. | | | |
| الدقة في عرض المحتوى و توافقه مع عنوان البحث | | | |
| توافق المراجع مع الموضوع البحثي. | | | |
| درجة النجاح 9 | نتيجة تقييم البحث: | | |

**رابعا التقييم النهائي**

| اجتاز | لم يجتاز | توقيع لجنة التقييم |
|---|---|---|
| | | ١-<br>٢-<br>٣- |

**في حالة عدم اجتياز الطالب للبحث يرجى ذكر الأسباب:**

١-

٢-

## Abstract

C ++ supports object-oriented programming, which is an advanced programming style, in which the program is divided into units called Objects. Each object is a bundle of data, variables, constants, functions, organizational units, and user interfaces. In a bank management system, I use oop as the system is divided into two classes , many oop tools, and many functions and function main to do. I can simplify all the functions through the OOP and the jobs and file you use to save data to create a new account, deposit and withdraw, balance inquiries, view all, close the account, and make an update.


## Introduction

The programming language is the programming language that programmers use to develop programs, scripts, or other sets of instructions for computers that are implemented to implement algorithms. Object Oriented Programming (OOP) and Procedural Programming are two types of programming paradigms. The programming model is a basic style of computer programming, and they differ in the way in which the various elements of the program are represented and how to define steps to solve problems. As the name implies, OOP focuses on representing problems using real things and their behavior, while procedural programming deals with representing problems with problems using procedures, which are sets of code that run in a specific order. There are programming languages that support the main aspects of OOP (called OOP languages), procedural (called procedural languages) and both. But one important thing to note is that OOP and Procedure are two ways to represent the problems to be solved, and it doesn't matter which language is used. In other words, OOP languages can be used for procedural programming while procedural languages can sometimes be used for OOP, with some effort. After learning OOP, problem solving became easier because OOP the focus is on thinking about the problem to be solved in terms of real-world elements and representing the problem in terms of objects and their behavior and Organize the code more easily and solve problems that are difficult to solve with regular programming. Which makes OOP a way of thinking. In my system bank the code is easy to handle parts .

## System Description

In the bank there are services such as creating a new account, updating an existing account information, viewing and managing transactions, checking the details of an existing .account, removing an existing account, and viewing the customer list

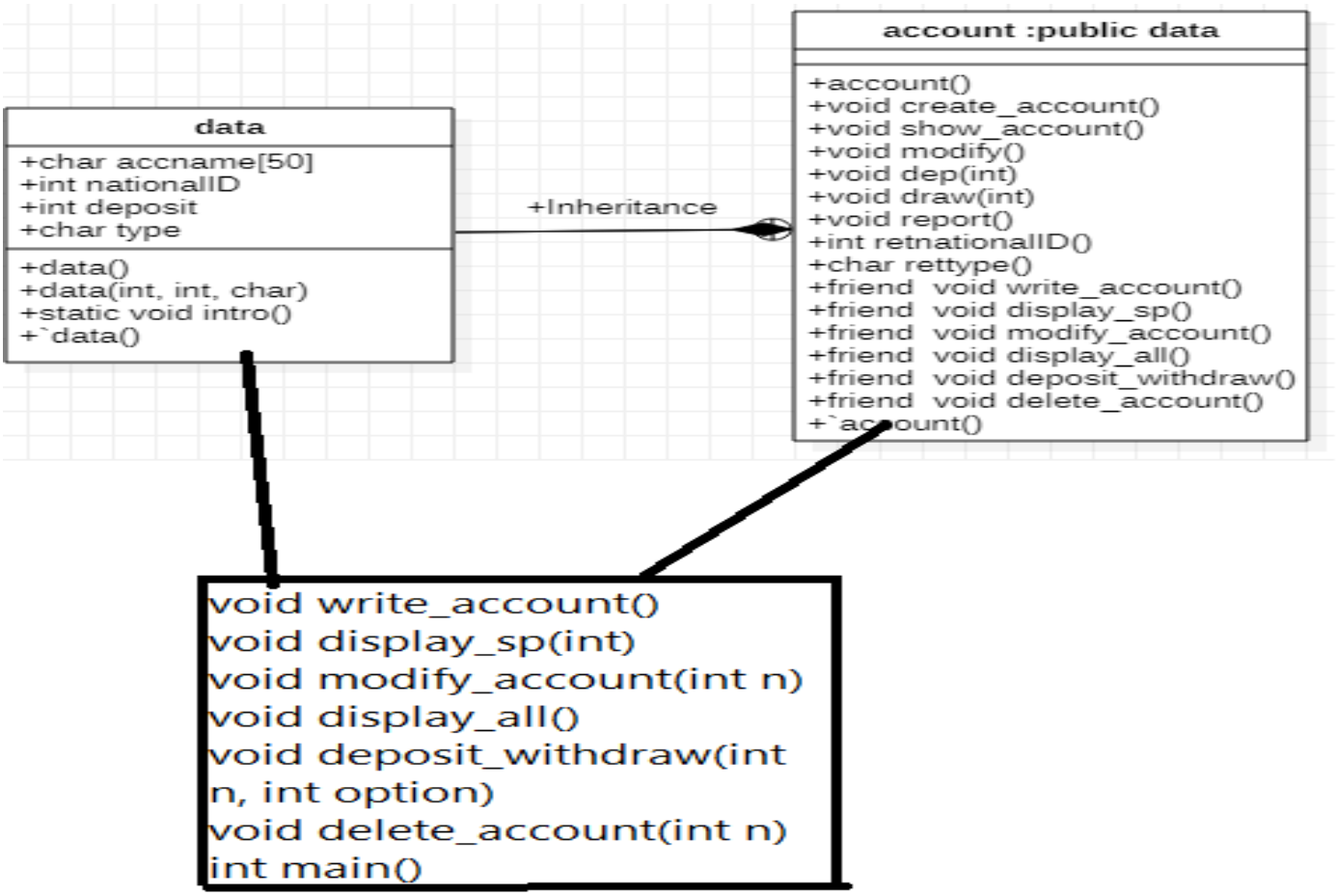
As in banks, error is forbidden and hard work is required to accomplish tasks for customers and keep their money important to the bank and customers in this system in general, you can do banking activities such as in a real bank. C ++ Bank Management Project is a console application .In order to do better work in this system I use oop in c ++. to OOP makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows .OOP provide data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere .OOP provide ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language .

In this system, it is divided into classes and functions, and the functions used in the bank are operated in an organized, simple, easy to control and modify manner .

Where he can use this, he opens a new account with the bank, specifies the type of account, adds a name to the account with a national number, deposits and withdraws the amount to the funds in the account in the bank, and can inquire about the balance in the bank, and the customer can make an adjustment to the account from a change of awareness and name The amount claimed, and the account can be deleted in the bank, and can view all account in bank.

All these functions are provided by the system in an organized way, and the data is saved in a file, which represents a database . C++ language allows us to perform important Disk input/output operations such as Creating a new file on the disk .Reading the file stored on the disk .Writing data to the file stored on the disk . Appending new data to the end of the file stored on the disk. Modifying the content of the file stored on the disk .To perform any file operations ,ifstream , to perform the file input operations .ofstream, to perform the file output operations .fstream , to perform any file input and output operation .

This is a plan to build this system to demonstrate that the system is working properly .




In the figure, it shows that the system consists of three parts, and each part has a specific-function.

**First part , class data :** It consists of two parts header file and cpp file , in this section contains : **declaring variable in program**, the variable is used to store data. The data type is specified according to its function that the variable can store, such as integer , character. Variable and datatype in system **( in public)** (char accname ,int nationalID , int deposit , char type ) .and **use constructor** is a special method which is invoked automatically at the time of object creation. It is used to initialize the data members of new object generally. The constructor in C++ has the same name as class or structure .There can be two types of constructors in c ++ .

Default constructor **data()** , Parameterized constructor **data**( int nationalID ,int deposit, char type)

{ this->nationalID=nationalID ;

this->deposit=deposit;

this->type=type;}

**this** is a keyword that refers to the current instance of the class. There can be 3 main usage of this keyword in C++.it can be used to pass current object as a parameter to another method . It can be used to refer current class instance variable .

It can be used to declare indexer**. A destructor** works opposite to constructor; it destructs the objects of classes. It can be defined only once in a class. Like constructors, it is invoked automatically. destructor is defined like constructor. It must have same name as class. But it is prefixed with a tilde sign (~) **: ~data() .**

And function **intro () :** it is void function and static **,** use static function ,static is a keyword or modifier that belongs to the type not instance. So, instance is not required to access the static members. In C++, static can be field, method, constructor, class, properties, operator and event .It is used to refer the common property of all objects . in this function contain information about program ,it like face **.**

**The second part : class account: public data,** it Inherits class data from type public in this

Class exist functions to create new account , account modify , account show , Withdraw ,deposit, Account report, and functions to return nationalID ,deposit, type .

**In public** exist Default constructor **account () ,** and destructor **~account () .**

**Function void create_account()** : in this function the user can open account in bank in this function enter national id and use **Exception Handling** when enter to keep because it a process to handle runtime errors. We perform exception handling so the normal flow of the application can be maintained even after runtime errors. All exceptions are derived from std::exception class. It is a runtime error which can be handled. If we don't handle the exception It maintains the normal flow of the application. In such case, rest of the code is executed even after exception. And enter account name in bank and use cin.getline ()

Even allowed to take the distance ,enter account type (Current , Saving or Investor) enter first character capital or small ,I use toupper(type) .and enter deposit when open account and also use **Exception Handling** when enter **to maintain the logical data . (not<0)**

**Function void modify () :** use this function when user wants to modify the data , before the user enter national id to show account and national id don't modify .then the user enter new data in account ,he can modify account name and account type enter first character capital or small ,I use toupper(type) , he can change the deposit when enter this, I use **Exception Handling** when enter **to maintain the logical data (not<0)** . it is like a new data.

**Function void dep() use** when the user deposit in account to add the next value to the account . It is called in function deposit_withdraw() .

**draw**() Use it when the user withdraws from the bank account . It is called in function **function void deposit_withdraw() .**

function **void show_account** () this function show data in account .

**function int retnationalID() const** function to return account national id .and this function const .

**function int retdeposit() const** : to return balance amount **.**

**function char rettype() const** to return account type .

**function void report() const :** this function to show data in tabular format **.**

**I use const in this functions** : because **Const member functions** ,Like member functions and member function arguments, the objects of a class can also be declared as const. an object declared as const cannot be modified and hence, can invoke only const member functions as these functions ensure not to modify the object.

A const object can be created by prefixing the const keyword to the object declaration. Any attempt to change the data member of const objects results in a compile-time error.

When a function is declared as const, it can be called on any type of object, const object as well as non-const objects.

Whenever an object is declared as const, it needs to be initialized at the time of declaration. however, the object initialization while declaring is possible only with the help of constructors.

A function becomes const when the const keyword is used in the function's declaration. The idea of const functions is not to allow them to modify the object on which they are called. It is recommended the practice to make as many functions const as possible so that accidental changes to objects are avoided.

**In the third stage, t**here are a group of functions outside the main function. These functions are friendly functions to **class account** , and use friend function because **a friend function**, that is a "friend" of a given class, is a function that is given the same access as methods to private and protected data. A friend function is declared by the class that is granting access, so friend functions are part of the class interface, like methods. Friend functions allow alternative syntax to use objects, for instance f(x) instead of x.f(), or g(x,y) instead of x.g(y). Friend functions have the same implications on encapsulation as methods .Use cases , this approach may be used in friendly function when a function needs to access private data in objects from two different classes. This may be accomplished in two similar ways ,a function of global or namespace scope may be declared as friend of both classes, a member function of one class may be declared as friend of another one.

**write_account ()** to add a new account in file , to write record in binary file by function write( ) This binary function is used to perform file output operation write the objects to a file, which is stored in the computer memory in a binary form . Only the data member of an object is written and not its member functions . and open file to save data like database and use mode ( ios::binary | ios::app ) and take **object from account class** to call function ***create_account().***

**void deposit_withdraw(int n, int option)** : This function is to withdraw and deposit in the account ,the user must enter the national ID to search for the account . When the user selects the job he wants, the user must enter the national ID to search for the account. Then open file to read all accounts ,take modes (ios::binary | ios::in | ios::out )and use a function read() This binary function is used to perform file input operation i.e. to read the

objects stored in a file. and take object from class account to call function show_account(). To view and verify account data . If the deposit option, should be call function dep() (this function exists in account class and take object from account class ) to add the value in the account. If the withdrew option , should be call function draw() ( this function exists in account class and take object from account class ) To take the required value from the account . use the seekp()function which takes the put pointer(used to write the content of a file) to one-byte position back from the current position of put pointer, after the searched character e is found . and call function write to save change . And close file .

**void display_sp(int n)** to display account details given by user. The user should enter national id to open file to read account details and take mode ( ios::binary ) and use function read () to search account and show by function show_account .

**void display_all ()** to display all accounts deposit list .open file take mode (ios::binary) to use function read to display all in bank and take object from account class to call function **report( )** and close file .

**function void modify_account(int n)** to modify record of file . take object from account class . the user should enter national id to search and chose account , open file and take mode (ios::binary | ios::in | ios::out) and use function read() to show account by function **show_account()** then use function seekp () , function write () to modify call function **modify()** and close file.

Function void **delete_account(int n)** this function to delete account in bank . take object from account class .the user should enter national id to Select the account to delete .open file take mode (ios::binary) and search in file ,use function seekp() and read () and delete this account by function remove() and close all files.

In **main function** take object from data class to call **function intro()** and use to The Switch / case statement implements one of several conditions for the functions to work in an orderly manner and also causes the user to choose the function according to choice from [1-8] : 1 to open New Account ,2 to Deposit Amount to Money ,3 to Withdraw Amount to

Money ,4 to Balance Enquiry ,5 to View ALL ACCOUNT in bank ,6 to Close an Account ,7 to Modify an Account ,8 to end . use system ("cls") To make each function on a new page.

**Syntax error**

```
#ifndef ACCOUNT_H
#define ACCOUNT_H
#include<iomanip>
#include<iostream>
#include<fstream>
using namespace std;
class account:public data
{
```

| | |
|---|---|
| C:\Users\student\Desktop\BANK\main.cpp | In file included from main.cpp |
| C:\Users\student\Desktop\BANK\account.h | [Error] expected class-name before '{' token |
| C:\Users\student\Desktop\BANK\account.h | In member function 'int account::retnationalID() const': |
| C:\Users\student\Desktop\BANK\account.h | [Error] 'nationalID' was not declared in this scope |
| C:\Users\student\Desktop\BANK\account.h | In member function 'int account::retdeposit() const': |
| C:\Users\student\Desktop\BANK\account.h | [Error] 'deposit' was not declared in this scope |
| C:\Users\student\Desktop\BANK\account.h | In member function 'char account::rettype() const': |

Don't write **#include "data.h"**

```
public
    account();
```

| | |
|---|---|
| C:\Users\student\Desktop\BANK\main.cpp | In file included from main.cpp |
| C:\Users\student\Desktop\BANK\account.h | [Error] expected ':' before 'account' |
| C:\Users\student\Desktop\BANK\Makefile.win | recipe for target 'main.o' failed |

Don't write **:**

```
                cout << "\n\n
                break;
            default:cout << "\a"
            cin.ignore();
            cin.get();
        } while (ch != '8');
        return 0;
    }
```

| | |
|---|---|
| C:\Users\student\Desktop\BANK\main.cpp | In function 'int main()': |
| C:\Users\student\Desktop\BANK\main.cpp | [Error] expected 'while' before 'write_account' |
| C:\Users\student\Desktop\BANK\main.cpp | [Error] expected '(' before 'write_account' |
| C:\Users\student\Desktop\BANK\main.cpp | [Error] could not convert 'write_account()' from 'void' to 'bool' |
| C:\Users\student\Desktop\BANK\main.cpp | [Error] expected ')' before '{' token |
| C:\Users\student\Desktop\BANK\main.cpp | [Error] expected ';' before '{' token |

Don't write **}** end switch/ case

The face the program

```
                                WELCAME  TO  MANAGEMENT  BANK  SYSTEM
                              ENTER  ANY  KEY  TO  ENTER  TO MENU TO WORK IN BANK

 BY OMAR  FATHALLAH  MOHAMMED  ELGAZARA


        Dr/ HAMDY M MOUSA
```

The menu in progrem

```
                                YOUR  SYSTEM  MANAGMENT  BANK
                              YOUR      SERVICES  IN  BANK

     01: open New Account
     02:  Deposet Amount  to Money
     03:  Withdraw Amount to Money
     04:  Balanc Enquiry
     05:  Veiw ALL ACCOUNT in bank
     06:  Close an Account
     07:  Modify an Account
     08:  EXIT AND END

     Option (1-8) to opration
```

When open account

```
 Enter your National ID : 147
 Enter The Name of The account in Bank    : hady_102
 Enter Type of The account (Current,Saving or Investor) AND ENTER 'C', 'S' OR 'I' : c
 Enter The Initial amount :1000

nt Created
```

Display all in bank

```
********************  ********************************************
A/cnational id.          NAME                    Type          Balance
********************  ********************************************
120                      omqr                    C                 120
111                      mohamed                   S               100000
```

Modify

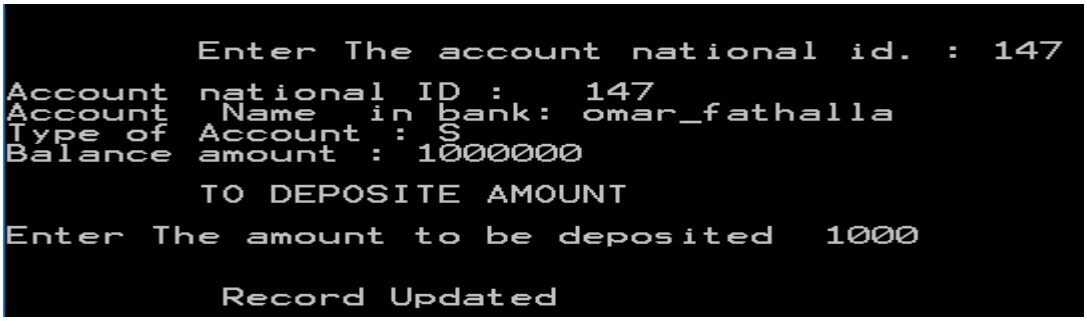
```
Account national ID :    147
Account  Name  in bank: hady_102
Type of Account : C
Balance amount : 1000

Enter The New Details of account

 nationalID your Account   : 147
Modify Account  Name : omar_fathalla
 Modify Type of Account(Current or Saving ) : s
 Modify Balance amount   (C >1000/S>500): 1000000

          Record Updated
```

Deposit money in account




You can add a class in the system to employee

**conclusion**

In this program, I followed the steps that were planned for the program by drawing the program that you set at the top so that the project achieves the goals required in the bank, which means that it can open or close an account and withdraw or deposit and do an account test and amend the account with all these capabilities that distinguish the system and used oop with the basic principles of the language and used the file to represent a database .

**references**

Book of the material 2019/2020 of Dr. Hamdy Mousa.

Deitel, Paul J. **C++ : how to program** / P.J. Deitel, H.M. Deitel. -- 8th ed .

**Thinking in C++,**Volume 1, 2nd Edition Completed January 13, 2000 Bruce Eckel, President, Mind View, Inc

**The C++ Programming Language** Third Edition Bjarne Stroustrup AT&T Labs Murray Hill, New Jerse .

https://www.decodejava.com/introduction-to-c-plus-plus-language.htm

https://www.javatpoint.com/cpp-tutorial

https://www.decodejava.com/cpp-file-and-file-modes.htm

**Appendix**

```
"include "account.h#
"include "data.h#
<include<fstream#
<include<cctype#
;using namespace std
void write_account();     //function to write record in binary file
void deposit_withdraw(int n, int option);// function to deposit and withdraw amounts
void display_sp(int);     //function to display account details given by user
void modify_account(int n);//    function to modify record of file
void display_all();// function to display all accounts deposit list
```

```
void delete_account(int n);//function to delete record of file
()int main
;data d     }
;()d.intro
;char ch
;int num
do
}
;system("cls")
;"cout << " \n\n\n\t\t\t\tYOUR  SYSTEM   MANAGMENT  BANK
cout << "
SERVICES  IN   BANK                                                                                                    \n\n\n\t\t\t      YOUR
;" \n
;"                          cout << " \n\n\t     01: open New Account
;"cout << " \n\n\t     02:  Deposet Amount  to Money
;"cout << " \n\n\t     03:  Withdraw Amount to Money
;"           cout << " \n\n\t     04:  Balanc Enquiry
;"cout << " \n\n\t     05:  Veiw ALL ACCOUNT in bank
;"        cout << " \n\n\t     06:  Close an Account
;"    cout << " \n\n\t     07:  Modify an Account
;" cout << " \n\n\t     08:  EXIT AND END\n\n\n
;"     cout << "\n\n Select Your Option (1-8) to opration
;cin >> ch
;system("cls")
switch (ch)
}
:'case '1
;system("cls")
;()write_account

;break
:'case '2
;system("cls")
;cout << "\n\n\tEnter The account national id. : "; cin >> num
;deposit_withdraw(num, 1)

;break
:'case '3
;system("cls")
```

```
;cout << "\n\n\tEnter The account national id. : "; cin >> num
;deposit_withdraw(num, 2)

;break
:'case '4
;system("cls")
;cout << "\n\n\tEnter The account national id. : "; cin >> num
;display_sp(num)

;break
:'case '5
;system("cls")
;()display_all

;break
:'case '6
;system("cls")
;cout << "\n\n\tEnter The account national id : "; cin >> num
;delete_account(num)
;break
:'case '7
;system("cls")
;cout << "\n\n\tEnter The account national id. : "; cin >> num
;modify_account(num)
;break
:'case '8
;system("cls")
;"cout << "\n\n\t\t\t\a\a\a THANKS YOUR IN IN YUOR BANK MANAGMENT SYSTEM
;break
;"default:cout << "\a
{
;()cin.ignore
;()cin.get
;('while (ch != '8 {
;return 0
{
void write_account() // to add anew account in file
}
```

```
account ac; // object
;ofstream outFile
;outFile.open("account.dat", ios::binary | ios::app)
;()ac.create_account
;outFile.write(reinterpret_cast<char *> (&ac), sizeof(account))
;()outFile.close
{
function to display account details given by user//
function to read specific record from file     //
void display_sp(int n)
}
;account ac
;bool flag = false
;ifstream inFile
;inFile.open("account.dat", ios::binary)
if (!inFile)
}
;"...cout << "File could not be open !! Press any Key
;return
{
;"cout << "\n Balance Detais  \n
while (inFile.read(reinterpret_cast<char *> (&ac), sizeof(account)))
}
if (ac.retnationalID() == n) //function to return national id
}
;()ac.show_account
;flag = true
{
{
;()inFile.close
if (flag == false)
;"cout << "\n\n\aAccount number does not exist
{
function to display all accounts deposit list //
()void display_all
}
;account ac
;ifstream inFile
```

```
;inFile.open("account.dat", ios::binary)
if (!inFile)
}
;"...cout << "File could not be open !! Press any Key
;return
{
;"cout << "\n\n\t\tAccount  list in bank \n\n
;"cout << "****************** ********************************************\n
;"cout << "A/cnational id.        NAME            Type        Balance\n
;"cout << "****************** ********************************************\n
while (inFile.read(reinterpret_cast<char *> (&ac), sizeof(account)))
}
;()ac.report
{
;()inFile.close
{
function to deposit and withdraw amounts    //
void deposit_withdraw(int n, int option)
}
;int amt
;bool found = false
;account ac
;fstream File
;File.open("account.dat", ios::binary | ios::in | ios::out)
if (!File)
}
;"...cout << "File could not be open !! Press any Key
;return
{
while (!File.eof() && found == false)
}
;File.read(reinterpret_cast<char *> (&ac), sizeof(account))
if (ac.retnationalID() == n)
}
;()ac.show_account
if (option == 1)
}
;" cout << "\n\n\tTO DEPOSITE AMOUNT
```

```
;"cout << "\n\nEnter The amount to be deposited
;cin >> amt
;ac.dep(amt)
{
if (option == 2)
}
;" cout << "\n\n\tTO WITHDRAW AMOUNT
;"cout << "\n\nEnter The amount to be withdraw
;cin >> amt
;int bal = ac.retdeposit() - amt
if ((bal<500 && ac.rettype() == 'S') || (bal<1000 && ac.rettype() == 'C'))
;"cout << "Insufficience balance
else
;ac.draw(amt)
{
;int pos = (-1)*static_cast<int>(sizeof(ac))
;File.seekp(pos, ios::cur)
;File.write(reinterpret_cast<char *> (&ac), sizeof(account))
;"cout << "\n\n\t Record Updated
;found = true
{
{
;()File.close
if (found == false)
;" cout << "\n\n Record Not Found
{
function to modify record of file     //
void modify_account(int n)
}
;bool found = false
;account ac
;fstream File
;File.open("account.dat", ios::binary | ios::in | ios::out)
if (!File)
}
;"...cout << "File could not be open !! Press any Key
;return
{
```

```
while (!File.eof() && found == false)
}
;File.read(reinterpret_cast<char *> (&ac), sizeof(account))
if (ac.retnationalID() == n)
}
ac.show_account(); // to show data of account in bank
;cout << "\n\nEnter The New Details of account" << endl
ac.modify();// function to modify class account
;int pos = (-1)*static_cast<int>(sizeof(account))
;File.seekp(pos, ios::cur)
;File.write(reinterpret_cast<char *> (&ac), sizeof(account))
;"cout << "\n\n\t Record Updated
;found = true
{
{
;()File.close
if (found == false)
;"cout << "\n\n Record Not Found \n

{
function to delete record of file     //
void delete_account(int n)//function to delete record of file
}
account ac; // object ac from class account

;ifstream inFile
;ofstream outFile
;inFile.open("account.dat", ios::binary)
if (!inFile)
}
;"...cout << "File could not be open !! Press any Key
;return
{
;outFile.open("Temp.dat", ios::binary)
;inFile.seekg(0, ios::beg)
while (inFile.read(reinterpret_cast<char *> (&ac), sizeof(account)))
}
if (ac.retnationalID() != n)
```

```
}
;outFile.write(reinterpret_cast<char *> (&ac), sizeof(account))
{
{
;()inFile.close
;()outFile.close
;remove("account.dat")
;rename("Temp.dat", "account.dat")
;".. cout << "\n\n\tRecord Deleted
{
```

---

```
ifndef DATA_H#
define DATA_H#
<include <iostream#
;using namespace std
class data
:public}
;[ᵒ٠]char accname
;int nationalID
;int deposit
;char type
;()data
data( int nationalID,int deposit, char type)
}
this-
;>nationalID=nationalID
;this->deposit=deposit
;this->type=type
{
() static void intro
}

;" cout<<"\n\n\n\n\n\n\t\t\t\t\t\t WELCAME  TO  MANAGEMENT  BANK  SYSTEM \n
cout<<" \n\n
;"\t\t\t\t\t   ENTER  ANY  KEY  TO  ENTER  TO MENU TO WORK IN BANK \n\n
cout<<" \n\n\n\n\t\t
;"BY OMAR  FATHALLAH  MOHAMMED  ELGAZARA\n\n\n\n\n
cout<<"
;"\n\n\n\n\t\t      Dr/ HAMDY M MOUSA \n\n\n\n\n
            ;()cin.get
{
```

```
;()data~
:protected
;{
endif#
"include "data.h#
()data::data
}
{
()data::~data
}
{
```

---

```
ifndef ACCOUNT_H#
define ACCOUNT_H#
<include<iomanip#
<include<iostream#
<include<fstream#
"include "data.h#
;using namespace std
class account:public data
}
:public
;()account
void
create_account();    //function to get data from user and this called in write account
void modify();
//function to add new data
void dep(int);      //function to accept amount and add to balance amount
void draw(int);     //function to accept amount and subtract from balance amount
void show_account()  ; //function to show data on screen>Balanc Enquiry
int retnationalID() const//function to return account number
}
; return nationalID
                                                                                {
int retdeposit() const
}
;return deposit
{
char rettype() const
}
```

```
;return type

{

void report() const// when check

}

;cout << nationalID << setw(20) << " " << accname << setw(15) << " " << type << setw(20) << deposit << endl

{

functions   frirnd in class account bank //

friend  void
;()write_account

friend  void
;()display_sp

friend  void
;()modify_account

friend  void
;()display_all

friend  void
;()deposit_withdraw

friend  void
;()delete_account

; ()account~

;{

endif#

"include "account.h#

<include<iostream#

<include<iomanip#

;using namespace std

()account::account

}

{

()void account::create_account

}

;try{// exception handing by used try and catch

;" : cout << "\n\tEnter your National ID

;cin >>nationalID

if (nationalID <= 0)

}

throw "
;"enter correct namber

} else {

{

(catch( const char *err  {
```

```
}

cout<<"\n
;exception :"<<err

cout <<
;" : "\n\n\n Enter anew your National ID

;cin >>nationalID

end try  //   {
and catch and exception handing of national id

;" :    cout << "\n\tEnter The Name of The account in Bank

;()cin.ignore

;cin.getline(accname, 50)

;" : 'cout << "\n\tEnter Type of The account (Current,Saving or Investor) AND ENTER 'C', 'S' OR 'I

;cin >> type

type = toupper(type); //t==T

}try

;": cout << "\n\tEnter The Initial amount

;cin >>deposit

if (deposit<=0)

}

throw "
;"enter correct namber

{;else// {continue//{

(catch( const char *err  {

}

cout<<"
;exception :"<<err

cout<< "
;":anew nember courct \n

cin
;>>deposit

end try  ////  {
and catch and exception handing of deposit

;"    cout << "\n\n\n  Account Created

{

()void account::modify

}

;cout << "\n nationalID your Account  : " << nationalID


;" : cout << "\nModify Account  Name

cin.ignore();// used becouse getline

;cin.getline(accname, 50)
```

```
;" : cout << "\n Modify Type of Account(Current or Saving )

;cin >> type

;type = toupper(type)


}try

;" :cout << "\n Modify Balance amount  (C >1000/S>500)

;cin >>deposit


if (deposit<=0)

}

throw "
;"enter correct namber

/{

(catch( const char *err  {

}

cout<<"
;exception :"<<err

cout<< "
;":anew nember courct \n

cin
;>>deposit

{

{

void account::dep(int x)

}

deposit += x;// when deposit

{

void account::draw(int x)

}

deposit -= x;//When withdraw

{

()void account::show_account

}

; cout << "\nAccount national ID :   " <<nationalID

;" :cout << "\nAccount  Name  in bank

;cout << accname

;cout << "\nType of Account : " << type

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

{

()account::~account{}
```