2/10/2025

Muhammad Arham.Usman.

Fast Nuces

Object Oriented Programming

Lab 4

Question 1:

# Source Code:

#include <iostream>

using namespace std;

class Student{

    int studentID;

    string name;

    int age;

    char grade;

    public:

    Student(int studentID=0, string name="", int age=0, char grade='N'){

        this->studentID=studentID;

        this->name=name;

        this->age=age;

        this->grade=grade;

    }

    void setID(int student\_id){

        studentID=student\_id;

    }

    void setName(string Name){

        name=Name;

    }

    void setAge(int Age){

        age=Age;

    }

    void setGrade(char Grade){

        grade=Grade;

    }

    int getID(){

        return studentID;

    }

    string getName(){

        return name;

    }

    int getAge(){

        return age;

    }

    char getGrade(){

        return grade;

    }

    void promoteStudent(){

        if (grade>'A' && grade<='D'){

            grade-=1;

        }

        else if (grade=='F'){

            grade='D';

        }

    }

    bool isEligibleForScholarship(){

        if (grade=='A') return true;

        return false;

    }

    void displayDetails(){

        cout<<"\n-----Displaying Student Details-----\n";

        cout<<"Student ID: "<<studentID;

        cout<<"\nName: "<<name;

        cout<<"\nAge: "<<age;

        cout<<"\nGrade: "<<grade;

        cout<<"\n----------\n";

    }

};

int main(){

    int n, temp\_i;

    string temp\_s;

    char temp\_c;

    //Getting no of student

    cout<<"Enter number of students: ";

    cin>>n;

    //using default constructor

    Student\* class\_1=new Student[n];

    for (int i=0; i<n; i++){

        cout<<"Enter Details for Student "<<i+1<<endl;

        cout<<"Enter Student ID: ";

        cin>>temp\_i;

        class\_1[i].setID(temp\_i);

        cout<<"Enter Name: ";

        cin.ignore();

        getline(cin,temp\_s);

        class\_1[i].setName(temp\_s);

        cout<<"Enter Age: ";

        cin>>temp\_i;

        class\_1[i].setAge(temp\_i);

        cout<<"Enter Grade: ";

        cin.ignore();

        cin>>temp\_c;

        cin.ignore();

        class\_1[i].setGrade(temp\_c);

    }

    //using parameterised constructor

    Student new\_student(1016, "Bilal", 30, 'B');

    new\_student.promoteStudent();

    cout<<"\nIs "<<new\_student.getName()<<" Eligible for scholarship: "<<new\_student.isEligibleForScholarship();

    new\_student.displayDetails();

    for (int i=0; i<n; i++){

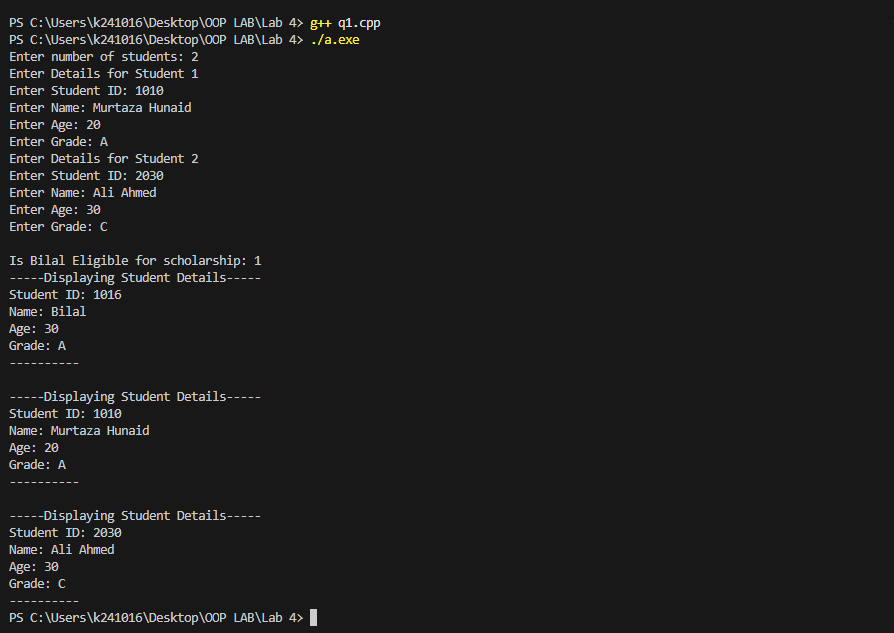
        class\_1[i].displayDetails();

    }

    delete[] class\_1;

}

# Output:



Question 2:

# Source Code:

/\*

A library system allows users to borrow and return books. The system should:

1. Add new books to the collection.

2. Borrow books (check availability).

3. Return books.

4. Display all available books.

Requirements:

- Implement a Book class with attributes: ID, title, author, availability.

- Implement a Library class with:

1. Method to add a book (with unique ID).

2. Method to borrow a book (updates availability).

3. Method to return a book (marks it available).

4. Method to display all available books.

- Store book records dynamically using pointers and DMA.

- Use constructor overloading to initialize books with or without availability status.

\*/

#include <iostream>

using namespace std;

class Book{

    string ID, title, author, availability;

    public:

    Book(){

        this->author="";

        this->availability="";

        this->ID="";

        this->title="";

    }

    Book(string id, string Title, string Author, string Availability){

        this->author=Author;

        this->availability=Availability;

        this->ID=id;

        this->title=Title;

    }

    string get\_ID(){

        return ID;

    }

    string get\_title(){

        return title;

    }

    string get\_author(){

        return author;

    }

    string get\_availability(){

        return availability;

    }

    void set\_ID(string ID){

        this->ID = ID;

    }

    void set\_title(string title){

        this->title = title;

    }

    void set\_author(string author){

        this->author = author;

    }

    void set\_availability(string availability){

        this->availability = availability;

    }

    void displayDetails(){

        cout<<"Id: "<<ID;

        cout<<"\nTitle: "<<title;

        cout<<"\nAuthor: "<<author<<endl;

    }

};

class Library{

    Book \*books;

    int noOfBooks;

    public:

        Library(Book\* Books=nullptr, int NoOfBooks=0){

            noOfBooks=NoOfBooks;

            if (noOfBooks==0) books=nullptr;

            else{

                books=new Book[noOfBooks];

                for (int i=0; i<noOfBooks; i++){

                    books[i]=Books[i];

                }

            }

        }

        ~Library(){

            delete[] books;

        }

        void AddBook(Book book){

            Book \*new\_stock=new Book[++noOfBooks];

            int i, found=0;

            for (i=0; i<noOfBooks-1; i++){

                if (books[i].get\_ID()==book.get\_ID()){

                    found=1; break;

                }

                new\_stock[i]=books[i];

            }

            if (found==0){

                new\_stock[i]=book;

                delete[] books;

                books=new\_stock;

                cout<<"\n-----Book added successfully-----\n";

            }

            else{

                delete[] new\_stock;

                noOfBooks--;

                cout<<"\n-----Error! Book must have a unique ID-----\n";

            }

        }

        void BorrowBook(string id){

            for (int i=0; i<noOfBooks; i++){

                if (id==books[i].get\_ID()){

                    if (books[i].get\_availability()=="y"){

                        books[i].set\_availability("n");

                        cout<<"\n-----Book issued successfully-----\n";

                    }

                    else{

                        cout<<"\n-----Book is not available-----\n";

                    }

                    return;

                }

            }

            cout<<"\n-----Book is not found-----\n";

        }

        void ReturnBook(string id){

            for (int i=0; i<noOfBooks; i++){

                if (id==books[i].get\_ID()){

                    books[i].set\_availability("y");

                    cout<<"\n-----Book returned sucessfully-----\n";

                    return;

                }

            }

            cout<<"\n-----Book is not found-----\n";

        }

        void DisplayAll(){

            cout<<"\n-----Displaying all books-----\n";

            for (int i=0; i<noOfBooks; i++){

                if (books[i].get\_availability()=="y"){

                    cout<<"Book "<<i+1<<endl;

                    books[i].displayDetails();

                    cout<<endl;

                }

            }

            cout<<"----------\n";

        }

};

int main(){

    string choice, id, title, author;

    Library lib1;

    Book b1;

    while (1){

        cout<<"\n=====Main Menu=====\n";

        cout<<"Press 1 to add a new book\nPress 2 to borrow a book\nPress 3 to return a book\nPress 4 to display all available books\nPress any other key to exit :)\n";

        cout<<"Enter your choice: ";

        cin>>choice;

        if (choice=="1"){

            cout<<"\n----------\n";

            cout<<"Enter id: ";

            cin>>id;

            cout<<"Enter title: ";

            cin.ignore();

            getline(cin, title);

            cout<<"Enter author: ";

            getline(cin, author);

            lib1.AddBook(Book(id, title, author, "y"));

        }

        else if (choice=="2"){

            cout<<"\n----------\n";

            cout<<"Enter id: ";

            cin>>id;

            lib1.BorrowBook(id);

        }

        else if (choice=="3"){

            cout<<"\n----------\n";

            cout<<"Enter id: ";

            cin>>id;

            lib1.ReturnBook(id);

        }

        else if (choice=="4"){

            lib1.DisplayAll();

        }

        else{

            cout<<"\n=====Thank You for using our service=====";

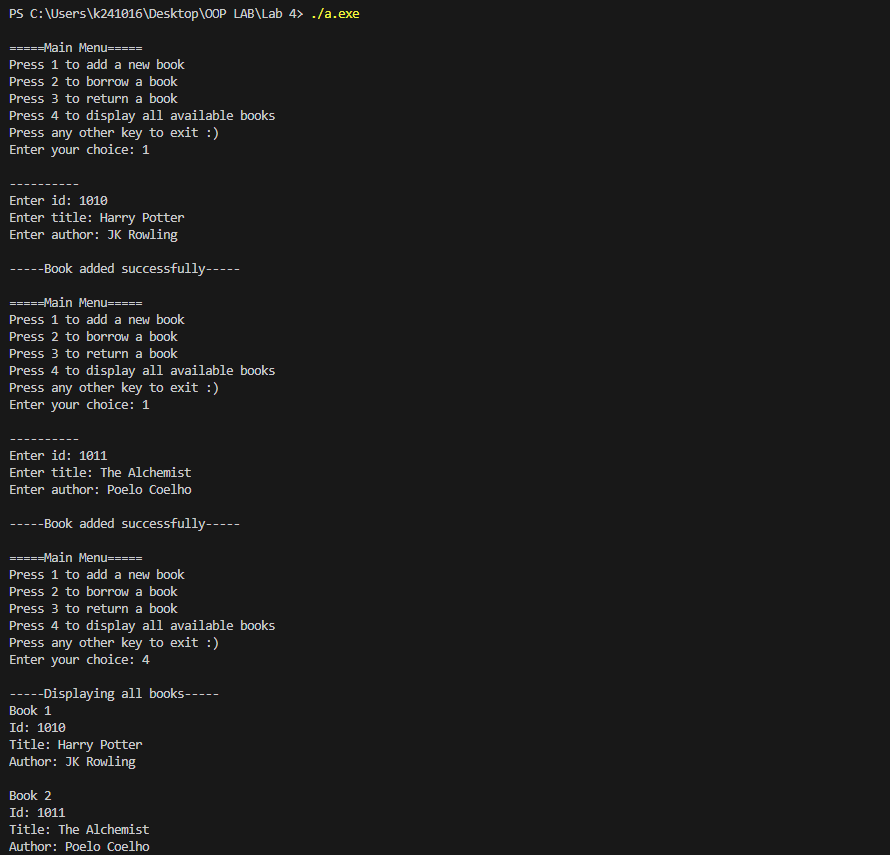
            break;

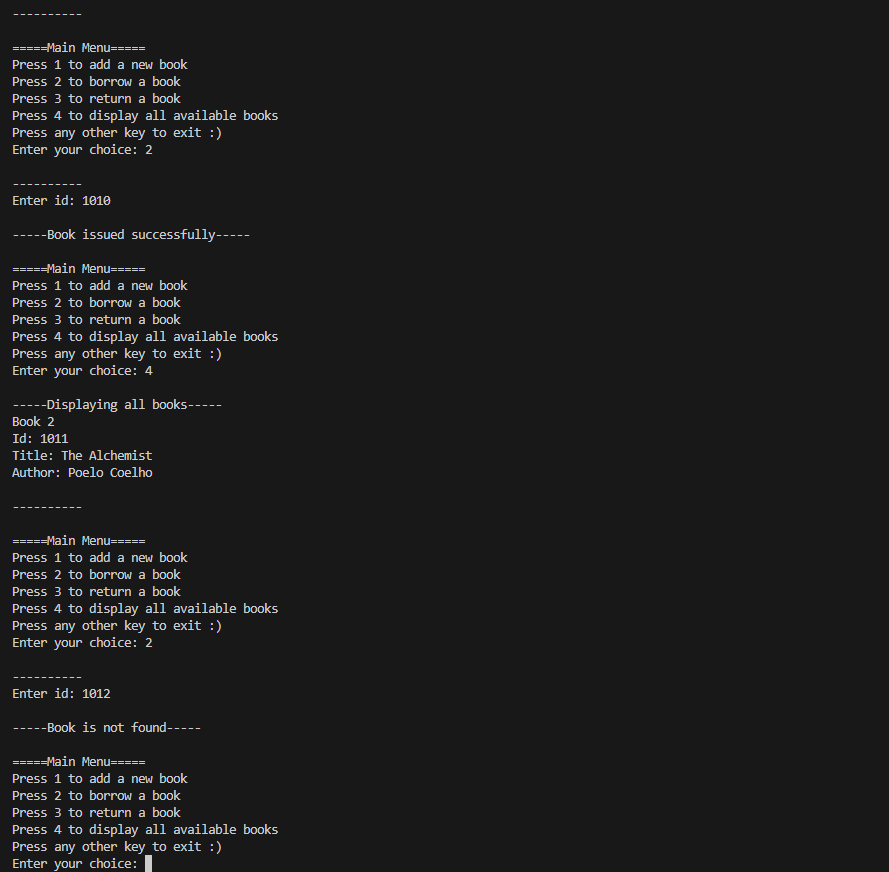
        }

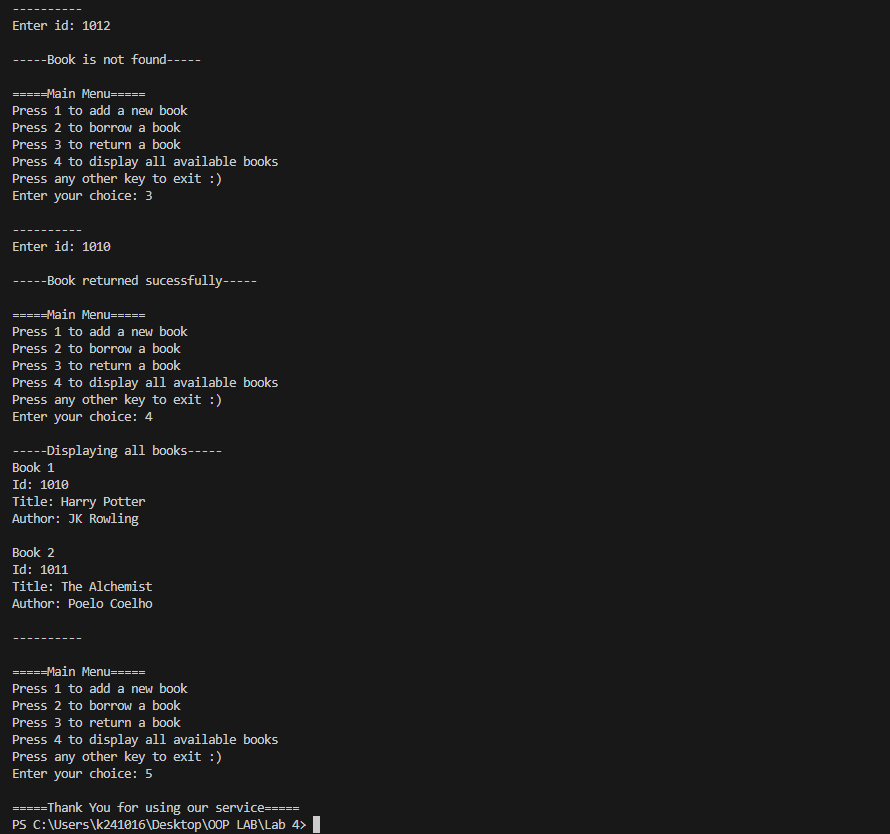
    }

}

# Output:







Question 3:

# Source Code:

/\*

You are building a bank account management system. Create a class `Account` to manage bank

accounts.

Requirements:

1. Attributes:

- `accountNumber` (string)

- `accountHolderName` (string)

- `balance` (double)

2. Define a default constructor that initializes `balance` to `0.0`.

3. Define a parameterized constructor to initialize all attributes.

4. Add methods:

- `deposit(double amount)`: Adds the amount to the balance.

- `withdraw(double amount)`: Deducts the amount from the balance (if sufficient funds are

available).

- `checkBalance()`: Displays the current balance.

5. Create a few `Account` objects and test the methods.

\*/

#include <iostream>

using namespace std;

class Account{

    string accountNumber;

    string accountHolderName;

    double balance;

    public:

    Account(string Account\_Number="", string Holder\_Name="", double Balance=0.0){

        accountNumber=Account\_Number;

        accountHolderName=Holder\_Name;

        balance=Balance;

    }

    void deposit(double amount){

        balance+=amount;

        cout<<"\nAmount deposited successfully in Account "<< accountNumber<<endl;

    }

    void withdraw(double amount){

        if (amount<=balance){

            balance-=amount;

            cout<<"\nAmount withdrawed successfully from Account "<< accountNumber<<endl;

        }

        else{

            cout<<"\nTransaction Failed! Insufficient Balance in account "<< accountNumber<<endl;

        }

    }

    void checkBalance(){

        cout<<"\n-------------";

        cout<<"\nAccount Title: "<<accountHolderName;

        cout<<"\nBalance: "<<balance;

        cout<<"\n-------------\n";

    }

};

int main(){

    Account \*a= new Account[5];

    a[0]=Account("PK1014", "Darwin", 1014);

    a[1]=Account("PK1010", "Ali", 1010);

    a[2]=Account("PK1011", "Ahmed", 1011);

    a[3]=Account("PK1012", "Bilal", 1012);

    a[4]=Account("PK1013", "Charlie", 1013);

    a[1].deposit(1000.50);

    a[2].withdraw(1011);

    a[3].withdraw(1014);

    for (int i=0; i<5; i++){

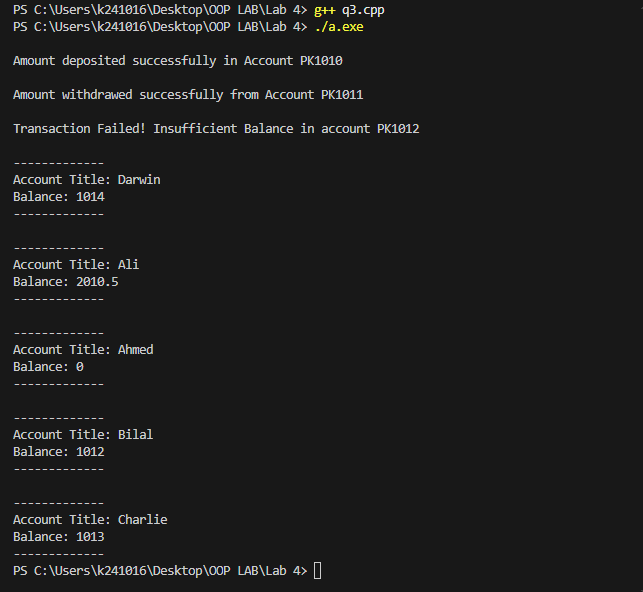
        a[i].checkBalance();

    }

    delete[] a;

}

# Output:



Question 4:

# Source Code:

/\*

You are building a car rental system. Create a class `Car` to manage cars available for rent.

Requirements:

1. Attributes:

- `carID` (int)

- `model` (string)

- `year` (int)

- `isRented` (bool)

2. Define a default constructor that initializes `isRented` to `false`.

3. Define a parameterized constructor to initialize all attributes.

4. Add methods:

- `rentCar()`: Marks the car as rented.

- `returnCar()`: Marks the car as available.

- `isVintage()`: Returns `true` if the car's year is before 2000.

5. Create a few `Car` objects and test the methods.

\*/

#include <iostream>

using namespace std;

class Car {

private:

    int carID;

    string model;

    int year;

    bool isRented;

public:

    Car() {

        isRented = false;

    }

    Car(int car\_id, string Model, int Year) {

        carID = car\_id;

        model = Model;

        year = Year;

        isRented = false;

    }

    void rentCar() {

        if (isRented) {

            cout << "Error! Car with ID " << carID << " is already rented.\n";

        }

        else {

            isRented = true;

            cout << "Success! Car with ID " << carID << " is now rented.\n";

        }

    }

    void returnCar() {

        if (isRented) {

            isRented = false;

            cout << "Success! Car with ID " << carID << " is now returned.\n"; }

        else {

            cout << "Error! Car with ID " << carID << " is already available.\n"; }

    }

    bool isVintage() const {

        if (year < 2000) {

            return true; }

        else {

            return false; }

    }

};

int main() {

    Car car1(1010, "Bugatti Chiron", 2010);

    Car car2(1011, "Porsche 911", 1985);

    Car car3(1012, "Benz s 6200", 2018);

    car1.rentCar();

    car1.returnCar();

    car1.rentCar();

    car2.rentCar();

    car2.returnCar();

    if (car3.isVintage()) {

        cout << "The car 3 is vintage" << endl;

    }

    else {

        cout << "The car 3 is not vintage" << endl;

    }

    if (car2.isVintage()) {

        cout << "The car 2 is vintage" << endl;

    }

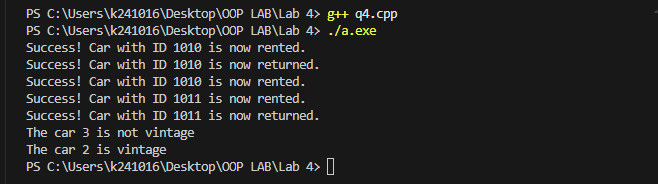
    else {

        cout << "The car 2 is not vintage" << endl;

    }

}

# Output:



Question 5:

# Source Code:

/\*

You are building an employee management system. Create a class `Employee` to manage employee

records.

Requirements:

1. Attributes:

- `employeeID` (int)

- `name` (string)

- `department` (string)

- `salary` (double)

2. Define a default constructor that initializes `salary` to `0.0`.

3. Define a parameterized constructor to initialize all attributes.

4. Add methods:

- `giveBonus(double amount)`: Adds the bonus amount to the employee's salary.

- `isManager()`: Returns `true` if the employee's department is "Management".

- `displayDetails()`: Displays the employee's details.

5. Create a few `Employee` objects and test the methods.

\*/

#include <iostream>

using namespace std;

class Employee{

    int employeeID;

    string name, department;

    double salary;

    public:

    Employee(int Employee\_ID=0, string Name="", string Department="", double Salary=0.0){

        employeeID=Employee\_ID;

        name=Name;

        department=Department;

        salary=Salary;

    }

    void giveBonus(double amount){

        salary+=amount;

        cout<<"Bonus is successfully given.\n";

    }

    bool isManager(){

        if (department=="Management") return true;

        return false;

    }

    void displayDetails(){

        cout<<"Employee Id: "<<employeeID<<endl;

        cout<<"Name: "<<name<<endl;

        cout<<"Deparment: "<<department<<endl;

        cout<<"Salary: "<<salary<<endl;

    }

};

int main(){

    Employee \*e=new Employee[5];

    e[0]=Employee(1010, "Ali", "Cleaning", 10000);

    e[1]=Employee(1011, "Ahmed", "Management", 100000);

    e[2]=Employee(1012, "Bilal", "Management", 100000);

    e[3]=Employee(1013, "Charlie", "Teaching", 50000);

    e[4]=Employee(1014, "Darwin", "Teaching", 50000);

    cout<<"Giving Bonus to Employee 1: \n";

    e[0].giveBonus(3000);

    cout<<"\nChecking if Employee 3 is a Manager:\n";

    if (e[2].isManager()) cout<<"Manager\n";

    else cout<<"Not Manager\n";

    cout<<"\nDisplaying Details:\n";

    for (int i=0; i<5; i++){

        cout<<"Employee "<<i+1<<" : \n";

        e[i].displayDetails();

        cout<<endl;

    }

    delete[] e;

}

# Output:



Question 6:

# Source Code:

/\*

Scenario: A bank wants to develop a system for managing customer accounts. The system

should allow customers to:

1. Create a new account with an account number, owner’s name, and initial balance

(default balance is 0 if not provided).

2. Deposit money into their account.

3. Withdraw money from their account, ensuring they cannot withdraw more than the

available balance.

4. Display account details including account number, owner’s name, and current balance.

Your task is to implement a C++ program that fulfills these requirements.

\*/

#include <iostream>

using namespace std;

class Account{

    string accountNumber;

    string accountHolderName;

    double balance;

    public:

    Account(string Account\_Number="", string Holder\_Name="", double Balance=0.0){

        accountNumber=Account\_Number;

        accountHolderName=Holder\_Name;

        balance=Balance;

    }

    string get\_ID(){

        return accountNumber;

    }

    void deposit(double amount){

        balance+=amount;

        cout<<"\nAmount deposited successfully"<<endl;

    }

    void withdraw(double amount){

        if (amount<=balance){

            balance-=amount;

            cout<<"\nAmount withdrawed successfully"<<endl;

        }

        else{

            cout<<"\nTransaction Failed! Insufficient Balance"<<endl;

        }

    }

    void displayDetails(){

        cout<<"Account Number: "<<accountNumber<<endl;

        cout<<"Owner Name: "<<accountHolderName<<endl;

        cout<<"Balance: "<<balance<<endl;

    }

};

class Bank{

    Account \*accounts;

    int noOfAccounts;

    public:

    Bank(){

        accounts=nullptr;

        noOfAccounts=0;

    }

    ~Bank(){

        delete[] accounts;

    }

    void newAccount(Account account){

        Account \*temp\_shifting=new Account[++noOfAccounts];

        int i, found=0;

        for (i=0; i<noOfAccounts-1; i++){

            if (accounts[i].get\_ID()==account.get\_ID()){

                found=1; break;

            }

            temp\_shifting[i]=accounts[i];

        }

        if (found==0){

            temp\_shifting[i]=account;

            delete[] accounts;

            accounts=temp\_shifting;

            cout<<"\n-----Account added successfully-----\n";

        }

        else{

            delete[] temp\_shifting;

            noOfAccounts--;

            cout<<"\n-----Error! Account must have a unique Account Number-----\n";

        }

    }

    void deposit\_money(){

        cout<<"\n-----------------\n";

        cout<<"Enter account number: ";

        string ac;

        cin>>ac;

        for (int i=0; i<noOfAccounts; i++){

            if (ac==accounts[i].get\_ID()){

                cout<<"Enter the amount you want to deposit: ";

                double temp;

                cin>>temp;

                accounts[i].deposit(temp);

                return;

            }

        }

        cout<<"Error! Account is not Found\n";

    }

    void withdraw\_money(){

        cout<<"\n-----------------\n";

        cout<<"Enter account number: ";

        string ac;

        cin>>ac;

        for (int i=0; i<noOfAccounts; i++){

            if (ac==accounts[i].get\_ID()){

                cout<<"Enter the amount you want to withdraw: ";

                double temp;

                cin>>temp;

                accounts[i].withdraw(temp);

                return;

            }

        }

        cout<<"Error! Account is not Found\n";

    }

    void display\_account\_details(){

        cout<<"\n-----------------\n";

        cout<<"Enter account number: ";

        string ac;

        cin>>ac;

        for (int i=0; i<noOfAccounts; i++){

            if (ac==accounts[i].get\_ID()){

                accounts[i].displayDetails();

                return;

            }

        }

        cout<<"Error! Account is not Found\n";

    }

};

int main(){

    string choice, id, title;

    double balance;

    Bank State\_Bank\_Of\_Pakistan;

    while (1){

        cout<<"\n=====Main Menu=====\n";

        cout<<"Press 1 to add a new account\nPress 2 to deposit money\nPress 3 to withdraw\nPress 4 to check account details\nPress any other key to exit :)\n";

        cout<<"Enter your choice: ";

        cin>>choice;

        if (choice=="1"){

            cout<<"\n----------\n";

            cout<<"Enter Account Number: ";

            cin>>id;

            cout<<"Enter Owner Name: ";

            cin.ignore();

            getline(cin, title);

            cout<<"Enter Initial Balance: ";

            cin>>balance;

            State\_Bank\_Of\_Pakistan.newAccount(Account(id, title, balance));

        }

        else if (choice=="2"){

            State\_Bank\_Of\_Pakistan.deposit\_money();

        }

        else if (choice=="3"){

            State\_Bank\_Of\_Pakistan.withdraw\_money();

        }

        else if (choice=="4"){

            State\_Bank\_Of\_Pakistan.display\_account\_details();

        }

        else{

            cout<<"\n=====Thank You for using our service=====";

            break;

        }

    }

}

# Output:

