1. Bank Account Management Create a BankAccount class that simulates a bank account. The class should have methods to deposit, withdraw, and check the account balance. Implement the necessary constructors and accessors. Write a program to demonstrate the functionality of the BankAccount class.

**Test cases:**

Custom Test Cases:

1. Test depositing and withdrawing funds from the account.
   * Deposit: $1000, Withdraw: $500
   * Expected Account Balance: $500
2. Test withdrawing more funds than the account balance.
   * Deposit: $1000, Withdraw: $1500
   * Expected Result: Error message indicating insufficient funds.

CODE:

#include <iostream>

using namespace std;

class BankAccount

{

private:

double balance;

public:

BankAccount (){

balance=0;

}

void deposit(double amount)

{

balance=balance+amount;

}

void withdraw(double amount)

{

if (amount>balance)

{

cout<<"Insufficient Balance";

}

else

{

balance=balance-amount;

}

}

double getBalance()

{

return balance;

}

};

int main() {

BankAccount acc;

int n;

double amt;

while (true)

{

amt=0;

cout<<"\nTo Deposit money press 1.\nTo Withdraw money press 2.\nTo Display balance press 3.\nTo Quit press 4."<<endl;

cin>>n;

if (n==1)

{

cout<<"Enter the Amount to deposit"<<endl;

cin>>amt;

acc.deposit(amt);

}

else if(n==2)

{

cout<<"Enter the Amount to Withdraw"<<endl;

cin>>amt;

acc.withdraw(amt);

}

else if (n==3)

{

cout<<"Balance is:"<<acc.getBalance()<<endl;

}

else if(n==4)

{

break;

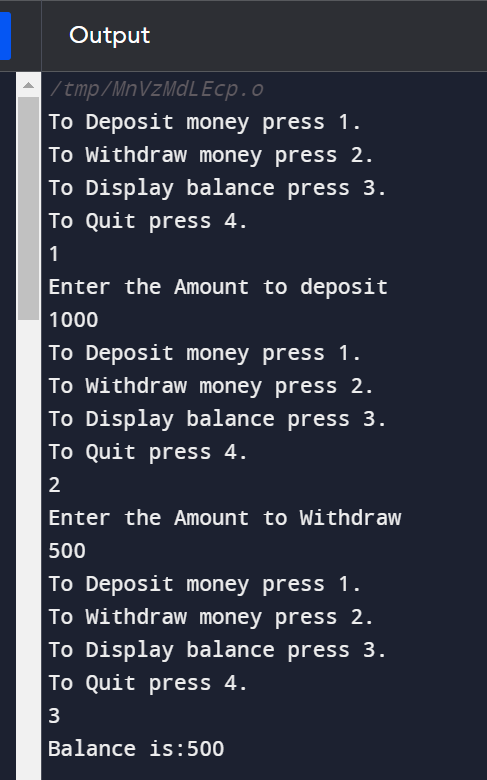
}

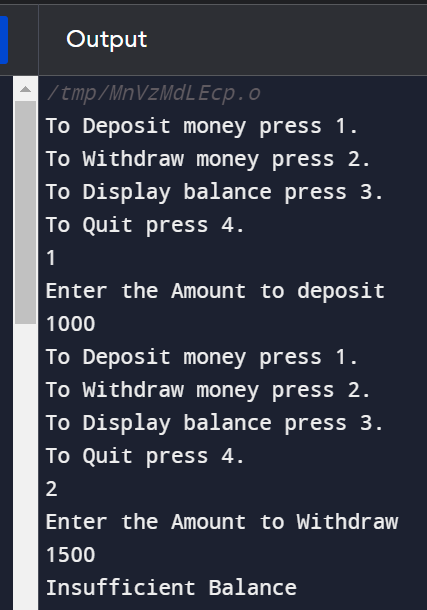
}

return 0;

}

OUTPUT:

Testcase 1

Testcase 2

1. Question: Create a class called Stack that represents a stack data structure. It should have methods push(), pop(), and isEmpty() to add elements, remove elements, and check if the stack is empty, respectively. Implement the stack using an array. Write a program that demonstrates the functionality of the Stack class by performing push and pop operations.

**Test cases:**

Stack Operations:

1. Push

2. Pop

3. Exit

Enter your choice: 1

Enter element to push: 10

Pushed element: 10

Stack Operations:

1. Push

2. Pop

3. Exit

Enter your choice: 1

Enter element to push: 20

Pushed element: 20

Stack Operations:

1. Push

2. Pop

3. Exit

Enter your choice: 2

Popped element: 20

Stack Operations:

1. Push

2. Pop

3. Exit

Enter your choice: 2

Popped element: 10

Stack Operations:

1. Push

2. Pop

3. Exit

Enter your choice: 2

Stack is empty. Cannot pop element.

CODE:

#include <iostream>

using namespace std;

class Stack {

private:

static const int MAX\_SIZE = 100;

int data[MAX\_SIZE];

int top;

public:

Stack() {

top = -1;

}

bool isEmpty() const {

return top == -1;

}

bool isFull() const {

return top == MAX\_SIZE - 1;

}

void push(int element) {

if (isFull()) {

cout << "Stack is full. Cannot push element." << endl;

} else {

top++;

data[top] = element;

cout << "Pushed element: " << element << endl;

}

}

void pop() {

if (isEmpty()) {

cout << "Stack is empty. Cannot pop element." << endl;

} else {

int poppedElement = data[top];

top--;

cout << "Popped element: " << poppedElement << endl;

}

}

};

int main() {

Stack stack;

int choice;

int element;

do {

cout << "Stack Operations:" << endl;

cout << "1. Push" << endl;

cout << "2. Pop" << endl;

cout << "3. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter element to push: ";

cin >> element;

stack.push(element);

break;

case 2:

stack.pop();

break;

case 3:

cout << "Exiting program." << endl;

break;

default:

cout << "Invalid choice. Please try again." << endl;

break;

}

cout << endl;

} while (choice != 3);

return 0;

}

OUTPUT:

