Lab 3

Objective: Practice **top-down design**, **call-by-reference parameters**.

In this lab, we design a program in which the user can deposit money, withdraw money, and check their current balance. We use top-down design to decompose the problem, and employ call-by-reference parameters for updating the balance after each transaction.

The system should offer a menu for the user to choose different options and execute the respective functions.

Step 1: Top-Down Design Breakdown

1. **Main Function**
   The main function should:
   a. Display a menu to the user
   b. Call the appropriate sub-functions based on the user's choice:
   - Deposit money
   - Withdraw money
   - Check balance
   - Exit the program
2. **Functions**
   - `displayMenu`: Display the available operations.
   - `depositMoney`: Accept a deposit amount from the user and update the balance (uses call-by-reference).
   - `withdrawMoney`: Accept a withdrawal amount from the user and update the balance (uses call-by-reference).
   - `checkBalance`: Display the current balance (call-by-value).
   - `processChoice`: Execute the user's choice and call the relevant function.

Step 2: Explanation of Key Features

1. **Top-Down Design**
   The problem is divided into distinct tasks such as displaying the menu, processing deposits and withdrawals, checking balance, and exiting. The `main` function calls these tasks in a sequence based on the user's choice.
2. **Call-by-Reference Parameters**
   - `depositMoney` and `withdrawMoney` use **call-by-reference** to modify the user's `balance` directly within the functions. This allows the balance to be updated after each transaction.
   - `checkBalance` uses **call-by-value** since it only needs to display the balance and does not modify it.
3. **Function Design**
   Each task has a dedicated function, making the code modular and easy to maintain.

Step 3: Code Implementation

Please finish the code in the following functions: `depositMoney`, `withdrawMoney`, `processChoice`.

```cpp
#include <iostream>
using namespace std;

// Function declarations
void displayMenu();
void
void
void processChoice(int choice, double &balance);

int main() {
    double balance = 0.0;
    int choice;

    do {
        displayMenu();
        cout << "Enter your choice: ";
        cin >> choice;
        processChoice(choice, balance);
    } while (choice != 4);  // 4 = Exit

    return 0;
}

// Function to display the menu
void displayMenu() {
    cout << "\n--- Bank Account Menu ---\n";
    cout << "1. Deposit Money\n";
    cout << "2. Withdraw Money\n";
    cout << "3. Check Balance\n";
    cout << "4. Exit\n";
}

```

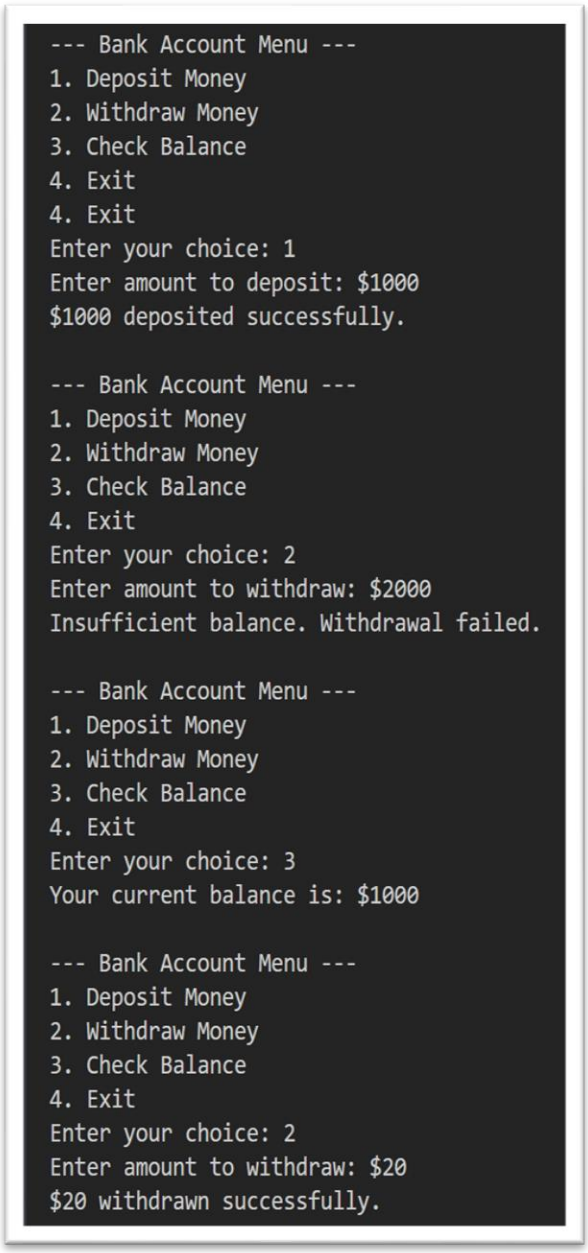
void processChoice(int choice, double &balance)

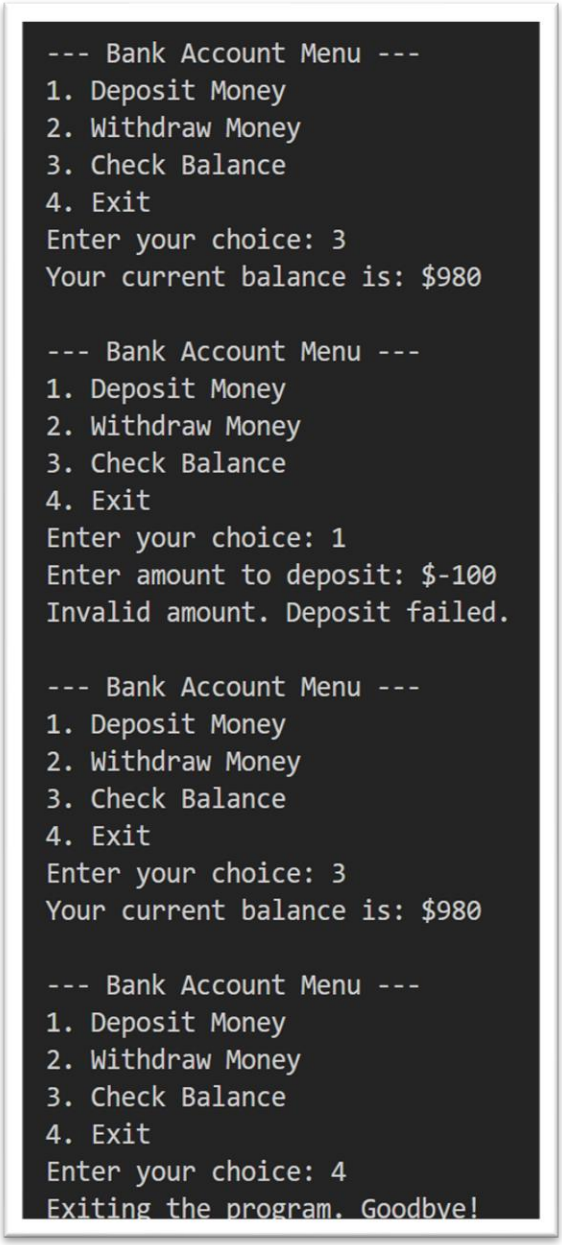
Function to process the user's choice

```
// Function to deposit money (call-by-reference to update balance)
void depositMoney(double &balance) {
    double amount;
    cout << "Enter amount to deposit: $";
    cin >> amount;
    //Please put code here to finish the function
    //if the amount entered is greater than 0, update the balance
    //and display the deposit is made successfully
    //otherwise announce invalid amount, deposit failed.
}

// Function to withdraw money (call-by-reference to update balance)
void withdrawMoney(double &balance) {
    double amount;
    cout << "Enter amount to withdraw: $";
    cin >> amount;

    //Please put code here to finish the function
    //if the amount entered is greater than 0 and less than the balance
    //update the balance, display the withdrawal is made successful
    //if the withdrawl amount is greater than the balance
    //do not update the balance and display the insufficient fund message
    //otherwise announce invalid amount, deposit failed

}

// Function to check balance (call-by-value since we don't need to modify it)
void checkBalance(double balance) {
    cout << "Your current balance is: $" << balance << endl;
}

```

```
 // Function to process the user's choice
 void processChoice(int choice, double &balance) {
     switch (choice) {
         case 1:
             // call depositMoney()function;
             break;
         case 2:
             // call withdrawMoney() function;
             break;
         case 3:
             //call checkBalance() function;
             break;
         case 4:
             cout << "Exiting the program. Goodbye!\n";
             break;
         default:
             cout << "Invalid choice. Please try again.\n";
     }
 }
```

Step 4: Example Output:





Step 5: Submission:

You need to upload 2 files on Moodle before 4pm Wednesday October 2, 2024: the source code (.cpp file) and screen shot of your program output.