Banking System Assignment Instructions

This assignment will help you practice **Object-Oriented Programming (OOP)** concepts, specifically working with **classes** and **abstract data types (ADTs)**. You will implement a **Banking System** that allows users to manage accounts, perform transactions, and display account information.

Objective

1. Understand the concept of **classes** in C++.
2. Learn to use **constructors**, **accessor** and **mutator** functions, and implement **data abstraction**.
3. Implement a menu-driven program to apply the above concepts.

Program Requirements

1. **Create a Class for Bank Accounts**:
    - The class should include:
        - Private data members:
            - int accountNumber (to store a unique account number).
            - std::string accountHolderName (to store the account holder's name).
            - double balance (to store the account balance).
        - Public member functions:
            - Constructors (default and parameterized).
            - Accessor (getter) and mutator (setter) functions.
            - Functions to **deposit** and **withdraw** money.
            - A function to display account details.
2. **Create a Class for the Banking System**:
    - The class should manage multiple accounts using an array.
    - Include:

        - An array of **BankAccount** objects (size = 100).
        - A counter to track the number of accounts.
        - Member functions to:
            - Add a new account by taking user input.
            - Perform transactions (deposit or withdraw) on a specific account.
            - Display details of all accounts.

3. **Implement a Menu-Driven Interface**:
    - Allow the user to choose from the following options:

1. Add Account: Create a new account by providing account number, holder's name, and initial balance.

2. Perform Transaction: Deposit or withdraw money by specifying the account number and amount.

3. Display All Accounts: List all the accounts with details (account number, holder name, balance).

4. Exit: Exit the program.

4. **User Input**:
    - Account details (number, holder name, balance) should be provided by the user through the console.
    - Validate inputs for correctness (e.g., balance cannot be negative, transactions cannot exceed the account balance).

Constraints

1. The program should not use **pointers** or **vectors**.
2. Limit the total number of accounts to **100**.
3. Transactions should handle:
    - Validating the account number.
    - Preventing negative or zero amounts.

    - Ensuring withdrawals do not exceed the current balance.

4. Use **default and parameterized constructors** for account initialization.

Steps to Complete

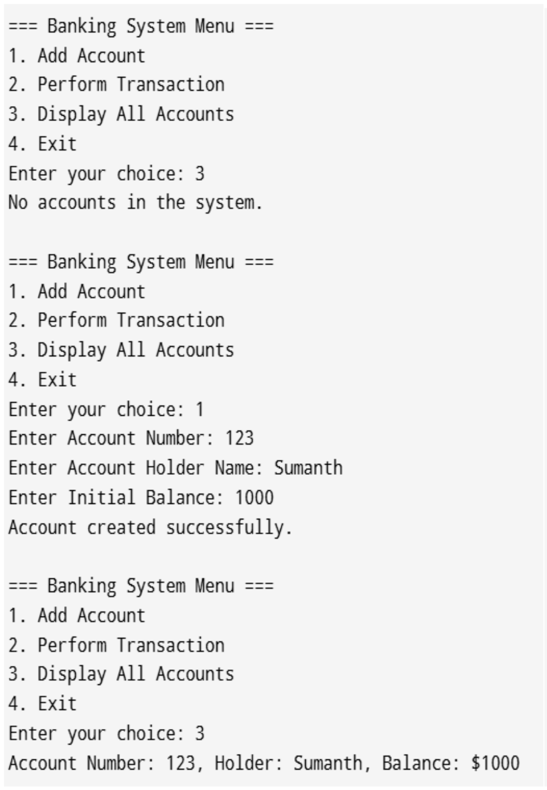
1. **Design and Write the Code**:
    - Create the BankAccount class with necessary member variables and functions.
    - Create the BankSystem class to manage accounts.
    - Implement a main() function with a menu-driven interface.
2. **Test Your Program**:
    - Add at least 3 accounts.
    - Perform multiple transactions (deposit and withdraw).
    - Display account details to verify functionality.
3. **Document Your Code**:
    - Use comments to explain the purpose of each class, function, and key code sections.

Submission Instructions

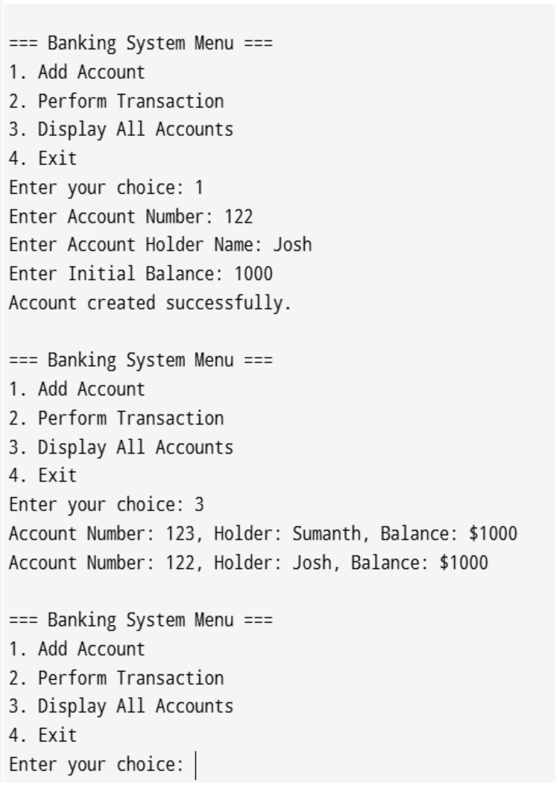
1. Save your program in a file named .cpp.
2. Provide the following in your submission:
    - The complete code with comments.
    - A sample output of the program execution for the following:
        - Adding accounts.
        - Performing deposit and withdrawal transactions.
        - Displaying all accounts.
3. Submit the assignment by the deadline through your designated platform.

Grading Criteria

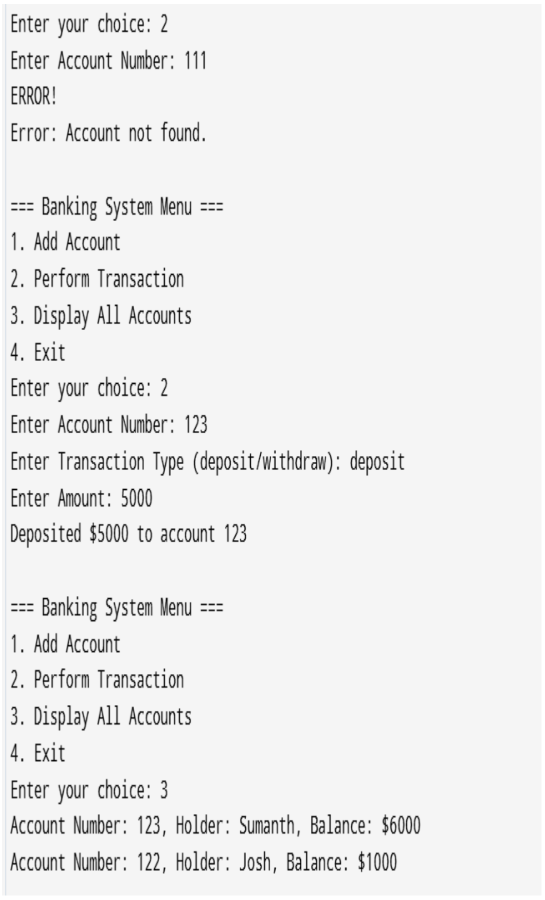
- **Code Completeness (40%)**:
    - Implementation of required classes, functions, and menu options.
- **Correctness (30%)**:
    - Proper handling of input and edge cases (e.g., invalid account number, overdrafts).
- **Code Organization (20%)**:
    - Use of clean and modular code with meaningful function names and comments.
- **Output (10%)**:
    - Accurate and user-friendly output



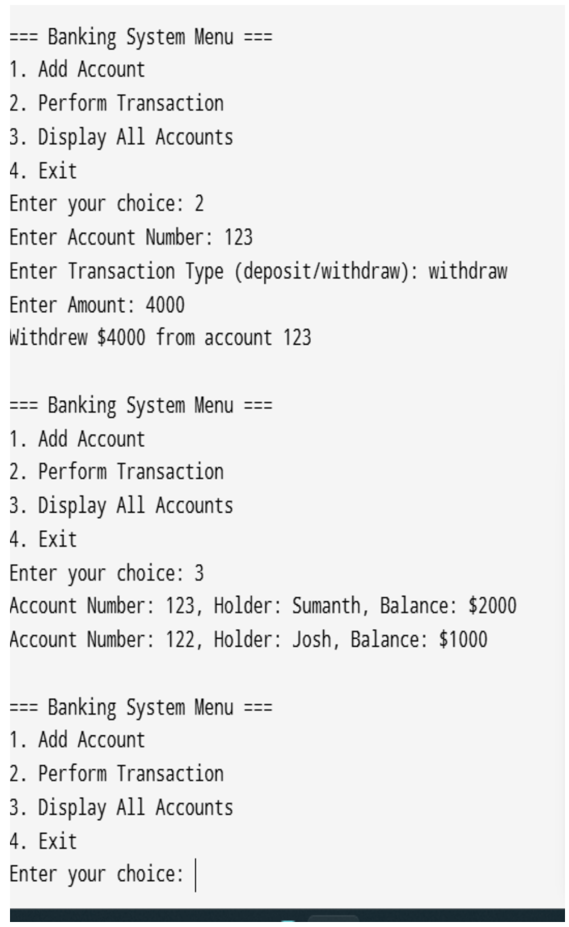
```
=== Banking System Menu ===
1. Add Account
2. Perform Transaction
3. Display All Accounts
4. Exit
Enter your choice: 3
No accounts in the system.

=== Banking System Menu ===
1. Add Account
2. Perform Transaction
3. Display All Accounts
4. Exit
Enter your choice: 1
Enter Account Number: 123
Enter Account Holder Name: Sumanth
Enter Initial Balance: 1000
Account created successfully.

=== Banking System Menu ===
1. Add Account
2. Perform Transaction
3. Display All Accounts
4. Exit
Enter your choice: 3
Account Number: 123, Holder: Sumanth, Balance: $1000
```



```
=== Banking System Menu ===
1. Add Account
2. Perform Transaction
3. Display All Accounts
4. Exit
Enter your choice: 1
Enter Account Number: 122
Enter Account Holder Name: Josh
Enter Initial Balance: 1000
Account created successfully.

=== Banking System Menu ===
1. Add Account
2. Perform Transaction
3. Display All Accounts
4. Exit
Enter your choice: 3
Account Number: 123, Holder: Sumanth, Balance: $1000
Account Number: 122, Holder: Josh, Balance: $1000

=== Banking System Menu ===
1. Add Account
2. Perform Transaction
3. Display All Accounts
4. Exit
Enter your choice:
```



```
Enter your choice: 2
Enter Account Number: 111
ERROR!
Error: Account not found.

=== Banking System Menu ===
1. Add Account
2. Perform Transaction
3. Display All Accounts
4. Exit
Enter your choice: 2
Enter Account Number: 123
Enter Transaction Type (deposit/withdraw): deposit
Enter Amount: 5000
Deposited $5000 to account 123

=== Banking System Menu ===
1. Add Account
2. Perform Transaction
3. Display All Accounts
4. Exit
Enter your choice: 3
Account Number: 123, Holder: Sumanth, Balance: $6000
Account Number: 122, Holder: Josh, Balance: $1000
```



```
=== Banking System Menu ===
1. Add Account
2. Perform Transaction
3. Display All Accounts
4. Exit
Enter your choice: 2
Enter Account Number: 123
Enter Transaction Type (deposit/withdraw): withdraw
Enter Amount: 4000
Withdrew $4000 from account 123

=== Banking System Menu ===
1. Add Account
2. Perform Transaction
3. Display All Accounts
4. Exit
Enter your choice: 3
Account Number: 123, Holder: Sumanth, Balance: $2000
Account Number: 122, Holder: Josh, Balance: $1000

=== Banking System Menu ===
1. Add Account
2. Perform Transaction
3. Display All Accounts
4. Exit
Enter your choice:
```