Quiz 2 (in-class)

- Due Oct 13 at 12:20pm
- Points 15
- Questions 1
- Available Oct 13 at 11:45am - Oct 13 at 12:25pm 40 minutes
- Time Limit 25 Minutes

Instructions

Due in class on Oct. 13

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	15 minutes	0 out of 15 *

* Some questions not yet graded

Correct answers are hidden.

Score for this quiz: 0 out of 15 *
* Some questions not yet graded
Submitted Oct 13 at 12:02pm
This attempt took 15 minutes.

Question 1
Not yet graded / 15 pts

Learning Objective

- Implement member functions of a class to perform tasks based on given logics, rules, and algorithms.

Problem Description

You are given the task of developing a system for a well-known bank to automate its incoming transactions associated with all accounts, including money transfers between two accounts, and deposits/withdrawals into an account. The bank operates with n accounts, where each account is identified by a number ranging from 1 to n. The starting balance for all accounts are stored in a 0-indexed array called `balance`, where the $(i+1)^{th}$ account is represented by the balance in the amount of `balance[i]`.

The goal is to process all valid transactions. A transaction is valid if the following conditions are met:

- The specified account number(s) fall within the range of 1 to n.
- The amount being withdrawn or transferred from an account does not exceed the available balance in the account.

You are required to implement a `Bank` class with the following functionality:

- `Bank(vector<long long>& balance)`: This constructor initializes the bank with the provided array `balance`, representing the starting balances of all accounts.
- `bool transfer(int account1, int account2, long long money)`: This method transfers the specified `money` from `account1` to `account2`. It should return `true` if the transaction is successful, and `false` otherwise.

  item `bool deposit(int account, long long money)`: This method adds the given amount of `money` to the balance of `account`. It returns `true` if the deposit is successful, otherwise it returns `false`.
- `bool withdraw(int account, long long money)`: This method deducts the specified `money` from `account`. It returns `true` if the withdrawal is successful, otherwise it returns `false`.
- You can think of the data type `long long` the same as data type `int` syntax-wise, but with the ability to store much larger integer values.

Input

The initial balance of each account is given in a 0-indexed integer array `balance`. You will also receive a series of operations that need to be executed, such as transfer, deposit, or withdrawal.

Output

For each operation, return `true` if the transaction was successful or `false` otherwise.

Example 1

Input:

```
["Bank", "withdraw", "transfer", "deposit", "transfer", "withdraw"]
[[[10, 100, 20, 50, 30]], [3, 10], [5, 1, 20], [5, 20], [3, 4, 15], [10, 50]]
```

Output:

```
[null, true, true, true, false, false]
```

Explanation

```
Bank bank = new Bank([10, 100, 20, 50, 30]);
bank.withdraw(3, 10);   // return true, account 3 has a balance of $20, valid to withdraw $10.
bank.transfer(5, 1, 20); // return true, account 5 has a balance of $30, valid to transfer $20.
bank.deposit(5, 20);     // return true, valid to deposit $20 to account 5.
bank.transfer(3, 4, 15); // return false, account 3 has a balance of $10, cannot transfer $15.
bank.withdraw(10, 50);   // return false, account 10 does not exist.
```

Requirements

1. The solution to this problem must be implemented using C++ only. No other programming languages (including Python, Java, Matlab, R, etc.) are allowed.
2. Please respect the honor code: you are not allowed to use the internet, including ChatGPT or any other AI tools, to assist in the implementation of this problem. Otherwise, points will be deducted.
3. Do not change the framework provided to you. You may only write your solution within the comments between `/* Your code for the [function_name] starts here */` and `/* Your code for the [function_name] ends here */`.
4. The main function is already provided to you and the initialization function is also implemented for you. Your task is to implement the three remaining functions (i.e., the transfer, deposit and withdraw function).

```
//You can copy/paste the partial code below into an IDE/editor and work out the solution based on

//the following partial code.

//****** partial code *****

#include <iostream>

#include <vector>

using namespace std;
```

```
class Bank {
public:
    vector<long long>& balance_ref;

   // Initialize the Bank with an array of account balances by reference
   Bank(vector<long long>& balance) : balance_ref(balance) {
   }

   // Transfers money from account1 to account2
   bool transfer(int account1, int account2, long long money) {
      /* Your code for the transfer function starts here */



      /* Your code for the transfer function ends here */
   }

   // Deposits money into the specified account
   bool deposit(int account, long long money) {
      /* Your code for the deposit function starts here */



      /* Your code for the deposit function ends here */
   }

   // Withdraws money from the specified account
   bool withdraw(int account, long long money) {
      /* Your code for the withdraw function starts here */



      /* Your code for the withdraw function ends here */

   }


};


int main() {

   vector<long long> balance = {10, 100, 20, 50, 30};

   Bank bank(balance);


   cout << bank.withdraw(3, 10) << endl;  // Output: true

   cout << bank.transfer(5, 1, 20) << endl;  // Output: true

   cout << bank.deposit(5, 20) << endl;  // Output: true

   cout << bank.transfer(3, 4, 15) << endl;  // Output: false

   cout << bank.withdraw(10, 50) << endl;  // Output: false

   return 0;

}
```
Your Answer:
```
#include <iostream>
#include <vector>

using namespace std;

class Bank {
public:
vector<long long>& balance_ref;

Bank(vector<long long>& balance) : balance_ref(balance) {
}
```

```
bool transfer(int account1, int account2, long long money) {
const int n = static_cast<int>(balance_ref.size());
if (account1 < 1 || account1 > n || account2 < 1 || account2 > n) {
return false;
}
const int from = account1 - 1;
const int to = account2 - 1;
if (balance_ref[from] < money) {
return false;
}
balance_ref[from] -= money;
balance_ref[to] += money;
return true;
}

bool deposit(int account, long long money) {
const int n = static_cast<int>(balance_ref.size());
if (account < 1 || account > n) {
return false;
}
balance_ref[account - 1] += money;
return true;
}

bool withdraw(int account, long long money) {
const int n = static_cast<int>(balance_ref.size());
if (account < 1 || account > n) {
return false;
}
const int index = account - 1;
if (balance_ref[index] < money) {
return false;
}
balance_ref[index] -= money;
return true;
}
};

int main() {
vector<long long> balance = {10, 100, 20, 50, 30};
```

```
Bank bank(balance);

cout << bank.withdraw(3, 10) << endl; // Output: true
cout << bank.transfer(5, 1, 20) << endl; // Output: true
cout << bank.deposit(5, 20) << endl; // Output: true
cout << bank.transfer(3, 4, 15) << endl; // Output: false
cout << bank.withdraw(10, 50) << endl; // Output: false
return 0;
}
```

Quiz Score: 0 out of 15

* Some questions not yet graded