HW #3 (Banking)

- In this assignment, you will use C++, or Java if you prefer, to write an object-oriented application for a simple banking program.

- The application should allow a user to check their balance, deposit a given amount of money, withdraw a given amount of money, transfer money to another user account, and list the transactions performed by the user.

HW #3 (2)

The task

- In this assignment, you are given the specification of the banking application which you need to implement. The application consists of the following classes:
- A Account class representing a user ID and the account balance.
- A User class representing the username and Account for a given user (customer). No pair of users share the same username.
- A Transaction class representing a log of transactions/operations performed by a user.

HW #3 (3)

- Following is an example test driver program. However, there is **NO** guarantee the code is completely correct. Kindly modify the test driver,

  as well as the class definitions, if necessary.

- Particularly, you are allowed to add the word "const", "&" (for reference), "=" (for default argument), "friend", "static", "private", "public", or others, whenever you consider it is appropriate.
- Moreover, for grading purpose, it is strictly requires that everyone submit their own test driver.

```
int main() {
    User u1(“john”);   // should print: New user john created
    User u2(“mary”); // should print: New user mary created

    Account a1 = u1.getAccount();
    a1.deposit(400);
    a1.withdraw(100);
    cout << “Balance of “ << u1.getUsername() << “ account is “ << a1.getAmount()
          << endl; // 300 = 400 - 100

    a1.transferMoney(u2.getAccount(), 200);
    cout << “Balance of “ << u1.getUsername() << “ account is “ << a1.getAmount()
          << endl; // 100 = 300 - 200
    cout << “Balance of “ << u2.getUsername() << “ account is “
          << u2.getAccount().getAmount() << endl; // 200 = 0 + 200

    u1.getTrans(); for-loop
                     print(); // should print: Type: Create
                                             // Type: Deposit 400
                                             // Type: Withdraw 100
                                             // Type: Transferred 200 to mary
}
```

HW #3 (4)

The classes

```
// This class implements a user account, represented by
// an unique user ID and an amount
1. class Account {
    // ID: an integer representing the account (common field
    // between Account and User)
    // amount: the account balance

// declare your own variables and functions, if needed
```

HW #3 (5)

```
// constructor with parameters
Account(int amount, int ID);

// Withdraw a given amount of money from the account
// and record the transaction
    // deductAmount: the amount to withdraw
    // return true if the withdraw succeeds, false otherwise
bool withdraw(int deductAmount);
```

HW #3 **(6)**

```
// Deposit a given amount to the account and
// record the transaction
    // addAmount: amount to be deposited
    // always return true
bool deposit(int addAmount);
```

HW #3 (7)

```
// Transfer money from this account to user B’s account,
// and record the transaction
    // AccountOfB: account B to transfer money to
    // amountToTransfer: the amount to transfer
    // return true if the transfer is possible, false otherwise
bool transferMoney(Account &AccountOfB,
                                   int amountToTransfer);

int getAmount(); // return amount
}; // end of Account class
```

HW #3 (8)

```
// This class represents the data for a user of the bank
2. class User {
    // name: a string representing the username
    // ID: an integer representing the user (common field
    // between User and Account)
    // account: an Account representing the account
    // trans[100]: keep track of transactions associated with
    // this User; you can assume there will NOT be more
    // than 100 elements for the array of transactions

// declare your own variables and functions, if needed
```

HW #3 (9)

```
// Constructor with parameter
// Create a new account and record the "transaction“
// Each user must be assigned a new ID
User(const char &name[]);

char *getUsername();     // Return the username
Account &getAccount(); // Return the user Account
int getID();                          // Return the user ID

}; // end of User class
```

HW #3 (10)

```
// This class implements a transaction performed by the
// bank users
3. class Transaction {
    // type: a string to record the type of transaction
    // account: an Account used by the transaction

// declare your own variables and functions, if needed

// constructor with parameters
Transaction(Account &account, char type[]);
```

HW #3 (11)

```
void print(); // Output the details of transaction

}; // end of Transaction class
```