Rubric For Project 1

In all the testcases, we will first start the server by `./server [ipAddr] [port]` (assume the executable file for server is `server`). Note that, the support for parameters `ipAddr` (ip address) and `port` (port number) is optional, you can hardcode them in your server program. And during tests, if the corresponding port is occupied (Don't Worry), we will change to another available one.

In the following tests, we assume that server is running (on the same machine as client) and client executable file is `client`.

For ease of grading, we fix the arguments of your client program. That is,

```
./client CMD [testaccount] [amount] ipAddr port
# CMD is one of BAL|WITHDRAW|TRANSFER
# testaccount is one or two (for TRANSFER) of myChecking|mySavings|myCD|my401k
# |my529 for WITHDRAW and TRANSFER, [amount] follows the acount name
# idAddr is the ip address for the sever (e.g., 127.0.0.1)
# port is the port number (e.g., 2019)
```

Note that, you can assume that we will always input valid arguments according to your requirements, though you are recommended to include error checking in case any of these arguments are missing or incorrectly formatted.

Implementation Based on TCP (1 point)

Please make sure you use TCP instead of UDP.

Balance Tests (3 points)

We will test balance function by using :

```
./client BAL testaccount ipAddr port
# testaccount is one of myChecking|mySavings|myCD|my401k|my529
```

Expected Outcome: An integer to indicate the amount of balance of the corresponding account.

Rubric

We might randomly test some or all of the five accounts.

- all fail ==> 0 point
- partially work ==> 2 point
- all work ==> 3 points

Withdraw Tests (2 points)

We will test withdraw function by the following steps.

Step 1

```
./client WITHDRAW testaccount amount ipAddr port
# testaccount is one of myChecking|mySavings|myCD|my401k|my529
```

You should tell the user whether withdraw succeeds (e.g., print out a message "Withdraw Succeeded!").

Expected Outcome: A message that tells the user withdraw succeeds or fails.

Step 2

You will check the balance of the corresponding account again.

Expected Outcome: The balance should "change" accordingly.

Rubric

We might randomly test some or all of the five accounts.

- all fail ==> 0 point
- first step succeeds, second fails ==> 1 point
- all work ==> 2 points

Transfer Tests (2 points)

We will test transfer function by the following steps.

Step 1

```
./client TRANSFER from-account to-account amount ipAddr port
# from-account/to-account is one of myChecking|mySavings|myCD|my401k|my529
```

You should tell the user whether transfer succeeds (e.g., print out a message "Transfer Succeeded!").

Expected Outcome: A message that tells the user transfer succeeds or not.

Step 2

You will check the balances of the corresponding accounts again.

Expected Outcome: The balances should "change" accordingly.

Rubric

We might randomly test some or all of the five accounts.

- all fail ==> 0 point

- first step succeeds, second fails ==> 1 point
- all work ==> 2 points

Account Protection Test (2 points)

In this part, we will randomly generate 10 testcases (5 timeouts, 5 not) (Do not worry about the boundary cases!).

An example of the possible testcases, where we should not see "Error: too many withdrawals in a minute!":

```
./client WITHDRAW myChecking amount ipAddr port
# after 5 seconds
./client WITHDRAW myChecking amount ipAddr port
# after 5 seconds
./client WITHDRAW myChecking amount ipAddr port
# after 10 seconds
./client WITHDRAW myChecking amount ipAddr port
# after 120 seconds
./client WITHDRAW myChecking amount ipAddr port
```

An example of the possible testcases, where we should see "Error: too many withdrawals in a minute!":

```
./client WITHDRAW myChecking amount ipAddr port
# after 5 seconds
./client WITHDRAW myChecking amount ipAddr port
# after 5 seconds
./client WITHDRAW myChecking amount ipAddr port
# after 10 seconds
./client WITHDRAW myChecking amount ipAddr port
# after 5 seconds
./client WITHDRAW myChecking amount ipAddr port
```

Rubric

- all fail ==> 0 point
- partially succeed ==> 1 point
- all work ==> 2 points

Notice

1. **You are suggested to test all the above functionalities on the shuttle server before trun in. We will test your codes on them!!**
2. **You are suggested to provide a README file (any format would be OK, e.g., README.txt) to tell TAs how to run your code.**
3. **Please put some comments in your codes!**