CSCE 4753 Computer Networks

Automated Teller Machine (ATM) Socket Program

30 points

1. This TCP socket programming assignment will involve the idea of modern-day banking using an automated teller machine (ATM). The server-side software maintains a bank account and a user accesses the account via client software. For this program, you can assume that a single account will be accessed by a single user. This avoids synchronization problems with multiple users. There must be two sides to this program: a server and a client. The server maintains the balance and does all updates (withdrawals and deposits) and also responds to queries on the amount of money in the account. The user issues commands via the client to check the available balance, withdraw money from the account, and deposit money into the account. The client does none of the calculations but only sends requests to the server that executes the commands and returns the answer to the client. The server should initialize the balance in the bank account to $100.

2. The user must be able to do the following activities from the client software:

- Deposit money into the account on the server.
- Withdraw money from the account on the server.
- Check the balance of the amount in the account on the server.

3. Instructions

- The programs MUST be written in the programming language ***Python 3***.
- Design the two programs to be executed on the command line, not with a graphic user interface.
- Name the server program "server.py" and the client program "client.py".
- **The programs will be tested on turing.uark.edu so make sure that they work there before turning them in.** On turing, Python 3 is implemented by the command "python3".
- Upload both programs to Blackboard before the due date and include instructions on how to run it. Therefore, there should be three files named server.py, client.py, and readme.txt. All three must be readable with a text editor.
- Put both your name and UA ID in all three files as a comment.

4. Socket Programming Rubric (This is how it will be graded.)

- If the server or client program does not run, -25 pts.
- If the server or client crashes during normal interactions, -10 pts.
- If the client processes the deposit, withdrawal, or balance function instead of the server, -15 pts.
- If a user can withdraw more money than is in the account, -5 pts.
- If the balance function does not work, -5 pts.
- If the withdraw function does not work, -5 pts.
- If the deposit function does not work, -5 pts.

- If the program does not perform general error checking on the input, -5 pts.
- If the server logs you out after one transaction instead of being persistent, -5 pts.
- If the account balance is not persistent across multiple connections, -5 pts.
- If there is no while loop in menu, causing the user to execute the client.py a separate time for each action, -5 pts.
- If Python 2.6/2.7 is used instead of Python3, -5 pts.
- If the program accepts negative or float numbers, -5 pts.
- If you cannot withdraw ALL of the current balance to make it = 0, -5 pts.

Typical grading process but other processes may be used:

- Start by withdrawing $100.
- Deposit $100.
- Check the balance.
- Exit if there.
- Check to see if you can withdraw -$100.
- Check to deposit $100.50.
- Withdraw all of the current balance to make the account equal to 0.
- Try to withdraw more than the current balance (overdraft).
- Try to enter character instead of integer in amount (withdraw/deposit).