CPSC 3500

Homework 6: Client-Server Programming (Are You Psychic!)

Due: June 2 (11:59 PM), Tuesday (on Canvas)

Description (Total points: 100)

NOTE: **Complete this assignment in C/C++. Please submit your work by May 5 on Canvas. This assignment is to be done individually; you can discuss the problem with your classmates, but you should code your own solutions independently. This programming assignment would be 25% of the total assignment grade across all homeworks. Your code should compile and run (as expected) on cs1.seattleu.edu.**

Directions:

1. Your programs will primarily be evaluated for functionality. The greater the number of test cases they pass, the greater will be the score.
2. Partial credit can be given. However, a program failing several test cases will receive a very low grade.
3. **No changes to the submitted programs** (however minor) will be accepted after the submission. Please test your programs thoroughly before submission and ensure that you are submitting the best version.
4. **Your programs will be evaluated on the CS1 server.** Please ensure they compile and run as expected on the CS1 server. Programs that fail to compile or run on the CS1 server will receive a zero.
5. Testing and debugging are integral part of programming. Before submission, you are responsible for creating test cases and testing your programs.
6. Sharing of code in any form is strictly prohibited. You can share the test cases, though.

In this assignment, you will be creating a client and a server program to play a simple number guessing game (***Are You Psychic!***)). Client programs can connect to the server to play the game. Clients send their guess to the server. The server responds by sending the current game status back to the client.

The game requires user to guess a three digit number (000 to 999).

Here is the process to play the game:

1. The player (at the client) is prompted to enter the player's name and is sent to the server.
2. The three digit number is selected randomly by the server.
3. The player enters a guess.
4. The client makes sure the guess is valid (in the range 000 to 999) and sends the guess to the server.
5. The server first checks if the player has successfully guessed the number. If so, go to step 8.
6. If the guess was not correct, the server responds to the client with either "Too high" or "Too low."

    - Example: Assume the mystery number is 207. The guess 381 would result in a response of "Too high"
    - Example: Assume the mystery number is 207. The guess 180 would result in a response of "Too low"
    - Example: Assume the mystery number is 207. The guess 207 would result in a response of "Correct guess!"
7. Repeat steps 3-6 until the game is won.
8. When the game is won, the server will respond with a victory message that indicates the number of turns it took to guess the mystery number.
9. In addition, the server will send a leader board indicating the name of the top three players along with the number of turns it took to guess the mystery number.

Are You Psychic! Client

Here is a more detailed description of the client:

1. Starting the client will have two *required* command line arguments:
    - IP address of the server
    - port of the server process
2. Connect with the server. Exit with an appropriate error message if a connection could not be established.
3. Ask the user for their name.
4. Send the name to the server.
5. Ask the user for a guess. A guess consists of an integer.
6. Check if the guess is valid (the number is in the range from 0 to 999). If the guess is invalid, print an error message and go back to step 5.
7. Send the guess to the server.
8. Receive the result of the guess from the server.
9. If the guess was unsuccessful, print the result of the guess to the screen and go back to step 5.

Note: The remaining steps are only carried out if the guess was successful.

10. Receive an additional victory message from the server.
11. Print the victory message to the screen.
12. Receive leader board information received from the server.
13. Print leader board information to the screen.
14. Close the connection with the server.

Are You Psychic! Server

Here is a more detailed description of the server:

1. Starting the server will have one *required* command line argument:
    - port number (on which the server listens)
2. Listen for a client.
3. When a client connects, create a new thread to process that client (see below).
4. The server will run forever until aborted (Ctrl-C).

The following steps correspond how to process each client:

1. Generate a random number (between 0 to 999) and print the number on the screen (while this is not necessary for actual gameplay, it helps debugging and grading).
2. The first communication step is to receive the player's name from the client.
3. Receive a guess of number from the client.
4. If the guess is not successful, send the result of the guess "Too high" or "Too low" (as described previously) and go back to step 3.

Note: The remaining steps are only carried out if the guess was successful.

5. Send the result of the guess ("Correct guess!").
6. Send a victory message indicating the number of turns it took.
7. Update the leader board if the player made the top three.
8. Send the current leader board to the client.
9. Close the connection with the client.

Leader Board

A few notes about the leader board:

- When the server is started, the leader board initially starts empty.
- If the leader board contains fewer than three entries, only display the current number of entries.
- Ties are broken by whoever completed the game first as determined by when the leader board is updated at the end of the game.
    - For example, let's say the following people played in the following order from earliest to latest with number of turns in parentheses: Aadya (19), Bryn (15), Mark (19), Neal (15). Then the leader board would be:

      1. Bryn 15

      2. Neal 15

      3. Aadya 19

    - Do not make this more complicated than necessary. This rule actually makes tie handling very simple.

- The output of the client should display each person on the leader board on its own line with each line displaying the user name and the number of turns it took (as shown in the example illustrating ties above.)
- Did you realize that the data structure storing the leader board is shared among the different client threads? Your program should protect against any issues that may arise from sharing the leader board. (Use appropriate synchronization primitives such as locks or semaphores.)

Additional Requirements and Assumptions

- You may assume the user's name will not be longer than 100 characters.
- The guess sent from the client to the server must be sent as an integer. The result returned from the server must be sent as a string.
- All integers must be sent in network order.
- Since the server runs forever, it must be robust. In particular:
    - Any error that occurs when processing a client causes only that thread to abort. These types of errors should not cause the server as a whole to crash.
    - The server should be free of leaks – dynamically allocated resources need to be reclaimed.
- If a socket function that creates / uses the listening socket fails, abort the server with an appropriate error message.
- You may assume that the client only connects to a program running your server and vice versa.
- Your implementation should support multiple concurrent clients/players at the server

Running your Program

To run your server and client, it is necessary to start the server before the client. The IP address for `cs1` is `10.124.72.20`. If either program terminates (either Ctrl-C or program error) while the connection is still active, you may get a bind error. If this happens, select a different port. In this situation, the OS thinks the port is still being used and it takes a minute or two for the OS to figure out the process using that port has terminated.

Sample Playing of the Game

For this sample playing, assume that the mystery number is 691.

```
Welcome to Number Guessing Game!
Enter your name:  Aditya


Turn: 1
Enter a guess: 500
Result of guess: Too low

Turn: 2
Enter a guess: 750
Result of guess: Too high

Turn: 3
Enter a guess: 625
Result of guess: Too low

Turn: 4
Enter a guess: 688
Result of guess: Too low

Turn: 5
Enter a guess: 719
Result of guess: Too high

Turn: 6
Enter a guess: 704
Result of guess: Too high

Turn: 7
Enter a guess: 696
Result of guess: Too high

Turn: 8
Enter a guess: 692
Result of guess: Too high

Turn: 9
Enter a guess: 690
Result of guess: Too low

Turn: 10
```

```
Enter a guess: 691
Result of guess: Correct guess!


Congratulations! It took 10 turns to guess the number!

Leader board:
1. Maya 3

2. Aadya 5

3. Aditya 10
```

Submitting your Program

You should submit the following in a **single zip file.** Don't upload/email multiple files separately---doing so will result in a penalty (-10). These are hard to track:

1. README file containing:
    a. Instructions for compiling and executing your program(s). Include an example command line for the same.
    b. Include any/all information needed to successfully compile and run your program. **Note that if we cannot compile/run your program based on instruction in the README, your program will receive a zero.** No extra suggestions/directions to this end would be accepted on email, etc.
    c. If no README file is submitted, we will try our best to run the program; if we succeed, there will be a **penalty of 10 points** on the final score---for not including README. If we don't succeed, the submission will receive a zero. No README files will be accepted separately/later.
    d. If your implementation does not work, you should also document the problems in the README file, preferably with your explanation of why it does not work and how you would solve it if you had more time.
2. All your code files and any other files that might be needed for executing your code.

Notes on Grading:

As mentioned before, do this assignment in C/C++.

1. Y**our code should compile and run (as expected) on cs1.seattleu.edu.** This machine will be used for grading. If a submission does not compile and run on cs1, it receives a zero.
2. No modifications will be allowed (or done by the instructor) to the submitted code for evaluation. **Only the functionality that can be verified/demonstrated without any code modifications and using the code you have submitted will be graded.** If some functionality is unverifiable it will receive no credit.
3. Unintelligible print messages, prints out of order or interweaved with other messages shall be severely penalized---as they interfere with verifiability of program correctness.
4. Only the last assignment submitted before the due date and time will be graded.

Please test your code thoroughly and make sure it meets all the stated requirements.