CAN201 In Class Test 1 Thursday Session 3

Multiplayer Number Guessing Game (TCP Sockets)

Objective

This in class test is required to use Python for socket programming. You will create a multiplayer "**number guessing"** game using TCP sockets programming, with one server and multiple clients (two clients for testing).

Task Description

You need to implement two simple programs: a server and a client. The server manages the game logic, while the clients communicate with the server to participate in the game.

Game Rules：

Server Behavior

1. **Initialization**: The server listens/waits on a specified **IP** and **port**.
2. **Client Connection**:
   - When a client connects, the server asks for the client's name.
   - After receiving the name, the client is marked as "ready".
3. **Game Start**:
   - The game starts when all connected clients are "ready".[1]
   - The server generates a random number (e.g., 1-100).
4. **Gameplay**:
   - The server prompts each client in turn to guess the number.
   - The server checks the guess:
     - If correct, the server announces the winner to all clients and ends the game.
     - If incorrect, the server informs all clients whether the guess is too high or too low, and then prompts the next client.
   - Next turn order: the player who guesses closest gets to guess **last** in the next turn.
5. **Game End**: The game ends when a client guesses correctly. (To make it simpler, no restart is required. That means we only test once.)

Client Behavior

1. **Connect to Server**: Clients connect to the server using its **IP** and **port** of the server.
2. **Submit Name**: Clients send their name when prompted by the server.
3. **Gameplay**:
   - Clients wait for their turn to guess.
   - When prompted, the client sends a guess to the server.
   - The client receives feedback from the server about their guess.
6. **Game End**: Clients are notified when the game ends. (To make it simpler, no restart is required. That means we only test once.)

---

[1] When we test your code, we will start two clients (two clients code should let the server know their joining) first, and then enter client's name one by one.

Implementation Requirements

- **Language**: Python 3.x
- **Sockets**: Use socket module, TCP protocol.
- **Error Handling**: Implement basic error handling for invalid inputs.

Program Execution

1. **Server Execution (in terminal)**:
   python server.py --ip <server_ip> --port <port>
   Example:
   python server.py --ip 192.168.0.100 --port 42345
2. **Client Execution (in terminal)**:
   python client.py --ip <server_ip> --port <port>
   Example:
   python client.py --ip 192.168.0.100 --port 42345

Submission

Package server.py and client.py into a zip file and submit it to Learningmall's submission link.

Sample Interaction

Server Output

```
Server started on 192.168.0.100:42345
Client connected: (192.168.0.101, 54321)
Client connected: (192.168.0.102, 54322)
Received name: Alice
Received name: Bob
All clients are ready. Starting the game.
Random number generated: 42
Prompting Alice to guess.
Received guess 30 from Alice. Too low.
Prompting Bob to guess.
Received guess 90 from Bob. Too high.
Prompting Bob to guess.
Received guess 42 from Bob. Bob wins!
```

Client Output (Alice)

```
Connected to server at 192.168.0.100:42345
Enter your name: Alice
Waiting for other players...
Game started!
Your turn to guess.
Enter your guess: 30
Your guess is too low.
Waiting for your turn...
Bob guessed 90. The guess is too high.
```

```
Waiting for your turn...
Bob guessed 42 and won the game!
```

Client Output (Bob)

```
Connected to server at 192.168.0.100:42345
Enter your name: Bob
Waiting for other players...
Game started!
Waiting for your turn...
Alice guessed 30. The guess is too low.
Your turn to guess.
Enter your guess: 90
Your guess is too high.
Your turn to guess.
Enter your guess: 42
Congratulations! You guessed the number!
```

Grading Criteria (Total: 5 points)

1. **Functionality (2 points)**
   - **2 points**: Correct implementation of game logic, including generating a random number, prompting clients to guess in turn, evaluating the guess (higher/lower), and broadcasting results to all clients.
   - **1 point**: Basic communication between the client and server is established, including client connection and name registration.
   - **0 point**: No submission or incorrect submission.
2. **Code Quality and Standards (1 point)**
   - **1 point**: Code is well-structured, with appropriate comments, clear variable naming, and logical organization.
   - **0.5 points**: The submitted program has no major syntax errors and can run correctly.
   - **0 point**: No submission or incorrect submission.
3. **Submission (2 point)**
   - **2 points**: Complete submission of server.py and client.py files during the lab.
   - **1.75 points**: Complete submission of server.py and client.py files within 24 hours.
   - **1.5 points**: Complete submission of server.py and client.py files within 2 days.
   - **1.25 points**: Complete submission of server.py and client.py files within 3 days.
   - **1.0 point**: Complete submission of server.py and client.py files within 4 days.
   - **0.75 points:** Complete submission of server.py and client.py files within 5 days.
   - **0 points**: Incomplete or late submission[2] or missing any file.

[2] After the cut-off day.