Spring 2020 Computer Networks

Project (2020. 4. 30.) (Total Points: 50)

In this project, you will write client-server programs for the Number Baseball Game using Python 3.7 (above). *The server and client play a game itself using your algorithm.* Submit the following materials by **2020.06.07 (Sun) 23:59:59**.

1) A client code (Name_client.py)

2) A server code (Name_server.py)

3) A document in PDF format describing your algorithm for guessing numbers.

1. Rules of the Number Baseball Game

- There are two players: 1) server player and 2) client player.

- Each player chooses 3-digit number at random, each of digit is between 1 and 9. ('0' is excluded.) These three digits must be different from each other. (e.g., '113' is not possible.)

- Each player guesses the number of other player by saying 3-digit guessing number in turn. These three digits must also be different from each other. (e.g., '113' is not possible.)

- For the guessed number, each player must inform the result by counting the number of strikes and balls. When a guessed digit is the same as the answer digit with the same position, it is called "strike". When a guessed digit is the same as the answer digit but the position is different, it is called "ball". For example, the client player guessed the number as '435' and the answer number of the server player is '365'. The server player must inform to the client player that there are 1 strike and 1 ball.

- The client player starts a guess first, but the server player will have the same number of chances for guessing numbers.

- Type of game results

  • Client Win: The client player guessed the right answer but the server player could not.

  • Server Win: The server player guessed the right answer but the client player could not.

  • Draw: Both the client player and the server player guessed the right answers.

2. Display and message formats for network communications

(1) Common

- The client should display the sent message and received message as follows.

    To Server: (Message here)

    From Server: (Message here)

- The server should display the sent message and received message as follows.

    To Client: (Message here)

    From Client: (Message here)

(2) Initialization

<Server>

- The server program must be run first. If the server program starts, display the following message on the server program.

    The server is ready to receive a game request.

- If the server receives the following message from the client, then the server sends the message of 'ok' and the game starts.

    request_game

<Client>

- If the client program starts, display the following message on the client program.

    Do you want a number baseball game? (Yes or No)

- If the client types 'Yes', then the client sends the following message to the server.

    request_game

- If the client types 'No', then the client program ends.

- If the client receives the following message from the server, then the game starts.

    ok

(2) During the game

<Client/Server>

- The client and the server send the following message including its guessed number and the response to the guess of other player.

[Digit_1,□Digit_2,□Digit_3]/[Number_of_Strikes,□Number_of_Balls]

※ □ is a white space between a comma and a number.

- The examples are as follows:

[1,□2,□3]/[2,□0]	→ Guessed number: 123, Result: 2 Strikes, 0 Ball
[4,□5,□2]/[1,□1]	→ Guessed number: 452, Result: 1 Strike, 1 Ball
[9,□1,□3]/[0,□0]	→ Guessed number: 913, Result: 0 Strike, 0 Ball (Out!)

<Client>

- When the client sends the message at first, then the number of strikes and the number of balls are set to zero initially.

(3) Game results

<Client>

- If the server guessed the right answer and client did not guess the right answer yet, then the client displays the following message on the client program.

```
Client Lose!
```

The client sends the following message to the server and ends the game.

[0, 0, 0]/[3, 0]

- If the client guessed the right answer, then calculate the number of strikes in the guessed number of the server.

If the server guessed the right answer, then the client displays the following message on the client program and ends the game.

```
Draw!
```

If the number of strikes of the server is less than 3, then the client displays the following message on the client program and ends the game.

Client Win!

After this, the client sends the right answers again with the response to the guess of the server so that the server can know it is right or not.

<Server>

- If the client guessed the right answer, then the server tries the last guess and wait the response from the client.

If the server also guessed the right answer, then the server displays the following message on the server program and ends the game.

Draw!

If the server did not guess the right answer, then the server displays the following message on the server program and ends the game

Server Lose!

- If the server guessed the right answer but the client did not guess the right answer yet, then the server displays the following message on the server program and ends the game.

Server Win!

3. Examples of games

For simplicity, it is assumed that the right answers of the client and the server are the same as '123'.

(1) Client Win!

Server	Client
The server is ready to receive a game request. From Client: request_game To Client: ok From Client: [3, 5, 4]/[0, 0] To Client: [6, 5, 8]/[0, 1] From Client: [1, 2, 3]/[0, 0] To Client: [8, 3, 7]/[3, 0] From Client: [1, 2, 3]/[0, 1] Server Lose!	Do you want to play a game? (Yes or No) **Yes** To Server: request_game From Server: ok To Server: [3, 5, 4]/[0, 0] From Server: [6, 5, 8]/[0, 1] To Server: [1, 2, 3]/[0, 0] From Server: [8, 3, 7]/[3, 0] Client Win! To Server: [1, 2, 3]/[0, 1]

(2) Server Win!

Server	Client
The server is ready to receive a game request. From Client: request_game To Client: ok From Client: [3, 5, 4]/[0, 0] To Client: [6, 5, 8]/[0, 1] From Client: [4, 6, 3]/[0, 0] To Client: [1, 2, 3]/[1, 0] From Client: [0, 0, 0]/[3, 0] Server Win!	Do you want to play a game? (Yes or No) **Yes** To Server: request_game From Server: ok To Server: [3, 5, 4]/[0, 0] From Server: [6, 5, 8]/[0, 1] To Server: [4, 6, 3]/[0, 0] From Server: [1, 2, 3]/[1, 0] Client Lose! To Server: [0, 0, 0]/[3, 0]

(3) Draw!

Server	Client
The server is ready to receive a game request. From Client: request_game To Client: ok From Client: [3, 5, 4]/[0, 0] To Client: [6, 5, 8]/[0, 1] From Client: [1, 2, 3]/[0, 0] To Client: [1, 2, 3]/[3, 0] From Client: [1, 2, 3]/[3, 0] Draw!	Do you want to play a game? (Yes or No) **Yes** To Server: request_game From Server: ok To Server: [3, 5, 4]/[0, 0] From Server: [6, 5, 8]/[0, 1] To Server: [1, 2, 3]/[0, 0] From Server: [1, 2, 3]/[3, 0] Draw! To Server: [1, 2, 3]/[3, 0]

4. Socket Programming

(1) Transport layer protocol: Use TCP.

(2) In the client program, write the following.

```
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect(('localhost', serverPort))
```

(3) In the server program, write the following.

```
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('localhost', serverPort))
```

5. Scores

(1) Socket programming [40 points]

- If your client/server programs work well with the programs made by other students, the maximum points are 40 points.

- If your client/server programs do not work with the programs made by other students, but your own client and server programs can play with each other, then the maximum points are 30 points.

- If even your own client and server programs cannot play with each other, then the maximum points are 20 points.

- If there are any problems in your code, some score will be deducted from the maximum points.

(2) Algorithm [10 points]

- Algorithm description in a document: 5 points

- Superiority of the algorithm: 5 points

※ If your client/server programs do not work with the programs made by other students, you cannot obtain these 5 points.

※ There will be a tournament of all of your programs.