Project 3 – A Networked Boggle Protocol Implementation

Due date of implementation – October 30th (at 11:59:59pm). Worth 100 points.

Students may work on this project alone or in a group of two. You can use Python3, Java, or C to implement the game, or a mix of two languages.

Deliverables:

- **Boggle client code:** This should be only the code (not executable).
- **Boggle server code:** This should be only the code (not executable).
- **Readme.txt document:** This should contain 1) names of your group members, 2) how you split up the work, 3) any sources used to help you complete your assignment, 4) any challenges you encountered, 5) example working gameplay (copy and paste the output of the terminals).

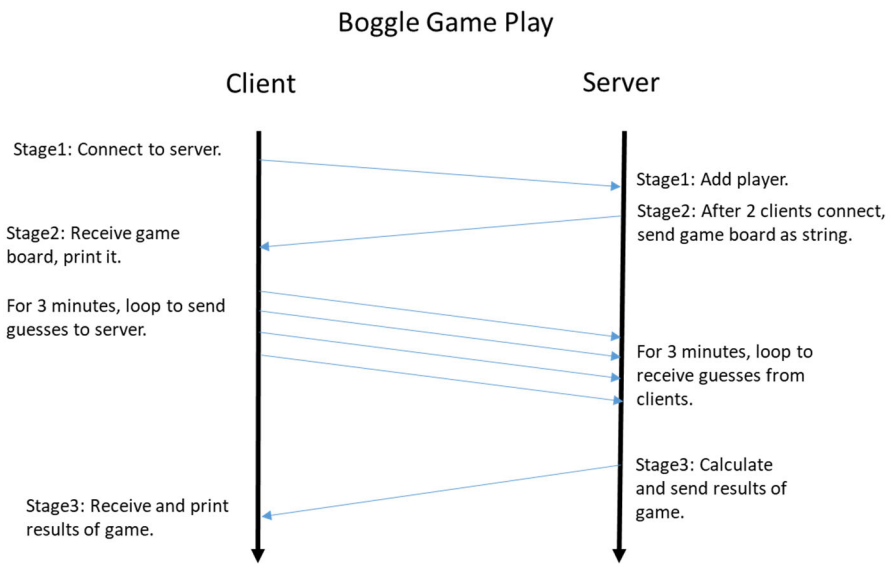
Intro

Boggle © is a challenging word-finding game that can be played in as little as 3 minutes and can be played by two or more players. You can find the official rules of the game at this website: http://www.hasbro.com/common/instruct/boggle.pdf . (Ignore the Boggle challenge cube.)

Our goal is to turn Boggle into a multi-player networked game consisting of a single server program which will interact with two clients. However, in the future, the game could easily be expanded to more players.

Start by modifying the starter code for the client and server programs in the language of your choice.

Game Play



Server: Stage 1 – Allow multiple clients to connect

The server should be started. Be sure the port variable is coded to be a port number in your assigned range (check Blackboard announcements if you've forgotten your range).

```
$> python3 server.py
```

The server should then display the following message:

```
"Waiting for clients to connect."
```

When the first client connects, the server should print:

```
"Player1 has joined the game."
```

When the second client connects, the server should print:

```
"Player2 has joined the game."
```

After the second client connects, it should not allow any more new connections. The server should send the game board to the clients. That will move us into stage 2.

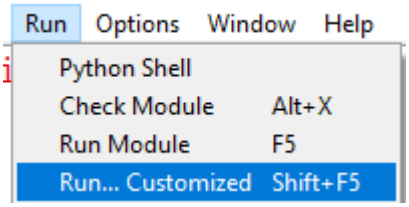
Client: Stage 1 – Connect to the server

The client should be started as shown below, where <server> is the IP address or hostname of the server, and <port> is the port number of the server.

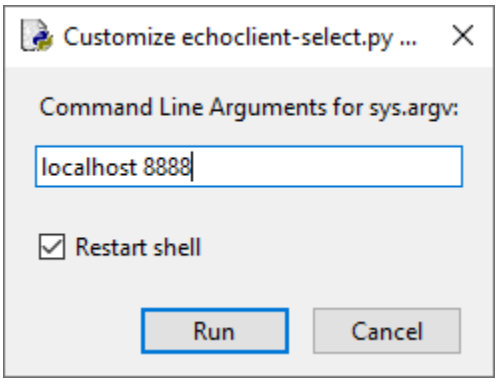
```
$> python3 client.py <server> <port>
```

Hint: You can pass command line arguments in IDLE now, as long as IDLE has been updated to the latest version.

Select Run -> Run… Customized



In this example, I am running the server on my own laptop. In that case, I can pass "localhost" or "127.0.0.1" in as the server. If you were running the server code on student.cs.uni.edu (so that multiple people in different locations can connect), then you could use "student.cs.uni.edu" as the server. I'm using 8888 as the port, but be sure to use a real port number in your assigned port range.



Once the client connects to the server, it should display the message:

"Connected and waiting for the game to start."

The client should then wait to receive the game board from the server. It should not allow a prompt or do anything else until it receives that message.

Stage 2 – Send out Boggle board information and accept words

The server will create a random Boggle board of 4x4 letters arranged in a cube and send this information out to each client. (Remember, clients cannot necessarily see what is printed on the server screen, so the board information must be sent to each client.) The clients must accept the board information and display it on each client screen in a text square (4 lines of 4 letters). A sample client screen may look like this:

```
E T S M
A S A P
W Z D E
U W H V

Start entering words!
```

Next, the server will allow the clients to send unlimited packets back to the server containing the client's username and word guesses for up to 3 minutes. (You may want to set a smaller time for testing, but 3 minutes is the official amount of time). After 3 minutes, the server and clients will move on to Stage 3.

Stage 3 – Tabulate the results

In Stage 3, the server will accept no more word guesses from the clients. The server should begin tabulating results.

The server then must eliminate duplicate words from each player's list and score the remaining words according to Boggle rules. The final results and words should be sent to each client and displayed on the clients' screens. A sample client screen is below.

```
Player 1 unique words: paste, mad, sad
Player 2 unique words: pasta, pastas, mast, saw, was
Player 1 points: 4
Player 2 points: 7
Player 2 wins!
```

The server could be improved by having subroutines to verify the submitted words are in the Boggle board and to check the submitted words against an English dictionary. However, that functionality is not strictly necessary, since each player will see the other players' submitted words and can call a game null and void if one of the players is a dirty cheater.

Finally, after the results have been sent to the clients, both the clients and server should eventually terminate.

Some Implementation Hints

Keeping Track of the Stage

The server needs to do different things depending on the stage it is in. The easiest way to keep track of stage is to have a variable. Change the variable when it is time to change stages. For example, if stage==2, the server should immediately disconnect any *new* clients that try to connect. If stage==3, the server should .send() the game results to all the clients in the read/inputList. You probably ***don't need to change the structure of the server loops, or add any loops, to the code***. (In other words, just use if statements to check the stage number and do different things within the code structure already there.)

The client also needs to do different things depending on the stage. For example, in stage 1, the client needs to connect to the server and wait for the game board. *Those things should not happen in a loop.* In stage 2, the client should enter a loop while where it asks for input and sends guesses to the server for 3 minutes. Once the time expires, the client should break out of the loop and move to stage 3 where it will .recv() one final message about who won.

Real Boggle Dice

The real Boggle game comes with sixteen letter cubes. Each cube has six sides with a letter on each side. The letters on each cube are not random because the English language is not random. Instead, they were chosen in such a way that common letters come up more often and it is easier to get a good mix of vowels and consonants.

The following lists all of the letters on all six faces of each of the sixteen cubes. You should decide on an appropriate way to represent this information in your program and declare it accordingly.

AAEEGN, ABBJOO, ACHOPS, AFFKPS

AOOTTW, CIMOTU, DEILRX, DELRVY
DISTTY, EEGHNW, EEINSU, EHRTVW
EIOSST, ELRTTY, HIMNQuL, HLNNRZ

Timing

One phase of the game relies on timing. You will want to look up how to get the current time in your language of choice.

In the server, notice that select() is inside of a while loop that is constantly looping and polling all connected sockets for incoming messages. When you change to stage 2, you could get the current time and add 3 minutes to it as the ending time. Every time the server loops in the while loop, you could have it check to see if the current time now exceeds the ending time. If it does, then it should send one final message about the winner to each client before exiting.

In the client, you could do something similar. When the client transitions to stage 2, it should enter a while loop that asks for input and allows the user to send a word to the server. Every time through the while loop, you could get the current time. Only loop while the current time is less than the ending time.

Select

You will not have to change any of the parameters to select from the sample code. Only use the sockets in the select read/input list (first list).

Turning in the assignment

You can submit the assignment on eLearning. Go to this class and navigate to the project 3 submission area. Please submit your files one at a time in the required places.

Rubric

Boggle server	**Points Possible (50)**
Phase 1 works	10
Phase 2 – generates and sends random boggle board	10
Phase 2 – accepts words for 3 mintues	10
Phase 3 – calculates winner correctly	10
Phase 3 – sends winner information to clients	10
Boggle client	**Possible Points (40)**
Phase 1 works	10
Phase 2 – displays Boggle board in square	10
Phase 2 – sends words to server for 3 minutes	10
Phase 3 – displays winner information	10
Full README	**Possible Points (10)**