Project 2 Report: Descriptions of Structures and Attributes of New functions

“s” – Set Command:

The purpose of the “s” command is to initialize the contest. The “s” command takes a contest number argument and allows other functions to use the given contest number. If a contest number is not supplied, the S command will throw an error telling the user that the contest number is missing.

The “s” command is sent from the contestmeister to the server. The s command is converted into a json structure using the json.dumps() function and encoded, then sent to the server using socket.send(). On the server side, the server receives the packet with the socket.recv(1024).decode which converts the encoded packet back to json, which can be loaded with the json.loads command. Since the server does not know what command it is, the server splits the function using string splicing and identifies what function it must use from the initial string value. If there are no other values, the server sends back an error message asking the client to enter a valid number.

In the server, there is a global array that holds the contest numbers. This ensures that all functions and threads would be able to access the contests number. If a function fails to find a contest number, then the contest number is not initialized.

“a” – Add Command:

The purpose of the “a” command is to add questions to a given contest number. Initially, all initialized contest numbers start with 0 questions. When the “a” command is called, a given contest number will be supplied a question.

The “a” command takes in 2 arguments: the contest number, and the question number. If either or arguments are not supplied, the server will throw an error that it is missing an argument. If the contest number is referring to a non-initialized contest number, the server will throw an error that the command has an invalid contest number. If the question number does not point to an actual question, then the server will throw an error explaining that the question number does not exist.

The “a” command is sent from the contestmeister to the server. In the contestmeister, the “a” command is converted into a json structure using the json.dumps() function and encoded to be sent to the server using the socket.send() function. On the server side, the server receives the packet and decodes the packet with the sock.recv(1024).decode() function. The arguments are analyzed by the add command function through string splitting using the split(“ “) function and expects the split string array to have 3 entries: The command, the contest number, and the question number.

Once the data is received, the “a” command checks through the question dictionary (which is connected to the qbank.txt file) to see if the question number is valid and then checks the contests array to see if the contest number is valid. Once it is determined that the two arguments are valid, the “a” command appends a global array that holds the contest number and the question number. The global array ensures that other functions are able to access the questions associated with the contest number. This is beneficial to actually holding the whole question because questions can hold a lot of data compared to a single string character (the numbers are held in a string).

“b” – Begin Command:
The purpose of the “b” command allows a contestant to connect to a contest and starts the contest when the first question is answered.
The “b” command takes 1 argument: the contest number. If the contest number is not supplied, then the server will throw an error that the contest number is missing. If the contest number is not set, then the server will throw an error message saying that the contest number is invalid.
The “b” command is set from the contestmeister to the server. In the contestmeister, the “b” command is converted into json structure using the json.dumps() function and encoded to be sent to the server using the socket.send() function. On the server side, the server receives the packet and decodes the packet with the sock.recv(1024).decode() function. The arguments are analyzed by the add command function through string stripping and splicing using the strip(“ “) and [1:] functions. The “b” command expects 1 argument: the contest number.
Once the data is received to the server, the “b” command checks to see if the argument is valid by searching through the contests array. If the contest number is in the array, then the “b” function creates a thread and calls a newClient() function. This function creates a new socket and outputs a new port number. Another thread is created within the thread for the countdown timer. The countdown timer uses the time.sleep() function to countdown 60 seconds and the deny any attempts to connect to the socket after the timer finishes.
When the connection is established, the contestant interface is setup. The server expects the client to send a nickname and the server checks if the nickname is already been used by searching through the nickname global array, which holds the nicknames and contestant number for each user. If the nickname is not unique for the contest, then the server will throw an error to the contestant to type a unique nickname.
Once the nicknames are established, the server registers the user to the global contestant array, which holds the contest number, nickname, and socket of each contestant. The server then proceeds by sending questions to the contestants. The client receives each question and organizes the question to the standard output. After every question, the server does not send a question until it counts all the contestants’ question and see if it matches the number of contestants in the contestants’ array for the given contest number. This ensures that each question will be sent to each contestant the same time. Each answer is documented to a global contest answer array for statistics of the contestmeister but also used to determine the top score and average . This process repeats until all questions are used up.

“l” – List command:
The purpose of the “l” command is to list all the contest numbers and some statistics of each contest. The “l” command takes no arguments.
The “l” command is set from the contestmeister to the server. In the contestmeister, the “l” command is converted into json structure using the json.dumps() function and encoded to be sent to the server using the socket.send() function. On the server side, the server receives the packet and decodes the packet with the sock.recv(1024).decode() function.
On the server side, the “l” function is identified by the first character and the function utilizes the contestanswers array to determine if the function ran or not. If the contestanswers array contains the contest number, then the function has run. The function then counts the correct

answers on a per user and all user basis to determine the top score and the average score. The data is then compiling to a string and sent back to the client to be outputted as a response.

"r" – Review command:
The purpose of the "r" command is to get information of a specific contest. This command outputs the maximum correct answers for the contest, the average score, and the percentage of users who obtained each specific question correct.
The "r" command takes one argument: the contest number. If the contest number is not supplied, then the server will throw an error that the contest number is missing. If the contest number is not set, then the server will throw an error message saying that the contest number is invalid.
The "r" command is set from the contestmeister to the server. In the contestmeister, the "r" command is called and the whole command is converted into json structure using the json.dumps() function and encoded to be sent to the server using the socket.send() function. On the server side, the server receives the packet and decodes the packet with the sock.recv(1024).decode() function.
On the server side, then "r" function is identified by the first character and the function utilizes the contestanswers array to determine if the function has run. The function the counts the correct answers on a per user and all user basis, similar to the "l" command. However, the "r" function iterates through the contestanswers once more to obtain how the percentage of uses who answered a given question correctly or not.

Brief Descriptions of the files:
Cserver.py: This file contains all the function that allow contestmeister to perform functions. The Cserver.py also contains the database files and global arrays to hold information from its clients. This file does not require any arguments.
Contestmeister.py: This file simply contains an interface that allows for creation of a question, deletion of a question, get a question, set contest, add to contest, begin contest, list contest results, and review contest. This interface is the only way to get the server to do these functions. The contestmeister, being a client does very little preprocessing for data which will be processed in the server (Most functions are just packages into a json encoding and sent to the server and just waiting for a response). This file requires the hostname and the port number as the argument. This also has an option argument that allows commands to be inputted through a file.
Contestant.py: This file is simply the interface to obtain questions and send answers to the server. This file specifically can only do those two functions. This file requires the host name and the port number as the argument.