Electrical and Computer Engineering, Purdue University Northwest
Intro. to Computer Comm. Networks (ECE 547), Fall 2019
Hands-on Assignment 1, Due Date: 11/08/2019

Problem 1 [10 points] Compare HTTP 1.0, HTTP 1.1, and HTTP 2. Cite references that you used for your answer.

Problem 2 [15 points] Write a short note (100-200 words) on BitTorrent protocol. Refer Chapter 2 from the textbook.

Task 0 [5 points]: Follow the steps mentioned below to install Virtual Box and run the virtual machine (VM) provided by SEED labs.

a) Download Virtual Box from https://www.virtualbox.org/ and install it in your system
b) Download VM image available at
https://drive.google.com/file/d/1HxdUhq-J_-_QKyjngpH9m6Kmuvy0_68a/view?usp=sharing
Check the user manual for username and password that you will use in Step c) for login.
c) Follow the tutorial available at
http://www.cis.syr.edu/~wedu/seed/Labs_16.04/Documents/SEEDVM_VirtualBoxManual.pdf
d) Once installed, put the screenshot in the report to support your completion of Task 0.

Note: Refer TCP Socket Programming and HTTP protocol to implement the given tasks

Task 1 [30 points]: Implement a client-server Quiz game. The server will read **quiz.txt** file whenever a client gets connect with it. Each line in the file has a question and its answer separated by comma. The server will **randomly** order the questions after reading them from the file and send them to the client one by one. The client will respond each question and if the response matches with the answer then it is correct. If the client answers three questions incorrectly, then the server will send a message to the client '**Please come again after practice!**' and client program will be terminated. If the client answers all the questions correctly with only two or less mistakes, then the server will send a message to the client that '**Congratulations, you played very well!**' and the client program will be terminated. Your server should be able to play with multiple client players at the same time. You may refer Python multithreading to achieve this from the following link:
https://www.geeksforgeeks.org/multithreading-python-set-1/

Playing the game

Put the attached **quiz.txt** file in the same directory that the server Python program is in. Open a terminal and run the server program. Now open two more terminals and start the client program in each terminal and play the game for each client.

Submit client and server code along with the screen shots, verifying that your game has implemented properly for all the cases. The name format for the client and server program files are **`firstname_lastname_game_client.py and firstname_lastname_game_server.py`**

Task 2 [40 points]: Develop a web server that handles HTTP requests from multiple browser windows or tabs simultaneously. Your web server should accept and parse each HTTP request, get

the requested file from the server's file system, create an HTTP response message consisting of the requested file preceded by header lines, and then send the response directly to the client. If the requested file is not present in the server, the server should send an HTTP "**404 Not Found**" message back to the client.

Running the web server

Put the attached HTML file **index.html** and **assignment1.txt** file in the same directory that the web server Python program is in. Run the server program. Open a browser and provide the corresponding URL. For example:
http://127.0.0.1:8080/index.html or http://127.0.0.1:8080/assignment1.txt
'index.html' and 'assignment1.txt' are the names of the files you placed in the server directory. Note also the use of the port number after the colon. You need to replace this port number with whatever port you have used in the server code. In the above example, we have used the port number 8080.
The browser should then display the contents of index.html. If you omit ":8080", the browser will assume port 80 and you will get the web page from the server only if your server is listening at port 80. If the user does not provide any file name, i.e. http://127.0.0.1:8080/, the server should return index.html page in response to the request.
Then try to get a file that is not present at the server (e.g. ece547.html). You should get a "**404 Not Found**" message.

Submit complete server code along with the screen shots of your client browser, verifying that you actually receive the contents of the HTML or text file from the server. The name format for the server program file is **`firstname_lastname_webserver.py`**.