Assignment 1

Due Date: April 8th, 2020

Contact: TA Seunggyu Ji (jiseunggyu@postech.ac.kr)

Instructions

The assignment is Python socket programming. Follow the instructions below:

- Write your code correctly in each blank.
- One line of code is sufficient for each blank.
- One point for each line with no partial point.
- -50% for one day late submission, -100% for more than one day without exception.
- Submit your .py file on LMS - Assignments.
- Please refer to the address below
  https://docs.python.org/ko/3.7/library/socket.html

Programming Problem 1. [14 points]

In this assignment, you will develop a simple Web server in Python that is capable of processing only one request. Specifically, your Web server will (i) create a connection socket when contacted by a client (browser); (ii) receive the HTTP request from this connection; (iii) parse the request to determine the specific file being requested; (iv) get the requested file from the server's file system; (v) create an HTTP response message consisting of the requested file preceded by header lines; and (vi) send the response over the TCP connection to the requesting browser. If a browser requests a file that is not present in your server, your server should return a "404 Not Found" error message.

```
# Import socket module
from socket import *
import sys # In order to terminate the program

# Create a TCP server socket
#(AF_INET is used for IPv4 protocols)
#(SOCK_STREAM is used for TCP)

serverSocket = 

# Assign a port number
serverPort = 6789

# Bind the socket to server address and server port

# Listen to at most 1 connection at a time

# Server should be up and running and listening to the incoming connections

while True:
        print('The server is ready to receive')

        # Set up a new connection from the client
        connectionSocket, addr = 

        # If an exception occurs during the execution of try clause
        # the rest of the clause is skipped
```

```
        # If the exception type matches the word after except
        # the except clause is executed
        try:
                # Receives the request message from the client
                message =
                # Extract the path of the requested object from the message
                # The path is the second part of HTTP header, identified by [1]
                filename =
                # Because the extracted path of the HTTP request includes
                # a character '₩', we read the path from the second character
                f = open(filename[1:])
                # Store the entire content of the requested file in a temporary buffer
                outputdata = f.read()
                # Send the HTTP response header line to the connection socket


                # Send the content of the requested file to the connection socket
                for i in range(0, len(outputdata)):

                # Send "₩r₩n" to the connection socket.


                # Close the client connection socket

        except IOError:
                        # Send HTTP response message for file not found


                        # Close the client connection socket

# Close the server socket

sys.exit() #Terminate the program after sending the corresponding data
```

Programming Problem 2. [6 points]

In this assignment, you will write a client ping program in Python. Your client will send a simple ping message to a server, receive a corresponding pong message back from the server, and determine the delay between when the client sent the ping message and received the pong message. This delay is called the Round Trip Time (RTT). The functionality provided by the client and server is similar to the functionality provided by standard ping program available in modern operating systems. However, standard ping programs use the Internet Control Message Protocol (ICMP) (which we will study in Chapter 5). Here we will create a nonstandard (but simple!) UDP-based ping program.

Your ping program is to send 10 ping messages to the target server over UDP. For each message, your client is to determine and print the RTT when the corresponding pong message is returned. Because UDP is an unreliable protocol, a packet sent by the client or server may be lost. For this reason, the client cannot wait indefinitely for a reply to a ping message. You should have the client wait up to one second for a reply from the server; if no reply is received, the client should assume that the packet was lost and print a message accordingly. Write your client ping program.

```
import sys, time
from socket import *
# Get the server hostname and port as command line arguments
argv = sys.argv
host = argv[1]
port = argv[2]
timeout = 1 # in second

# Create UDP client socket
# Note the use of SOCK_DGRAM for UDP datagram packet
clientsocket = dd
# Set socket timeout as 1 second

# Command line argument is a string, change the port into integer
port = int(port)
# Sequence number of the ping message
ptime = 0

# Ping for 10 times
while ptime < 10:
        ptime += 1
        # Format the message to be sent
```

```
data = "Ping " + str(ptime) + " " + time.asctime()

try:
# Sent time
        RTTb = time.time()
# Send the UDP packet with the ping message
        [                                    ]
# Receive the server response
        message, address = [                                    ]
# Received time
        RTTa = time.time()
# Display the server response as an output
        print("Reply from " + address[0] + ": " + [                    ] )
# Round trip time is the difference between sent and received time
        print("RTT: " + str(RTTa - RTTb))
except:
        # Server does not response
# Assume the packet is lost
        print ("Request timed out.")
        continue

# Close the client socket
[                                    ]
```