**CIS527 Fall 2021 Programming Assignment 1**

***Instructor: Zheng Song***

***Due Friday, Dec 10th, 2021***

1. **Introduction**

This programming assignment is designed to let you familiar with the socket interface and client-server application. This assignment weights 15 % of your final grade.

In this programming assignment, you will create a server process that implements yet another "message of the day" (yamotd) protocol with a client process that you will also create. The client and server processes will communicate using TCP sockets and will implement the yamotd protocol discussed below.

The yamotd server performs two functions:

* returns the message of the day to any user that sends the server a MSGGET message
* allows a client to send the server a MSGSTORE message to upload one messages of the day to the server. The uploaded message will replace the originally stored message, to be returned to another client upon receiving the MSGGET message.

1. **The Assignment**

You will write two programs, a server and a client. The server creates a socket in the Internet domain bound to port SERVER\_PORT (a constant you should define in both programs, you may use last 4-5 digits of your UM-ID or your SSN). The server receives requests through this socket, acts on those requests, and returns the results to the requester. The client will also create a socket in the Internet domain, send requests to the SERVER\_PORT of a computer specified on the **command-line**, and receive responses through this socket from a server.

Your client operates by sending a MSGGET, MSGSTORE commands to the server. You should create a client that is able to send any of the two commands above, and allows a user to specify which of the commands the client should send to the server.

**MSGGET**

The MSGGET command, which is sent from the client to the server, consists solely of the ASCII string "MSGGET" followed by the newline character (i.e., '\n'). After sending the MSGGET command, the client will wait to receive a message of the day message back from the server via the socket. After displaying the message of the day, the client should loop back and allow the user to indicate the next command to be sent.

When your server receives a MSGGET command from a client, it should return the string "200 OK" (terminated with a newline), followed by one of the messages of the day (also terminated with a newline). The server should just return one message.

A client-server interaction with the MSGGET command thus looks like:

c: MSGGET   
s: 200 OK   
   Anyone who has never made a mistake has never tried anything new.

*Note, all these messages should be displayed at the client side.*

**MSGSTORE**

The MSGSTORE command allows the client to upload one message to the server. A client that wants to store one message should begin by sending the ASCII string "MSGSTORE", followed by the newline character (i.e., '\n'). The client should wait for the server to return a "200 OK" message (indicating that the client is authorized to upload messages), and then respond to the "200 OK message" by sending one message to replace the server’s message of the day. The message should be terminated with a newline. After sending the MSGSTORE command, and the message of the day, the client should read and display the return code sent by the server.

Upon receiving the message of the day, the server should read in and store the new message of the day. If the message of the day is received correctly, your server should return the string "200 OK" (terminated with a newline).

A client-server interaction with the MSGSTORE command thus looks like:

c: MSGSTORE  
s: 200 OK   
c: Imagination is more important than knowledge.   
s: 200 OK

**Requirements**

The following items are required for full-credit:

* implement the two commands: MSGGET, MSGSTORE
* the server IP address should be a command line parameter for the client program.
* the server should print out all messages received from clients on the screen.
* your source codes must be commented

1. **Grading (100 points)**

* Project report (50 points):
  + Describe the functions that have been implemented, the instructions about how to compile and run your program.
  + design test cases to demonstrate how your system works. Screenshots of both the server and the client are required.
* Correctness and Robustness (45 points)
  + You will lose at least 10 points for any bugs that cause the system crash.
  + You will lose at least 5 points for any other bugs.
  + To dispute the grading of a submitted program, you should provide video clips to demonstrate how your system works as your described.
* Comments and style (5 points)
* The final submission should be a PDF file and a zip file containing only source code (no object file and executable file included).
* No restrictions on the programming language --- feel free to use anything you are familiar with. Make sure you provide enough instructions for me/our TA to compile/run it.