CSCI 631 **Computer Assignment 5** **Spring 2015**
(30 points)

Adding Sequence of Numbers (Concurrent Server)

For this computer assignment a TCP client and a concurrent server program. The client sends a sequence of numbers to the server with the first number is used as an exponent, where each numbers is entered on a separate line. The server apply the exponent to the rest of the numbers in the sequence, add all those numbers, sends the result to the client, and the client prints out the result on stdout. For example, if the following numbers are sent from the client: 2 2 3.5 –7 4.25 –11.11, the client gets the final result of 206.7446 from the server.

To apply an exponent to a number you can use the double pow (double base, double exponent) in math library. This function returns the value $base^{exponent}$ for given base and exponent values. The server starts with an initial value of sum = 0 value, and updates this value after each number is received from the client. After the client sends the last number in the sequence, it calls the shutdown ( ) function by closing only the write-half of the socket, since the server needs to send the final summation result after the client is done by sending its numbers.

For this assignment a client uses the name of the server instead of its IP address. To convert the name of a machine to its IP address, use the system call gethostbyname ( ). You can assume that the target machine has only one IP address.

Use a wrapper function for each of the system calls that you use in your program. In case of an error, execute either the function perror ( ) or fprintf ( ). If a socket function, such as connect ( ), bind ( ), etc., generates an error, do not forget to close the socket before executing the exit ( ) function.

Name the source file of your client program as prog5_client.c, the source file of your server program as prog5_server.c, and the source file of your wrapper functions as wrapper.c; however, you can give them any names you like, but make it sure that name of a source file should clearly identify the contents of the file.

When your program is ready, login both the *tiger* (1st system) and *lambda* (2nd system) machines on separate windows. Then, do the followings:

1. On the 2nd system, make a link to the script P5_2: ln –s ~631/bin/P5_2 and start the script utility: script prog5_2.out. When you are in the script utility, execute the command: P5_2 prog5_client.exe prog5_server.exe. This will execute the server in the background and the client in the foreground, and the client enters the following numbers from the stdin: 1 7.3 –2 6.95 0 13.5 –1.3 on separate lines.

2. On the 1st system, make a link to the script P5_1: ln –s ~631/bin/P5_1 and start the script utility: script prog5_1.out. When you are in the script utility, execute the command: P5_1 prog5_client.exe. This will execute the client in the

foreground, and the following numbers from the stdin: 3 1 2 3 4 5 on separate lines. At the completion of data entry, the script P5_1 will list all your active processes. Terminate the script prog5_1.out by the control character <ctrl>-D, and logout from the 1st system.

3. On the 2nd system, terminate the data entry by the control character <ctrl>-D. At the completion of data entry, the script P5_2 will list all your active processes, and after terminating the server, it will list all your active processes one more time. Finally, terminate your script by the control character <ctrl>-D.

4. Just before you logout from either the 1st or the 2nd system, execute the command: /bin/ps –f –u $USER. This will display a list of all processes generated by your still active programs – some of those processes might even be created in one your previous terminal sessions. In that list, if you see any processes except your login shells, terminate all those processes by the kill command: kill pid … pidn, where pid … pidn is the list of process ids of all processes that you want to terminate.

The correct outputs for this assignment can be found in files prog5_1.out (1st system) and prog5_2.out (2nd system) in directory: ~631/progs/p5.

When your program is ready, on the 1st system: submit the source/header files for both your client and server, and on the 2nd system, submit only the source/header files for your client.

Proper documentation is required for all of your source and header files that you submit for grading.