# SOFE 3200U - Systems Programming
# Fall 2018

## Assignment-2: Using System Calls

## Problems:

1. (**10 Marks**) Write a C program called duplicate which simply copies a file from one place to another. The program will be invoked as follows:
   **duplicate SourceFile TargetFile**
   duplicate must create an exact duplicate of SourceFile under the new name TargetFile. Upon successful completion, duplicate should report the total number of bytes copied and exit with result zero. For example:
   **duplicate: Copied 38475 bytes from file foobar to bizbaz.**

   where **"SourceFile"** is **foobar** and **"TargetFile"** is **bizbaz**

   If the duplicate takes longer than one second, then every second the program will emit a short message:
   **duplicate: still duplicateing...**
   **duplicate: still duplicateing...**
   **duplicate: still duplicateing...**
   If duplicate encounters any kind of error or user mistake, it must immediately stop and emit a message that states the program name, the failing operation, and the reason for the failure, and then exit with result 1. For example:
   **duplicate: Couldn't open file foobar: Permission Denied.**
   **duplicate: Couldn't write to file bizbaz: Disk Full.**
   If the program is invoked incorrectly, then it should immediately exit with a helpful message:
   **duplicate: Too many arguments!**
   **usage: duplicate <sourcefile> <targetfile>**

   Write also a test C program showing the use "duplicate" program. Provide source codes and screenshot of the tests.

2. (**10 Marks**) Implement a simple “chatbot” using “Sockets”. . Client will ask a question and the server will reply with your predefined answer. At least 5 different answers are required for 5 different questions. A sample **clientSocket.c** and **serverSocket.c** are provided.

   Provide source codes and screenshot of the tests.

OR

(**10 Marks + 5 Bonus Marks**) Modifying a system call.

Add a new system call "noopen(void)" that works the same as getpid() and also uses printf() every time it is called to log the pid of the process making the call.

Find (grep can help find these) and examine the kernel code for the:

2.1. getpid() code in the kernel

2.2. two files needed to add a new system call, to see how getpid() is handled, and add new entries for your noopen() system call.

2.3. the file where the "open()" system call is handled (I suggest that you place your noopen() kernel function there).

2.4. the header where the task structure type is defined.

Recompile a new kernel with the modified new system call. To test the new system call, write the application program which can call the new system call and show the output. Provide all source codes and screenshot of the tests.

## Deliverables:

All of the following items should be packaged in a zip folder.
- Working code (60% marks)
- A documentation explaining the code (20% marks)
- Screenshots of test run (20% marks)

**Notes: Assignments have to be completed individually. So, please avoid any sort of plagiarism. Please submit your assignment deliverables to Blackboard.**

**Deadline: November 19, 2018**