CECS 326-01 Assignment 3 (10 points)
Due: 10/12/2021 by class time on BeachBoard

In this assignment you will make use of the POSIX implementation of the Linux shared memory mechanism, and make use of the fork() and exec() system calls to create child processes and control the program that a child process is to execute. For this assignment you need to write two C programs named *master.c* and *slave.c*, which should be compiled into executables *master* and *slave*, respectively. Together they should do the following:

When *master* executes, it should first output a message to identify itself. It should then request to create a shared memory segment of a certain name *xxxxx*, followed by creating *n* child processes, where both *xxxxx* and the number *n* are obtained from the commandline parameters. Each child process is to execute *slave*, with its child number (i.e., 1, 2, etc.) and the shared memory segment name *xxxxx* passed to it from the exec() system call. The *master* process should output the number of *slave*s it has created, and wait for all of them to terminate. Upon receiving termination signals from all child processes, *master* then outputs the content of the shared memory segment filled in by the *slave*s, removes the shared memory and then exits.

The following header file that defines a C struct may be used to structure the shared memory segment:

```
/* myShm.h */
/* Header file to be used with master.c and slave.c
*/

struct CLASS {
     int index;           // index to next available response slot
     int response[10];    // each child writes its child number here
};
```

Suppose program execution is launched as follows:

```
./master 3 my_shm_name
```

master should produce the following sequence of output:

```
Master begins execution
Master created a shared memory segment named my_shm_name     [ my_shm_name is from comandline ]
Master created 3 child processes to execute slave            [ The number 3 is from commandline]
Master waits for all child processes to terminate
Master received termination signals from all 3 child processes
Content of shared memory segment filled by child processes:
 --- content of shared memory ---                           [ Show what child processes wrote ]
Master removed shared memory segment, and is exiting
```

When a child process executes *slave*, it should first output a message to identify itself, and show its child number and the shared memory segment name it obtained from the exec() system call that invokes its execution. It should then open the existing shared memory segment, acquire access to it, and write its child number into the next available slot in the shared memory, close the shared memory and terminate.

slave should produce the following sequence of output:

Slave begins execution
I am child number *x*, received shared memory name *my_shm_name*
I have written my child number to shared memory
Slave closed access to shared memory and terminates

It should be noted that the display of the above sets of output may interleave.

The program must run successfully on Linux.

Do the following for this assignment:

1. Develop two C programs (*master.c*, and *slave.c*) that work as described above. Make sure your source programs are properly formatted as well as adequately and clearly commented. Detailed explanations on all system calls are required and must be in your own words. Simply copying those from the man pages are not acceptable.
2. Submit on BeachBoard the two C programs, a screenshot that shows successful compile of both programs as well as a successful run, and a cover page that provides your name, your student ID, course # and section, assignment #, due date, submission date, and a clear program description detailing what the programs are about. Format of the cover page should follow the cover page template on BeachBoard. The programs must be properly formatted and adequately commented to enhance readability and understanding. Detailed documentation on <u>all</u> system calls are especially needed.