PROGRAMMING ASSIGNMENT#1 (DUE BY 11:59PM JULY 19TH)

PURPOSE

This assignment intends to familiarize you with system calls such as fork, exec, wait, exit and etc., to spawn and terminate other user programs.

ASSIGNMENT

This assignment builds on programming assignment#0, so you must have two executable files ready: `testalphabet` and `testspecial` under `~/programming/zero` folder before taking this assignment.

You will be provided two new files:

```
makefile
mulproc.c
```

Your assignment is to write code in `mulproc.c` to fork two child processes running the two programs `testalphabet` and `testspecial` generated in programming assignment#0 in parallel.

Detailed Requirements:

1) Spawn exactly two child processes, one is to run `testspecial` program and the other is to run `testalphabet` program;
2) When a child process starts executing a program, print a message to the output screen showing which child process (with PID) is running this program (with program name), for example:

   `"CHILD <PID: 16741> process is executing testalphabet program!"`
3) When one child process finishes executing a program, this child process should be terminated, and at the same time, a message should be print to the output screen showing which process (with PID) is done with this program (with program name), for example:

   `"CHILD <PID: 16741> process has done with testalphabet program !"`
4) The messages should match the real execution orders, i.e. when the `testspecial` program starts/ends, the right message should pop up immediately. Hence, you need to track the starting/ending point of each child process.

   The expected screen printing should be similar as follows:

```
  CHILD <PID: 16741> process is executing testalphabet program!
  CHILD <PID: 16742> process is executing testspecial program!
  , -> 745668
  . -> 798072
  ... ...
  CHILD <PID: 16742> process has done with testspecial program! See the results
above!
  a -> 2861232
  b -> 494472
... ...
CHILD <PID: 16741> process has done with testalphabet program! See the results
above!
```

Note: Feel free to do any change of the file (fill the code, create new functions and etc). You also can create new .h and .c files (but I don't think you really need).

Your program must execute correctly on Edoras machine, the instructor will type the following commands to test your code:

```
make// generate executable file mulproc
./mulproc  // two child processes will be generated to run testspecial and
testalphabet programs as stated above.
```

Directions to complete your assignment

1. Double check you have successfully completed programming assignment#0 and two executable files `testalphabet` and `testspecial` are generated under zero folder on edoras
2. Download the source code one.zip from piazza
3. Upload one.zip to Edoras using sftp to programming folder
4. Unzip the one.zip on edoras using the commands:
   `unzip one.zip`
   so you will have one more folders: one (source files) under programming directory on edoras machine
5. Copy two executable programs: `testalphabet` and `testspecial` from folder zero to one
6. Modify source files under folder one to complete this assignment
7. Test your program to make sure it works correctly.

How to submit your assignment

- The source files under the *one* folder on edoras machine will be considered for grading.
- Please finish your coding by **11:59pm July 19th** and make sure your program must execute correctly on edoras. Your grading may be started immediately after the deadline unless notice otherwise, so please make your files ready by the deadline.
- Excuses of "but it worked on my machine" will not be accepted, so if you develop elsewhere, plan to leave time for any migration problems that might arise.