CECS 326-01 Assignment 1 (10 points)
Due: 10/1/2024 online on Canvas

This assignment has two parts. The program done in Part 1 will be needed in Part 2.

This assignment is about process creation and coordination using fork(), exec(), and wait() system calls, and commandline arguments on Linux. Examples of using commandline arguments in C++ program may be found in the Linux Resources module on Canvas. Programming examples of using the needed system calls are available on the #2 lecture-slides set named Processes. Syntax of these system calls and their use can be found using the man pages on Linux by typing the following command:

man fork (or any other system call of interest)

Part 1

Write a C++ program to be named *child.cpp* and compiled into executable *child*. Run *child* with the command as follows:

./child name_of_child age

child will behave as follows:

1. receive a child's name and child's age from the commandline arguments
2. output "I am xxxxx, y years old."
   (Note: content of output depends on data received from the command line arguments, i.e., xxxxx is child's name and y is child's age received from the commandline arguments.)
3. exit

For example, the command below

./*child* Mary 10

should produce an output line

I am Mary, 10 years old

Part 2

Write a second C++ program to be named *parent.cpp* and compiled into executable *parent*. Run *parent* as follows:

./*parent* name_1 age_1 name_2 age_2 … name_k age_k

parent will behave as follows:

1. take in the list of name-age pairs from the commandline arguments
2. create as many child processes as there are name-age pairs on the commandline arguments and have the child process execute the *child* executable from Part 1, and passe to each child process the name of a child and the child's age
3. loop until all child processes have been created and launched
4. wait for all child processes to terminate
5. output "All child processes terminated. Parent exits." then terminates.

Each child process invoked by *parent* will behave as described in Part 1.

Sample run

To invoke the execution:

./*parent* Nancy 12 Roberto 9 Joseph 8

parent process does the following:

1. output "I have 3 children." -- Note: the number 3 comes from the number of name-age pairs in the commandline arguments
2. create 3 child processes, and have each execute *child* and pass to it an integer that represents the the next name and age from the name-age list in the commandline arguments (Note: this step should be done in a loop that iterates as many times as the number of name-age pairs in the commandline arguments.)
3. wait for all child processes to terminate, then
4. output "All child processes terminated. Parent exits."

Output from *child* processes (It should be noted that the order of child's output may vary.)
From first child process:
I am Nancy, 12 years old.
From second child process:
I am Roberto, 9 years old.
From third child process:
I am Joseph, 8 years old.

Output from *parent* process
I have 3 children.
All child processes terminated. Parent exits.

Note: Due to concurrent execution of the parent and child processes, the outputs from these processes may be intermingled, i.e., you may find outputs from another process in the middle of a process' output lines.

Submit on Convas the following:

1. The source programs *parent.cpp* and *child.cpp*;
2. A screenshot that shows successful compilation of *child.cpp* to *child* and a successful run of *child* for Part 1;
3. A screenshot that shows successful compilation of *parent.cpp* to *parent* and a successful run of *parent* for Part 2; and
4. 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 posted on Canvas.
5. The programs must be properly formatted and adequately commented to enhance readability and understanding. Detailed documentation on all system calls are especially needed.