For this lab we had to create a process management simulator. This lab utilized the fork(), wait(), and execvp() functions to simulate how processes are created, managed, and executed.

To begin, create a loop for the number of child processes you want to create. Use the fork() function to create the child process. If pid is less than 0, the fork() function failed to create a child process. If it equals 0, a child process if created. Then call your system command using the exec() or execvp() function. I just chose to use echo, but you can do ls, mkdir, or any other system command. One issue I had was I only used the same echo for all child processes. I believe the goal was to do all different commands, but I think it would require probably an array or list of commands, but I was a little short on time. I then created a loop to loop through all the child processes. Within the loop, call the wait() function which makes the parent process wait for the child processes to finish before it can complete. You also check whether the child processes completed normally or terminated due to some signal. In this case, all our processes completed normally. After all the child processes are either completed or terminated, the parent process can complete itself.

Through this lab, we saw how to create ad manage processes. It was interesting to see how the fork() function worked because from my research, it returns two values, one for the child process created and one for the child process it belongs to. We will probably go more into depth with processes in future labs. Parent processes have to wait for all their child processes to complete using either wait() or waitpid(). I used wait() but waitpid() gives you more options and control over which processes you want to wait for.