A) (2 points)

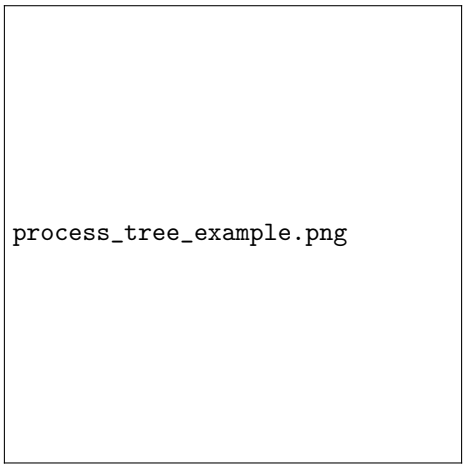
If you create a main() routine that calls fork() twice, i.e., if it includes the following code:

```
pid_t x=-11, y=-22;
x = fork();
if(x==0) y = fork();
```

Assuming all fork() calls succeed, draw a process tree similar to that of Fig. 3.8 (page 116) in your textbook, clearly indicating the values of x and y for each process in the tree (i.e., whether 0, -11, -22, or larger than 0). Note that the process tree should only have one node for each process and thus the number of nodes should be equal to the number of processes. The process tree should be a snapshot just after all forks completed but before any process exists. Each line/arrow in the process tree diagram shall represent a creation of a process, or alternatively a parent/child relationship.

B) (4 points)

Write a program that creates the process tree shown below:



Assume each node represents a process, and the arrows represent the parent-child relationship. The image should show a clear parent-child hierarchy. You must use fork() system calls.

C) (4 points)

Write a program whose main routine obtains two parameters, n and m, from the user (i.e., passed to your program when it was invoked from the shell, n¿2,

m¿0). The program creates m child processes. Each child process calculates the factorial of n using unsigned long long integers and prints the result along with its process ID. The parent waits for all children to exit before terminating. Handle potential errors, such as invalid input (n¡0 or m¡0) and excessively large n values that might cause overflow. Do not use IPC in your solution to this problem (i.e., neither shared memory nor message passing).

D) (3 points)

Explain the difference between the exit() and $_{e}xit()system call sin a Unix-like operating system. Give examples o$

E) (3 points)

Describe a scenario where using multiple processes might be more efficient than using multiple threads within a single process. Explain your reasoning, considering factors like memory management, inter-process communication, and potential benefits of parallelism.

F) (4 points)

Write a C program that uses the exec family of functions to replace the current process image with a shell command provided by the user. The program should handle potential errors during execution, such as incorrect command syntax or the inability to locate the command.

G) (4 points)

Write a program that creates two child processes. The first child process calculates and prints the sum of integers from 1 to 1000. The second child process calculates and prints the product of integers from 1 to 10. The parent process waits for both children to complete and then prints a message indicating the completion of all child processes. Use appropriate error handling to manage potential issues. Do not use IPC.