Problem 1 (2 points)

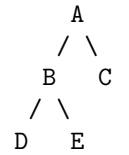
If you create a main() routine that calls `fork()` twice, i.e., if it includes the following code:

```
   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.

Problem 2 (4 points)

Write a C program that creates the following process tree:



Each process should print its own process ID and the PID of its parent. Assume that process A is the initial process.

Problem 3 (4 points)

Write a C program that takes two integer command-line arguments, ‘n‘ and ‘m‘, where ‘n‘ (n ¿ 0) represents the number of times to print ”Hello” and ‘m‘ (m ¿ 0) represents the number of times to print ”World”. The program should create a child process. The parent process should print ”Hello” ‘n‘ times, and the child process should print ”World” ‘m‘ times. The parent process must wait for the child process to complete before exiting. Do not use any inter-process communication mechanisms (pipes, shared memory, etc.).