OPERATING SYSTEMS II

EAP SEMESTER 2021-2022

1st Laboratory **Exercise (10%)**

(Date of Delivery: 20/4/2022)

1. Let there be a program in C at the beginning of which the following commands are given:

```
pid1=fork();
if (pid1==0) { pid2=fork(); }
else { pid3=fork(); pid4=fork(); }
..............................
```

   How many processes were spawned by the previous commands? What is the "affinity" between them? Briefly explain how the above processes were produced (step/command by step/command, with which fork was each one created / which pid corresponds to each one, etc.). Also provide a diagram that clearly shows the hierarchy between the above processes.

2. Write a C program with appropriate **fork()** commands to create a total of (along with the main program) seven processes, with the following affinity structure (tree):

```
          P0
        /    \
      P1      P2
     /       / | \
   P3 P4 P5 P6
```

   Then (that is, after all the processes dictated by the above tree have been created), as main part of its work each Pi process should simply print a message to the screen stating its name (Pi), its PID and its PPID. Your program should additionally include the appropriate wait-communicate commands to satisfy the following constraints-requirements:

   (a) Process P0 before completing its execution should wait for process P1 to complete and print its exit value. Upon completion of its execution P0 should also be replaced by the *cat command,* which will print the source code of your program.

   (b) Process P1 before completing its execution should additionally wait to receive a 'hello' message from your child' from the P3 process (and print it to the screen).

   (c) Process P2 before completing its execution should wait for the completion of at least two of her direct children and print who they are (their PIDs).

   You also extended your program so that process P2 spawns N processes/immediate children (instead of the only three processes/immediate children P4,P5,P6 it appears to spawn in the figure above), where N will be given by the user. Each of the N spawned processes should simply print a diagnostic message with its PID and PPID and exit.

   Is it possible to create orphan or zombie processes while running your program? And what might they be? Comment / adequately substantiate your answer.

<u>Deliverables</u>:

Code, comment/ documentation, indicative runs.

Regarding the structure and format of the deliverable(s), you are invited to follow the instructions given more specifically by the Professor of your department.