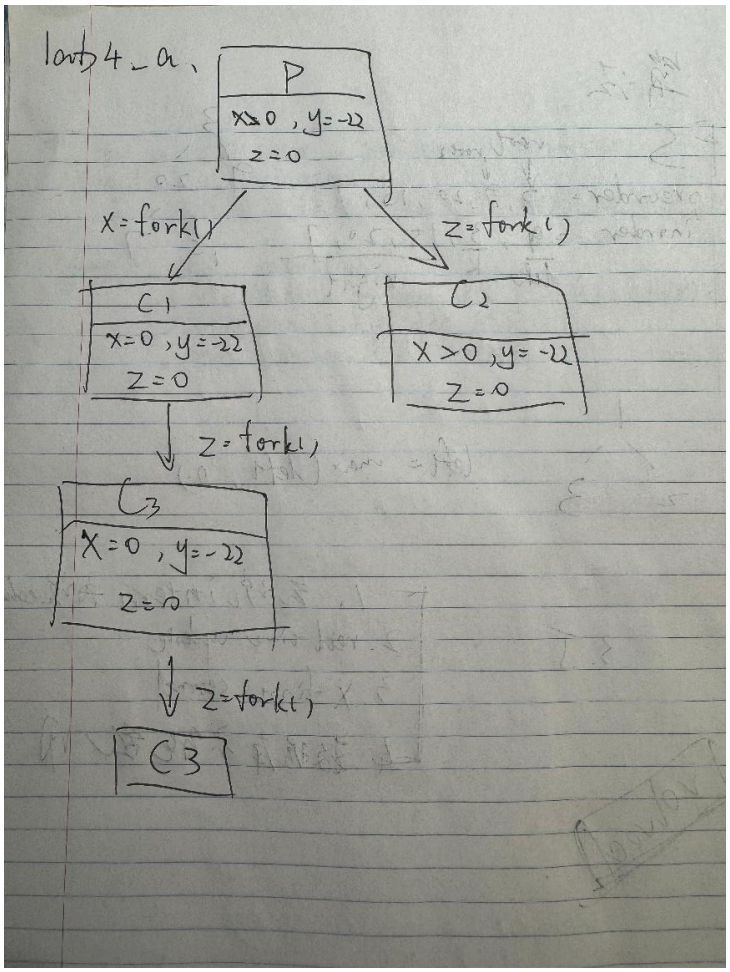
LAB_A

Since y = -22, the condition (y > 0) is false, so the second fork() does not execute.

The first and third fork() calls together create four processes: P, C1, C2, and C3.

The process tree and variable values (x, y, z) are shown in the figure.



LAB_B

The program creates a process tree where the parent has two children, and the second child creates another child.

Each process returns: parent = 0, first = 1, second = 2, third = 3.

The output verifies the correct parent–child relationships.

```
mycroft@mycroft-VMware-Virtual-Platform:~/hw4$ touch lab4_c.c
mycroft@mycroft-VMware-Virtual-Platform:~/hw4$ gcc lab4_b.c -o lab4_b
mycroft@mycroft-VMware-Virtual-Platform:~/hw4$ ./lab4_b
ret=1, PID=4846, PPID=4845
ret=3, PID=4848, PPID=4847
ret=2, PID=4847, PPID=4845
ret=0, PID=4845, PPID=4051
```

LAB_C

The program takes n and d as input.

The child prints n arithmetic terms (0, d, 2d, ⋯), then the parent waits and prints two more (n*d, (n+1)*d).

Example: ./lab4_c 5 2 →

0 2 4 6 8

10 12.