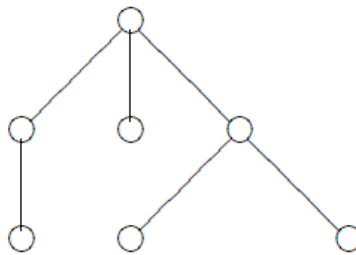


**Problem 1:**

**(20 minutes)**

- A. Write in one statement the code to create the following tree of processes



**P.S:** **if** and **else** are considered as one statement (nested if else is not permitted)

- B. We consider the following piece of code:

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>

void main() {
    printf("I'm the main with pid=%d\n",getpid());
    if (fork())
        fork() && (fork() || fork());
    else
        fork()*fork();
    while(1);
}
```

1. How many processes are created after the execution of the above program?
2. Draw the tree of processes generated.

**Problem 2: (parts A and B are independent)**

**(40 minutes)**

- A. A parent process creates N parallel child processes that have a point of appointment (RDV). The two processes should execute two functions: **Pre\_RDV()** and **RDV()** consecutively. A process arriving at the RDV point (i.e., before executing the function **RDV()**) should wait if there is at least one other process that did not

arrived. The last arriving process wakes up all the blocked processes. Write a solution that solves this problem.

- B. In this part, we suppose that we have a shared data segment that contains a shared variable between **N separated processes** (each process is an independent C program). These processes should access the shared variable and increment it by 1. The shared variable is created by the first process.
- Write the code of the first process that creates the shared data segment and the shared variable and initialize it to 1. This process should maintain the **way of communicating** the (ID) of the shared data segment between the different processes.
  - Write the code of the other processes (all have similar behavior) that should access the shared data segment via its ID and increment the shared variable and print out on the screen the value of the shared variable along with its pid.