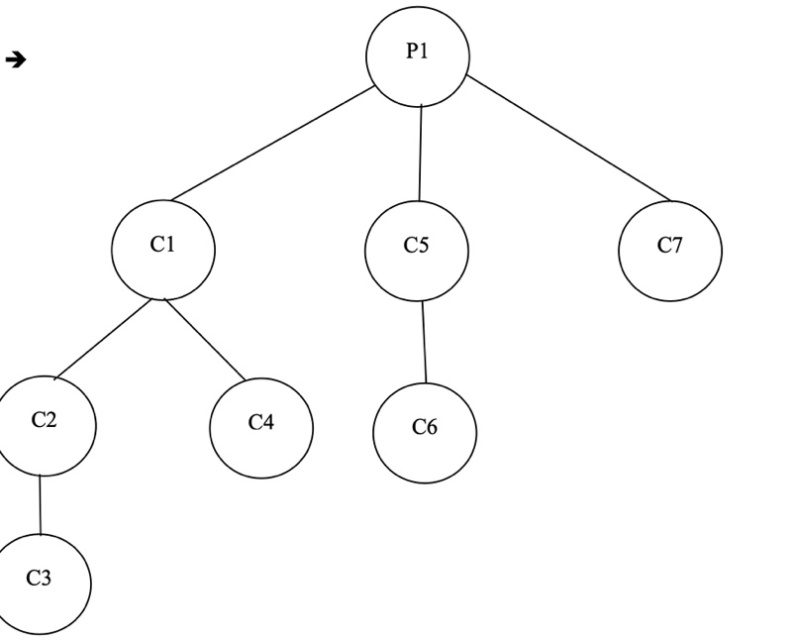
**Q1:**

int main(void)

{  
fork();  
fork();

fork();

}



As per the diagram, there will be total 8 processes, 1 parent process and 7 child processes.

Q2:

int main(void)  
{  
int x = 10, status, childpid;  
int pid1 = fork();  
if (pid1 != 0)  
{  
childpid = wait(&status);  
x = x + WEXITSTATUS(status);

printf("%d",x);  
}  
else  
{  
int pid2 = fork();  
if (pid2 != 0)  
{  
childpid = wait(&status);  
x = x + WEXITSTATUS(status); printf("%d",x);  
exit(x);  
}  
else  
x = x + 5;  
printf("%d",x);

exit(x);

}

}

* There will be one parent and there will be a child, as well as there will be child’s child. So, the output here will be in following sequence: 15, 25 and 35.
* Here, x is declared as 10, and pid1 will create one parent and one child process, but as we see in else part, pid2 will create another child process, so, now moving towards the innermost else part, we can see x = x+5, so here, value for the x printed by innermost child will be 15, and exit(x) will pass the value 15 to its parent, which is actually a child process created by pid1.
* Now, will observing innermost if condition, which is the parent of innermost child process, and it is waiting for innermost child process to terminate, as soon as the innermost child process terminates, it passes 15 to its immediate parent process, which we are observing now, and here there is line x = x + WEXITSTATUS(status), which will retrieve the exit() value returned by its child process, which is 15 in our case, and x will be treated as 10 so now, the x’s value will be 25, which will get printed and passed to its parent process using exit().
* Same process as mentioned in above step will be repeated by parent process of pid1, so WEXITSTATUS(status) here will return 25, which will be added with 10 as mentioned x = x + WEXITSTATUS(status) and it will print 35.