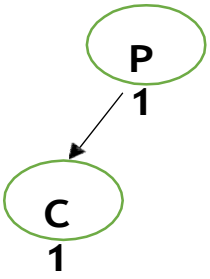
LAB SHEET 6

Process Creation using fork() System call

Write programs to create processes according to the tree structures given below. All processes should print their Process ID and Parent Process ID and the label given in the process.

1.

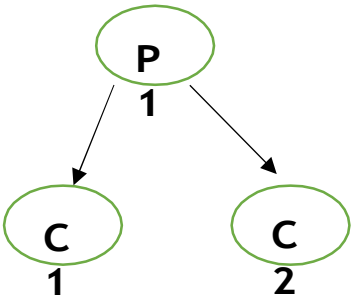



```
IdeaPad-Flex-5-14ALC7:~$ nano tree1.c
IdeaPad-Flex-5-14ALC7:~$ gcc tree1.c -o tree1
IdeaPad-Flex-5-14ALC7:~$ ./tree1
```

```
 GNU nano 6.2                                  tree1.c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    printf("P1: PID=%d, PPID=%d\n", getpid(), getppid());

    pid_t pid = fork();
    if (pid == 0) {
        printf("C1: PID=%d, PPID=%d\n", getpid(), getppid());
    }

    return 0;
}
```

```
P1: PID=5185, PPID=5135
C1: PID=5186, PPID=5185
```

2.

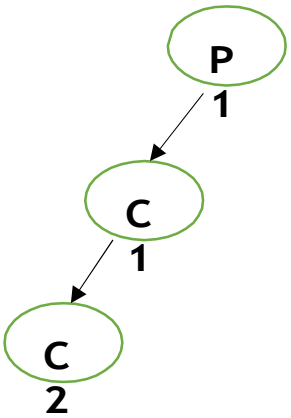



```
IdeaPad-Flex-5-14ALC7:~$ nano tree2.c
IdeaPad-Flex-5-14ALC7:~$ gcc tree2.c -o tree2
IdeaPad-Flex-5-14ALC7:~$ ./tree2
```

```
  GNU nano 6.2                                tree2.c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    printf("P1: PID=%d, PPID=%d\n", getpid(), getppid());

    pid_t pid1 = fork();
    if (pid1 == 0) {
        printf("C1: PID=%d, PPID=%d\n", getpid(), getppid());
    } else {
        pid_t pid2 = fork();
        if (pid2 == 0) {
            printf("C2: PID=%d, PPID=%d\n", getpid(), getppid());
        }
    }

    return 0;
}
```

```
P1: PID=5939, PPID=5135
C1: PID=5940, PPID=5939
C2: PID=5941, PPID=5939
```

3.




```
IdeaPad-Flex-5-14ALC7:~$ nano tree3.c
IdeaPad-Flex-5-14ALC7:~$ gcc tree3.c -o tree3
IdeaPad-Flex-5-14ALC7:~$ ./tree3
```

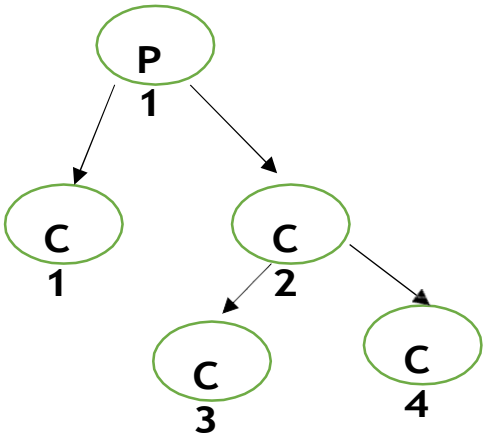
GNU nano 6.2

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    printf("P1: PID=%d, PPID=%d\n", getpid(), getppid());
    pid_t pid1 = fork();
    if (pid1 == 0) {
        printf("C1: PID=%d, PPID=%d\n", getpid(), getppid());
    } else {
        pid_t pid2 = fork();
        if (pid2 == 0) {
            printf("C2: PID=%d, PPID=%d\n", getpid(), getppid());

            pid_t pid3 = fork();
            if (pid3 == 0) {
                printf("C3: PID=%d, PPID=%d\n", getpid(), getppid());
            } else {
                pid_t pid4 = fork();
                if (pid4 == 0) {
                    printf("C4: PID=%d, PPID=%d\n", getpid(), getppid());
                }
            }
        }
    }

    return 0;
}
```

```
P1: PID=6373, PPID=5135
C1: PID=6374, PPID=6373
C2: PID=6375, PPID=6373
```

```
C3: PID=6376, PPID=6375
C4: PID=6377, PPID=6375
```

4.

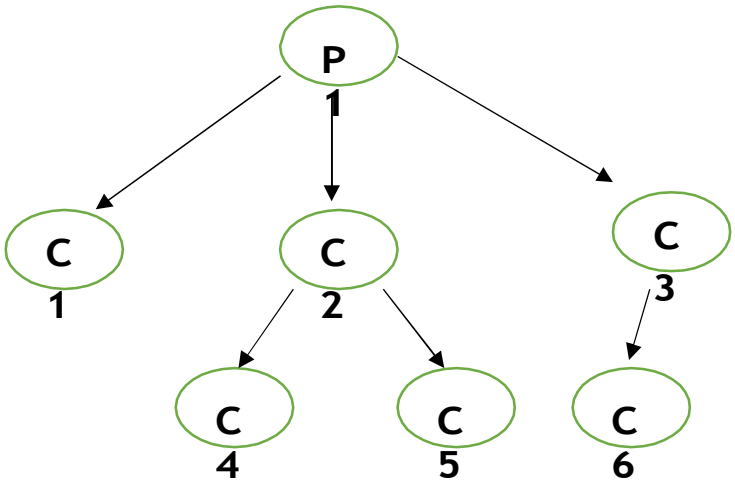



```
IdeaPad-Flex-5-14ALC7:~$ nano tree4.c
IdeaPad-Flex-5-14ALC7:~$ gcc tree4.c -o tree4
IdeaPad-Flex-5-14ALC7:~$ ./tree4
```

```
  GNU nano 6.2
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    printf("P1: PID=%d, PPID=%d\n", getpid(), getppid());

    pid_t pid1 = fork();
    if (pid1 == 0) {
        printf("C1: PID=%d, PPID=%d\n", getpid(), getppid());
    } else {
        pid_t pid2 = fork();
        if (pid2 == 0) {
            printf("C2: PID=%d, PPID=%d\n", getpid(), getppid());
        }
    }

    return 0;
}
```

```
P1: PID=6839, PPID=5135
C1: PID=6840, PPID=6839
C2: PID=6841, PPID=6839
```

5.




```
-IdeaPad-Flex-5-14ALC7:~$ nano tree5.c
-IdeaPad-Flex-5-14ALC7:~$ gcc tree5.c -o tree5
-IdeaPad-Flex-5-14ALC7:~$ ./tree5
```

```
  GNU nano 6.2
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    printf("P1: PID=%d, PPID=%d\n", getpid(), getppid());

    for (int i = 1; i <= 6; i++) {
        pid_t pid = fork();
        if (pid == 0) {
            printf("C%d: PID=%d, PPID=%d\n", i, getpid(), getppid());
            break; // Prevent child from creating more children
        }
    }

    return 0;
}
```

```
P1: PID=7152, PPID=5135
C1: PID=7153, PPID=7152
C2: PID=7154, PPID=7152
C3: PID=7155, PPID=7152
C4: PID=7156, PPID=7152
C5: PID=7157, PPID=7152
C6: PID=7158, PPID=7152
```