1. LINUX operating system. Which commands will he use to complete the given task with the help of the following operation?

●Kill processes by name

●Kill a process based on the process name

●Kill a single process at a time with the given process ID

```
Barkha@DESKTOP-35GK287 MSYS ~
$ sleep 500 &
[1] 1207

Barkha@DESKTOP-35GK287 MSYS ~
$ killall sleep
[1]+  Terminated                  sleep 500
```

```
Barkha@DESKTOP-35GK287 MSYS ~
$ sleep 500 &
[2] 1252

Barkha@DESKTOP-35GK287 MSYS ~
$ pkill sle
[1]-  Terminated                  sleep 500
[2]+  Terminated                  sleep 500
```

```
Barkha@DESKTOP-35GK287 MSYS ~
$ sleep 300 &
[1] 1254

Barkha@DESKTOP-35GK287 MSYS ~
$ kill 1254

Barkha@DESKTOP-35GK287 MSYS ~
$
[1]+  Terminated                  sleep 300
```

2 .Write a program for process creation using C

●Orphan Process

●Zombie Process

orphan process

```
~
  GNU nano 8.7                                                    hello.c
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid > 0) {
        // Parent process
        printf("Parent process exiting\n");
    } else {
        // Child process
        sleep(5);
        printf("Child process becomes orphan\n");
        printf("PID: %d, PPID: %d\n", getpid(), getppid());
    }
    return 0;
}
```

```
Barkha@DESKTOP-35GK287 MSYS ~
$ nano hello.c

Barkha@DESKTOP-35GK287 MSYS ~
$ gcc hello.c -o hello

Barkha@DESKTOP-35GK287 MSYS ~
$ ./hello
Parent process exiting

Barkha@DESKTOP-35GK287 MSYS ~
$ Child process becomes orphan
PID: 1262, PPID: 1
$
```

zombie Process

~

```
  GNU nano 8.7                                                hello2.c
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid == 0) {
        // child process
        printf("Child process exiting\n");
    } else {
        // Parent process
        sleep(10); // Parent does not call wait()
        printf("Parent process running\n");
    }
    return 0;
}
```

```
Barkha@DESKTOP-35GK287 MSYS ~
$ nano hello2.c

Barkha@DESKTOP-35GK287 MSYS ~
$ gcc hello2.c -o hello2

Barkha@DESKTOP-35GK287 MSYS ~
$ ./hello2
Child process exiting
Parent process running

Barkha@DESKTOP-35GK287 MSYS ~
$ ps -ef | grep Z
  Barkha    1273    1089 pty1     14:29:11 grep Z

Barkha@DESKTOP-35GK287 MSYS ~
$
```

3. Create the process using fork () system call.

●Child Process creation

●Parent process creation

●PPID and PID

~

```
  GNU nano 8.7                                                hello3.c
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid;

    pid = fork();

    if (pid == 0) {
        // child process
        printf("child Process\n");
        printf("PID = %d\n", getpid());
        printf("PPID = %d\n", getppid());
    } else {
        // Parent process
        printf("Parent Process\n");
        printf("PID = %d\n", getpid());
        printf("Child PID = %d\n", pid);
    }
    return 0;
}
```

```
Barkha@DESKTOP-35GK287 MSYS ~
$ nano hello3.c

Barkha@DESKTOP-35GK287 MSYS ~
$ gcc hello3.c -o hello3

Barkha@DESKTOP-35GK287 MSYS ~
$ ./hello3
child Process
PID = 1281
PPID = 1280
Parent Process
PID = 1280
Child PID = 1281

Barkha@DESKTOP-35GK287 MSYS ~
$
```