OS HOME TASKS:

1. **Run the code discussed in class (from the book).**

```c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
int main() {
    pid_t pid = fork();

    if (pid < 0) {
        perror("Fork failed");
        exit(1);
    }
    else if (pid == 0) {
        printf("Child Process: PID = %d, Parent PID = %d\n", getpid(), getppid());
        execlp("ps", "ps", "-f", "--forest", NULL);
    }
    else {
        printf("Parent Process: PID = %d, Child PID = %d\n", getpid(), pid);
        wait(NULL);
    }

    return 0;
}
```

```
k200281kainat@k200281kainat-VirtualBox:~$ touch OShometask1.c
k200281kainat@k200281kainat-VirtualBox:~$ gedit OShometask1.c
k200281kainat@k200281kainat-VirtualBox:~$ gcc OShometask1.c -o out
k200281kainat@k200281kainat-VirtualBox:~$ ./out
Parent Process: PID = 3279, Child PID = 3280
Child Process: PID = 3280, Parent PID = 3279
UID          PID    PPID  C STIME TTY          TIME CMD
k200281+    2969    2962  0 19:16 pts/2    00:00:00 bash
k200281+    3279    2969  0 19:31 pts/2    00:00:00  \_ ./out
k200281+    3280    3279  0 19:31 pts/2    00:00:00      \_ ps -f --forest
k200281kainat@k200281kainat-VirtualBox:~$ gedit OShometask1.c
```

2. **Write a program to create the following hierarchy of processes:**
    - A parent process creates a child process, which in turn creates another child process, forming a linear hierarchy.
    - Use getpid() to get the process ID.
    - Display the IDs of all processes.

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

int main() {
    pid_t pid1, pid2;

    pid1 = fork();
    if (pid1 < 0) {
        perror("Fork failed");
        exit(1);
    }
    if (pid1 == 0) {

        printf("Child 1: PID = %d, Parent PID = %d\n", getpid(), getppid());

        pid2 = fork();
        if (pid2 < 0) {
            perror("Fork failed");
            exit(1);
        }
        if (pid2 == 0) {
            printf("Child 2: PID = %d, Parent PID = %d\n", getpid(), getppid());
        }
        else {
            wait(NULL);
        }
    }
    else {
        wait(NULL);
        printf("Parent: PID = %d\n", getpid());
    }

    return 0;
}
```

```
k200281kainat@k200281kainat-VirtualBox:~$ gedit OShometask2.c
k200281kainat@k200281kainat-VirtualBox:~$ gcc OShometask2.c -o out
k200281kainat@k200281kainat-VirtualBox:~$ ./out
Child 1: PID = 3053, Parent PID = 3052
Child 2: PID = 3054, Parent PID = 3053
Parent: PID = 3052
k200281kainat@k200281kainat-VirtualBox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
k200281+    2969    2962  0 19:16 pts/2    00:00:00 bash
k200281+    3055    2969  0 19:19 pts/2    00:00:00 ps -f
k200281kainat@k200281kainat-VirtualBox:~$ ps -f --forest
UID          PID    PPID  C STIME TTY          TIME CMD
k200281+    2969    2962  0 19:16 pts/2    00:00:00 bash
k200281+    3062    2969  0 19:20 pts/2    00:00:00  \_ ps -f --forest
```

3. **Implement the following Process Tree Diagram:**

   - One parent process creates two child processes.
   - One child prints even numbers, and the other prints odd numbers.
   - Use wait() to synchronize them.
   - Verify the order of values printed by running the program multiple times.

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

void print_even() {
    for (int i = 0; i <= 10; i += 2) {
        printf("Even: %d (PID: %d)\n", i, getpid());
        usleep(100000);
    }
}
void print_odd() {
    for (int i = 1; i <= 10; i += 2) {
        printf("Odd: %d (PID: %d)\n", i, getpid());
        usleep(100000);
    }
}

int main() {
    pid_t pid1, pid2;

    pid1 = fork();
    if (pid1 < 0) {
        perror("Fork failed");
        exit(1);
    }
    if (pid1 == 0) {
        print_even();
        exit(0);
    }

    pid2 = fork();
    if (pid2 < 0) {
        perror("Fork failed");
        exit(1);
    }
    if (pid2 == 0) {
        print_odd();
        exit(0);
    }
    wait(NULL);
    wait(NULL);

    printf("Parent process (PID: %d) finished execution.\n", getpid());
    return 0;
}
```

```
k200281kainat@k200281kainat-VirtualBox:~$ gedit OShometask3.c
k200281kainat@k200281kainat-VirtualBox:~$ gedit OShometask3.c
k200281kainat@k200281kainat-VirtualBox:~$ gcc OShometask3.c -o out
k200281kainat@k200281kainat-VirtualBox:~$ ./out
Odd: 1 (PID: 3406)
Even: 0 (PID: 3405)
Even: 2 (PID: 3405)
Odd: 3 (PID: 3406)
Even: 4 (PID: 3405)
Odd: 5 (PID: 3406)
Even: 6 (PID: 3405)
Odd: 7 (PID: 3406)
Odd: 9 (PID: 3406)
Even: 8 (PID: 3405)
Even: 10 (PID: 3405)
Parent process (PID: 3404) finished execution.
```

PIDS ANALYSIS:

```c
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
int main()
{
pid_t pid,pid1,pid2;
pid = fork();
if(pid<0)
{
fprintf(stderr, "Fork failed");
return 1;
}
else if (pid == 0)
{
pid1 = getpid();
printf("Child pid: %d\n",pid);
printf("Child pid: %d\n",pid1);
}
else
{
wait(NULL);
pid1 = getpid();
printf("Parent pid: %d\n",pid);
printf("Parent pid: %d\n",pid1);
printf("Child complete");
}
return 0;
}
```

```
k200281kainat@k200281kainat-VirtualBox:~$ gcc idscode.c -o out
k200281kainat@k200281kainat-VirtualBox:~$ ./out
Child pid: 0
Child pid: 2520
Parent pid: 2520
Parent pid: 2519
Child completek200281kainat@k200281kainat-VirtualBox:~$ gedit idscode.c
```