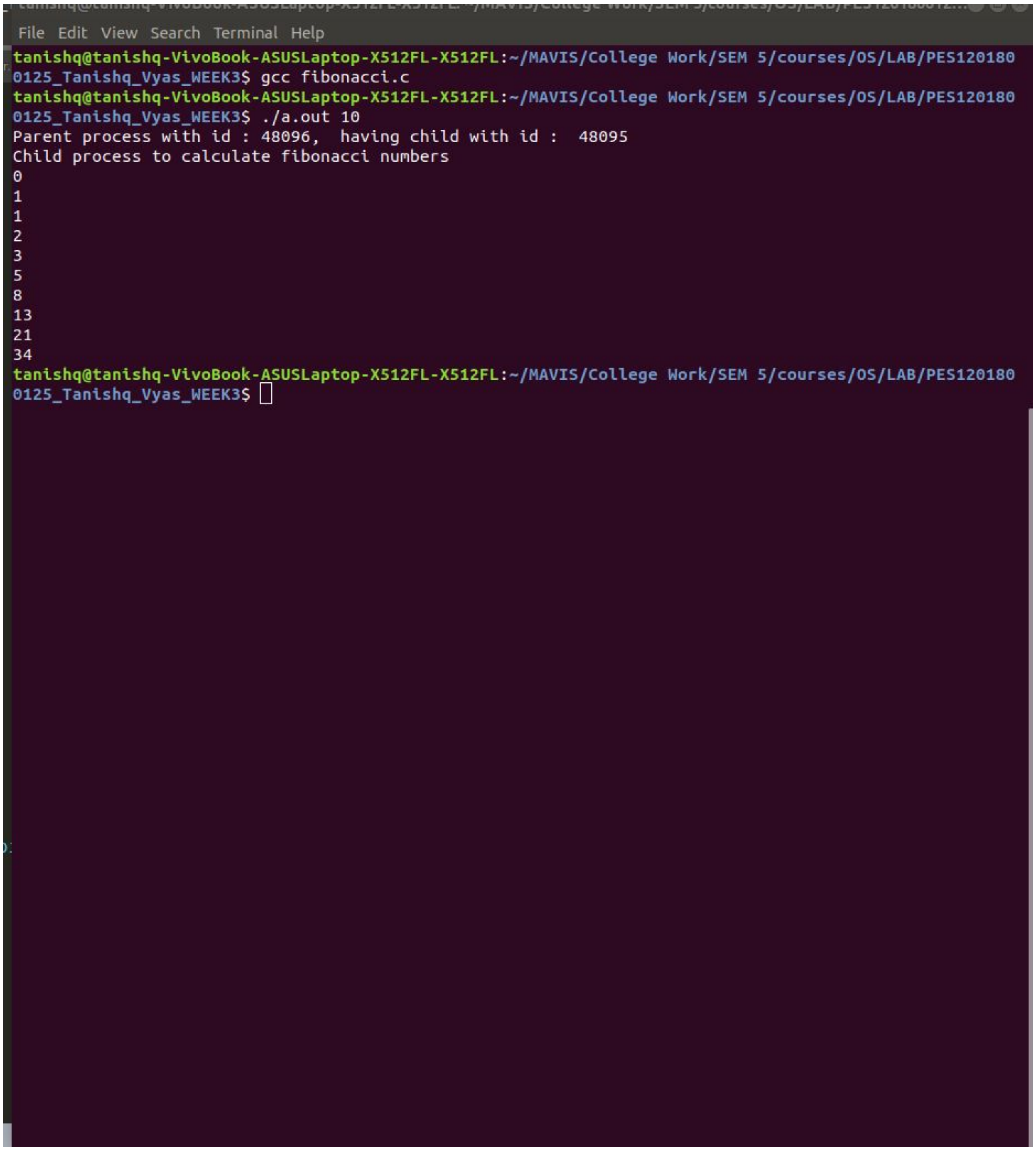
Operating System Week-3

Name	: Tanishq Vyas
SRN	: PES1201800125
Section	: H
Semester	: 5
Branch	: CSE

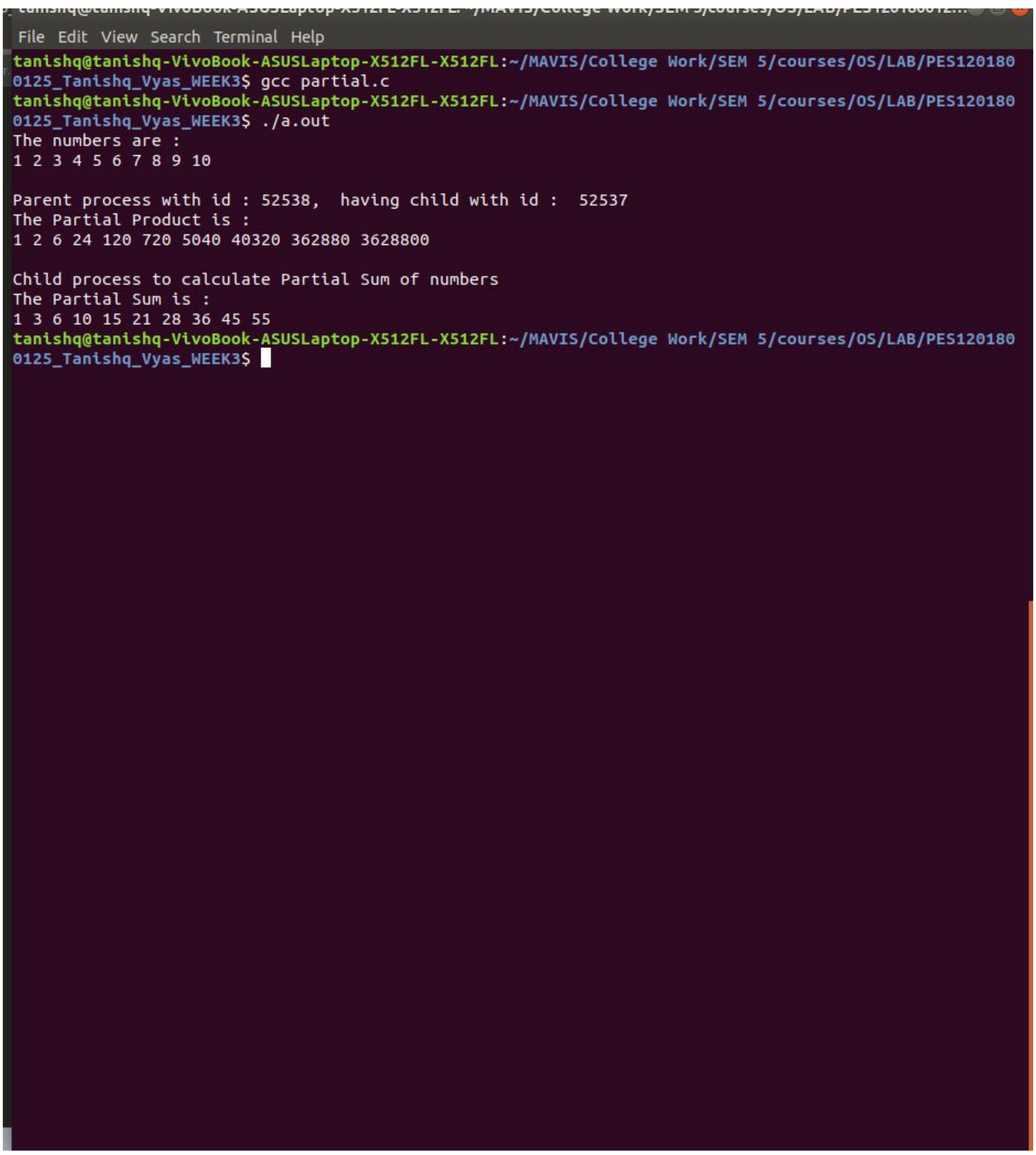
Task 1



File Edit View Search Terminal Help

```
tanishq@tanishq-VivoBook-ASUSLaptop-X512FL-X512FL:~/MAVIS/College Work/SEM 5/courses/OS/LAB/PES120180
0125_Tanishq_Vyas_WEEK3$ gcc fibonacci.c
tanishq@tanishq-VivoBook-ASUSLaptop-X512FL-X512FL:~/MAVIS/College Work/SEM 5/courses/OS/LAB/PES120180
0125_Tanishq_Vyas_WEEK3$ ./a.out 10
Parent process with id : 48096,  having child with id :  48095
Child process to calculate fibonacci numbers
0
1
1
2
3
5
8
13
21
34
tanishq@tanishq-VivoBook-ASUSLaptop-X512FL-X512FL:~/MAVIS/College Work/SEM 5/courses/OS/LAB/PES120180
0125_Tanishq_Vyas_WEEK3$ 
```

Task 2

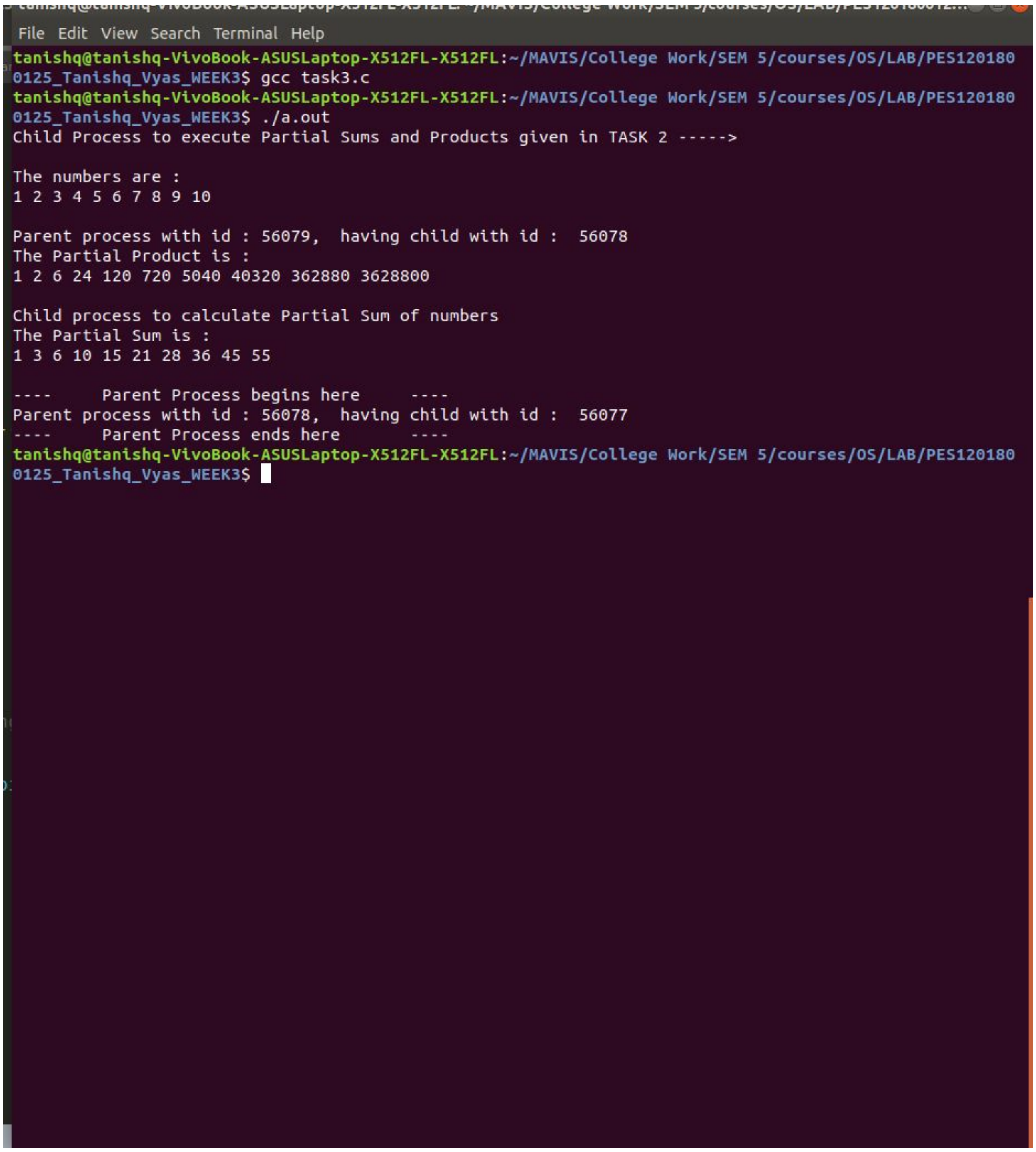


```
tanishq@tanishq-VivoBook-ASUSLaptop-X512FL-X512FL:~/MAVIS/College Work/SEM 5/courses/OS/LAB/PES120180
0125_Tanishq_Vyas_WEEK3$ gcc partial.c
tanishq@tanishq-VivoBook-ASUSLaptop-X512FL-X512FL:~/MAVIS/College Work/SEM 5/courses/OS/LAB/PES120180
0125_Tanishq_Vyas_WEEK3$ ./a.out
The numbers are :
1 2 3 4 5 6 7 8 9 10

Parent process with id : 52538,  having child with id :  52537
The Partial Product is :
1 2 6 24 120 720 5040 40320 362880 3628800

Child process to calculate Partial Sum of numbers
The Partial Sum is :
1 3 6 10 15 21 28 36 45 55
tanishq@tanishq-VivoBook-ASUSLaptop-X512FL-X512FL:~/MAVIS/College Work/SEM 5/courses/OS/LAB/PES120180
0125_Tanishq_Vyas_WEEK3$ 
```

Task 3



```
tanishq@tanishq-VivoBook-ASUSLaptop-X512FL-X512FL:~/MAVIS/College Work/SEM 5/courses/OS/LAB/PES120180
0125_Tanishq_Vyas_WEEK3$ gcc task3.c
tanishq@tanishq-VivoBook-ASUSLaptop-X512FL-X512FL:~/MAVIS/College Work/SEM 5/courses/OS/LAB/PES120180
0125_Tanishq_Vyas_WEEK3$ ./a.out
Child Process to execute Partial Sums and Products given in TASK 2 ----->

The numbers are :
1 2 3 4 5 6 7 8 9 10

Parent process with id : 56079,  having child with id :  56078
The Partial Product is :
1 2 6 24 120 720 5040 40320 362880 3628800

Child process to calculate Partial Sum of numbers
The Partial Sum is :
1 3 6 10 15 21 28 36 45 55

----     Parent Process begins here     ----
Parent process with id : 56078,  having child with id :  56077
----     Parent Process ends here       ----
tanishq@tanishq-VivoBook-ASUSLaptop-X512FL-X512FL:~/MAVIS/College Work/SEM 5/courses/OS/LAB/PES120180
0125_Tanishq_Vyas_WEEK3$
```

Answers :

1. When a process is terminated, it briefly moves to the zombie state and remains in that state until the parent invokes a call to wait(). When this occurs, the process id as well as entry in the process table are both released. However, if a parent does not invoke wait(), the child process remains a zombie as long as the parent remains alive. Once the parent process terminates, the init process becomes the new parent of the zombie (child process). Periodically, the init process calls wait() which ultimately releases the pid and entry in the process table of the zombie process. Thus it helps in clearing up the executed process in case the person never called the wait() in the parent process. Thus preventing the system from stop functioning in case of persistent zombie processes.

2. A subreaper fulfills the role of init(1) for its descendant processes. When a process becomes orphaned (i.e., its immediate parent terminates) then that process will be reparented to the nearest still living ancestor which is the **subreaper process**. Thus a subreaper process becomes the parent of a child process in case the process is orphaned thus helps in clearing (reaping) the process after termination.

3. A defunct process or a zombie process is a process who terminates but does not get reaped by its aprent process, thus it enters a zombie state. Eventually it is reaped by the init process when the parent terminates.
In order to avoid the defunct process, the parent process must reap the child process using **wait** or **waitpid** or it can use `local $SIG{CHLD} = 'IGNORE';`. This helps us in preventing the zombie process or defunct process.

4. We can easily identify the zombie process in a Linux system with the help of **ps** command. The status column for the process has **Z** as the value and they often have the words <defunct> in the CMD column as well.

5. A child process inherits most of its attributes, such as file descriptors, from its parent. In Unix, a child process is typically created as a copy of the parent, using the fork system call. The child process can then overlay itself with a different program (using exec) as required.