****

**Term Work**

**On**

**OPERATING SYSTEM**

**(PCS 506)**

**Submitted to: Submitted by:**

Dr. Pardeep Singh Prahlad Singh Aswal  
Associate Professor University Roll. No.: 2018550  
GEHU, D. Dun Class Roll No./Section: 39/A

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GRAPHIC ERA HILL UNIVERSITY, DEHRADUN**

A picture containing logo

Description automatically generated

**DEPARTMENT OF CSE**

|  |
| --- |
| Photograph  Passport Size |

**STUDENT LAB REPORT SHEET**

**Name of Student ....................................................………Mob. No ...............................**

**Address Permanent .....................................................................................................**

**Father’s Name ................................. Occupation .................. Mob. No .......................**

**Mother’s Name ........................... Occupation ...................... Mob. No .......................**

**Section ............ Branch ............ Semester ............ Class Roll No ............ Grade A B C**

**Local Address .....................................................Email ............................................ Marks 5 3 1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No.** | **Practical** | **D.O.P.** | **Date of Submission** | **Grade (Viva)** | **Grade (Report File)** | **Total Marks (out of 10)** | **Student’s Signature** | **Teacher’s Signature** |
| **1** |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |
| **5** |  |  |  |  |  |  |  |  |
| **6** |  |  |  |  |  |  |  |  |
| **7** |  |  |  |  |  |  |  |  |
| **8** |  |  |  |  |  |  |  |  |
| **9** |  |  |  |  |  |  |  |  |
| **10** |  |  |  |  |  |  |  |  |
| **11** |  |  |  |  |  |  |  |  |
| **12** |  |  |  |  |  |  |  |  |

**PRACTICAL 1**

**Question:** Write a C program to implement first come first serve scheduling algorithm.

**About Fork() function:**

We use the fork() system call to create a new process from the calling process by duplicating it. fork() system call is used to create child processes in a C program. fork() is used where parallel processing is required in application. The fork() system function is defined in the headers **unistd.h**.

**PID: -** Process ID

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

typedef struct FCFS\_scheduling

{

int at, bt, ct, tat; // arrival,burst,completion,turnaround time

int wt, rt, st; // waiting,response,start time

} fcfs;

int comparator(const void \*num1, const void \*num2)

{

struct FCFS\_scheduling \*a = (struct FCFS\_scheduling \*)num1;

struct FCFS\_scheduling \*b = (struct FCFS\_scheduling \*)num2;

if (a->at >= b->at)

return 1;

else

return -1;

}

int main()

{

int n;

printf("\nEnter the total number of Processs: ");

scanf("%d", &n);

fcfs arr[n];

for (int i = 0; i < n; i++)

{

printf("Enter Arrival time and Burst time respectively for process %d : ", i + 1);

// arr[i].pid = i + 1;

scanf("%d%d", &arr[i].at, &arr[i].bt);

}

qsort(arr, n, sizeof(fcfs), comparator);

arr[0].ct = arr[0].at + arr[0].bt;

arr[0].tat = arr[0].ct - arr[0].at;

arr[0].wt = arr[0].tat - arr[0].bt;

arr[0].rt = arr[0].ct - arr[0].bt;

float avg\_tat = arr[0].tat, avg\_wt = arr[0].wt;

float avg\_rt = arr[0].rt;

float cycle\_length;

for (int i = 1; i < n; i++)

{

if (arr[i].at >= arr[i - 1].ct)

{

arr[i].ct = arr[i].at + arr[i].bt;

}

else

arr[i].ct = arr[i - 1].ct + arr[i].bt;

}

for (int i = 1; i < n; i++)

{

arr[i].tat = arr[i].ct - arr[i].at;

arr[i].wt = arr[i].tat - arr[i].bt;

arr[i].rt = arr[i].ct - arr[i].bt;

}

int idt = 0; // idle time variable

for (int i = 1; i < n; i++)

{

int x;

avg\_tat += arr[i].tat;

avg\_wt += arr[i].wt;

avg\_rt += arr[i].rt;

if (arr[i].at > arr[i - 1].at)

{

x = arr[i].at - arr[i - 1].at;

idt += x;

}

}

// printing table

printf(" PID\tAT\tBT\tCT\tTAT\tWT\tRT\n");

for (int j = 0; j < n; j++)

{

printf(" P%d\t%d\t%d\t%d\t%d\t%d\t%d\n", j + 1, arr[j].at, arr[j].bt, arr[j].ct, arr[j].tat, arr[j].wt, arr[j].rt);

}

cycle\_length = arr[n - 1].ct;

float cpu\_util = (idt / cycle\_length) \* 100;

float tp = (float)n / (cycle\_length); // Throughput

// printing all values

printf("Average turn around time of all processes is: %f\n", avg\_tat / (float)n);

printf("Average waiting time of total processes is: %f\n", avg\_wt / (float)n);

printf("Average response time of total processes is: %f\n\n", avg\_rt / (float)n);

printf("CPU idle time is: %d\n", idt);

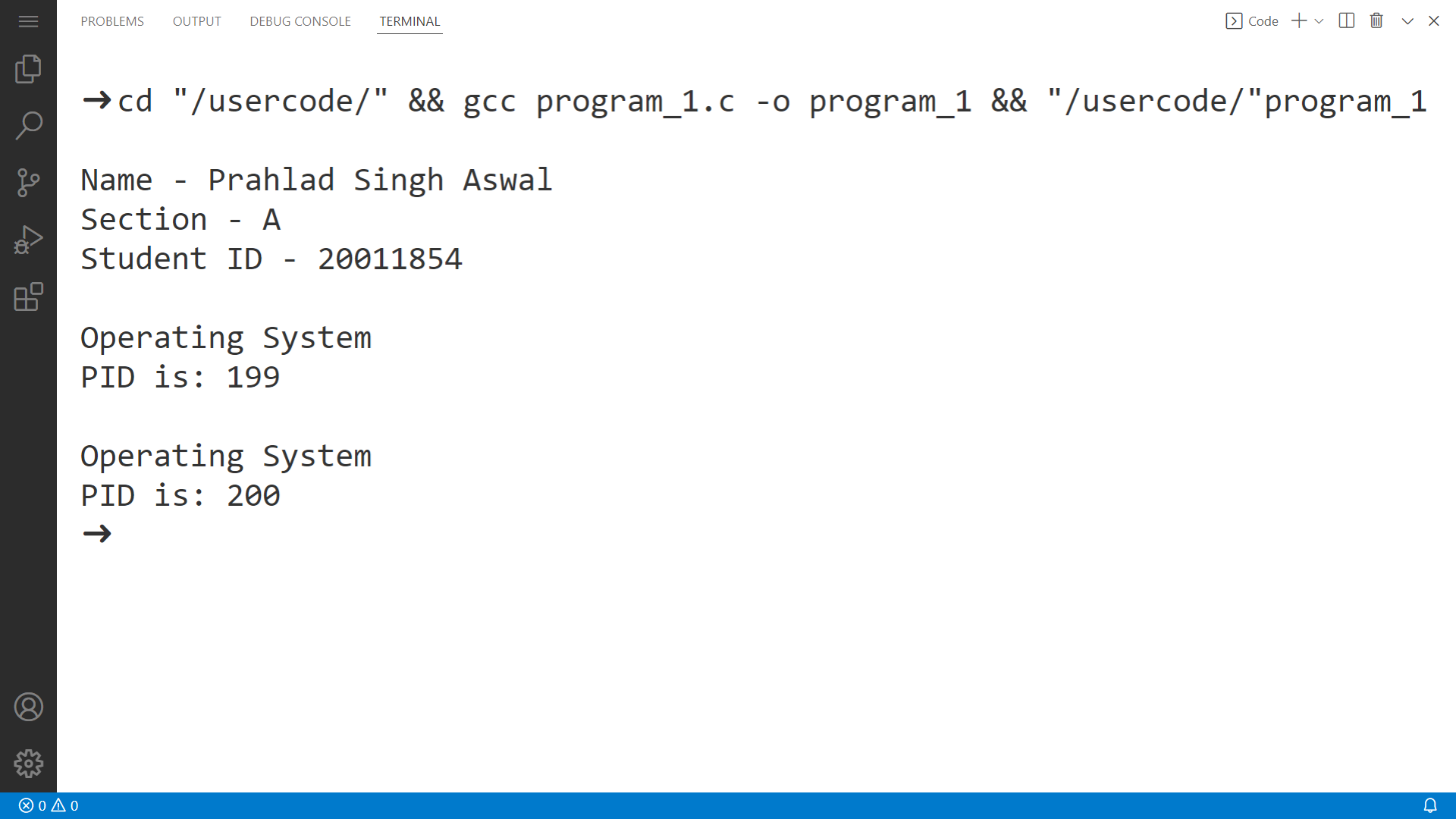
printf("CPU utilization(in percentage) is: %.2f \n", (float)cpu\_util);

printf("Throughput of all processes is: %f\n\n", tp);

return 0;

}

**Output**

****

**PRACTICAL 2**

**Question:** Write a C program in which parent process computes the sum of even numbers and child process computes the sum of odd number stored in an array using a fork().

**About Fork( ) Function:**

Fork system call is used for creating a new process, which is called child process, which runs concurrently with the process that makes the fork() call (parent process). After a new child process is created, both processes will execute the next instruction following the fork() system call.

In this program, parent process computes the sum of even numbers in array and child process computer the sum of odd numbers in the array.

**Source Code:-**

#include <stdio.h>

#include <unistd.h>

int main()

{

printf("\nName - Prahlad Singh Aswal\nSection - A\n");

printf("Student ID - 20011854\n\n");

int n;

int e\_sum = 0, o\_sum = 0;

printf("\nEnter the size of the array: ");

scanf("%d", &n);

int arr[n];

printf("Enter the elements of the array: ");

for (int i = 0; i < n; i++)

scanf("%d", &arr[i]);

int pid = fork();

int i = 0;

if (pid == 0)

{

while (i < n)

{

if (arr[i] % 2 != 0)

o\_sum += arr[i];

i++;

}

printf("\nSum of all odd numbers in array = %d\n\n", o\_sum);

}

else

{

while (i < n)

{

if (arr[i] % 2 == 0)

e\_sum += arr[i];

i++;

}

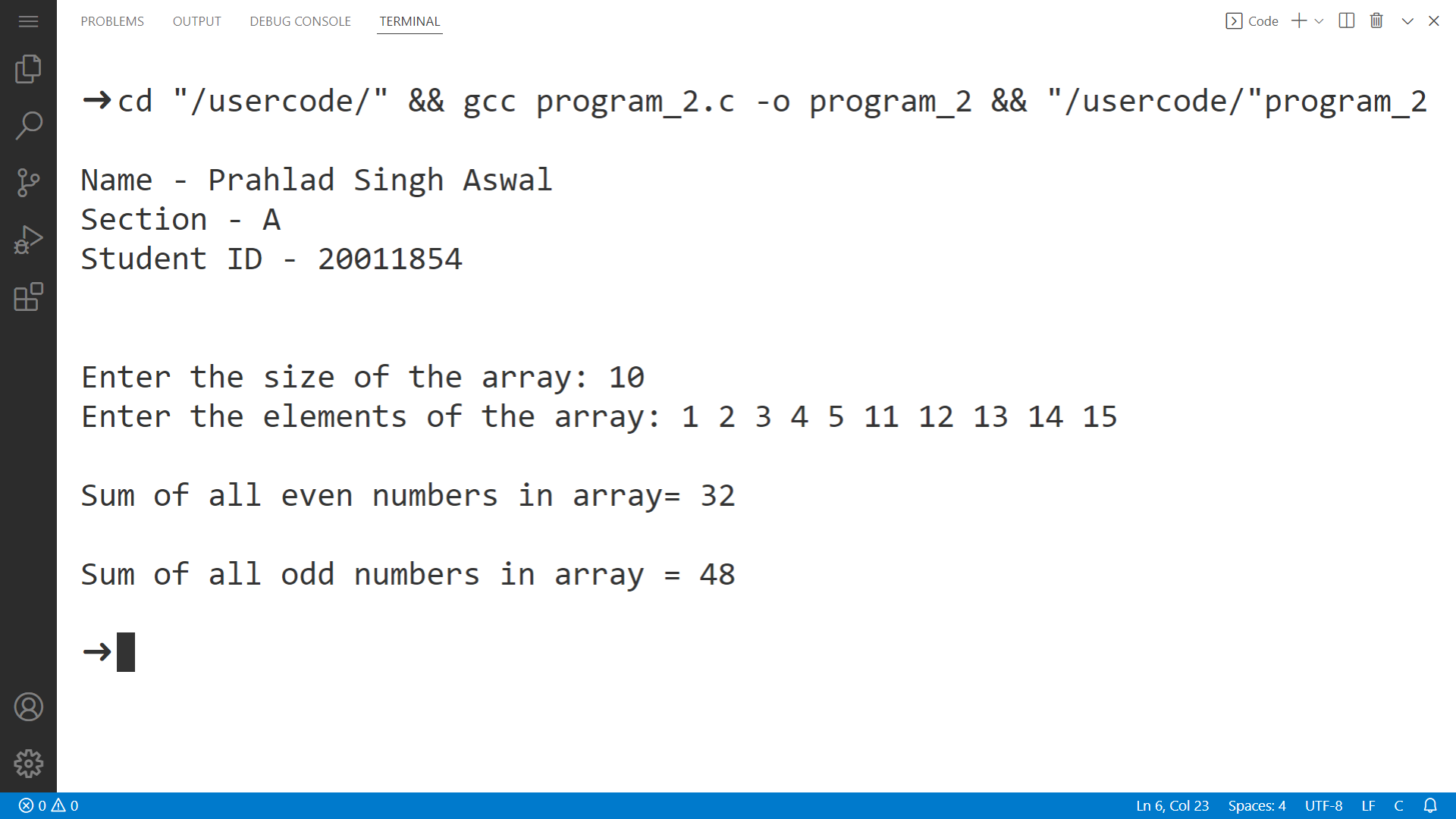
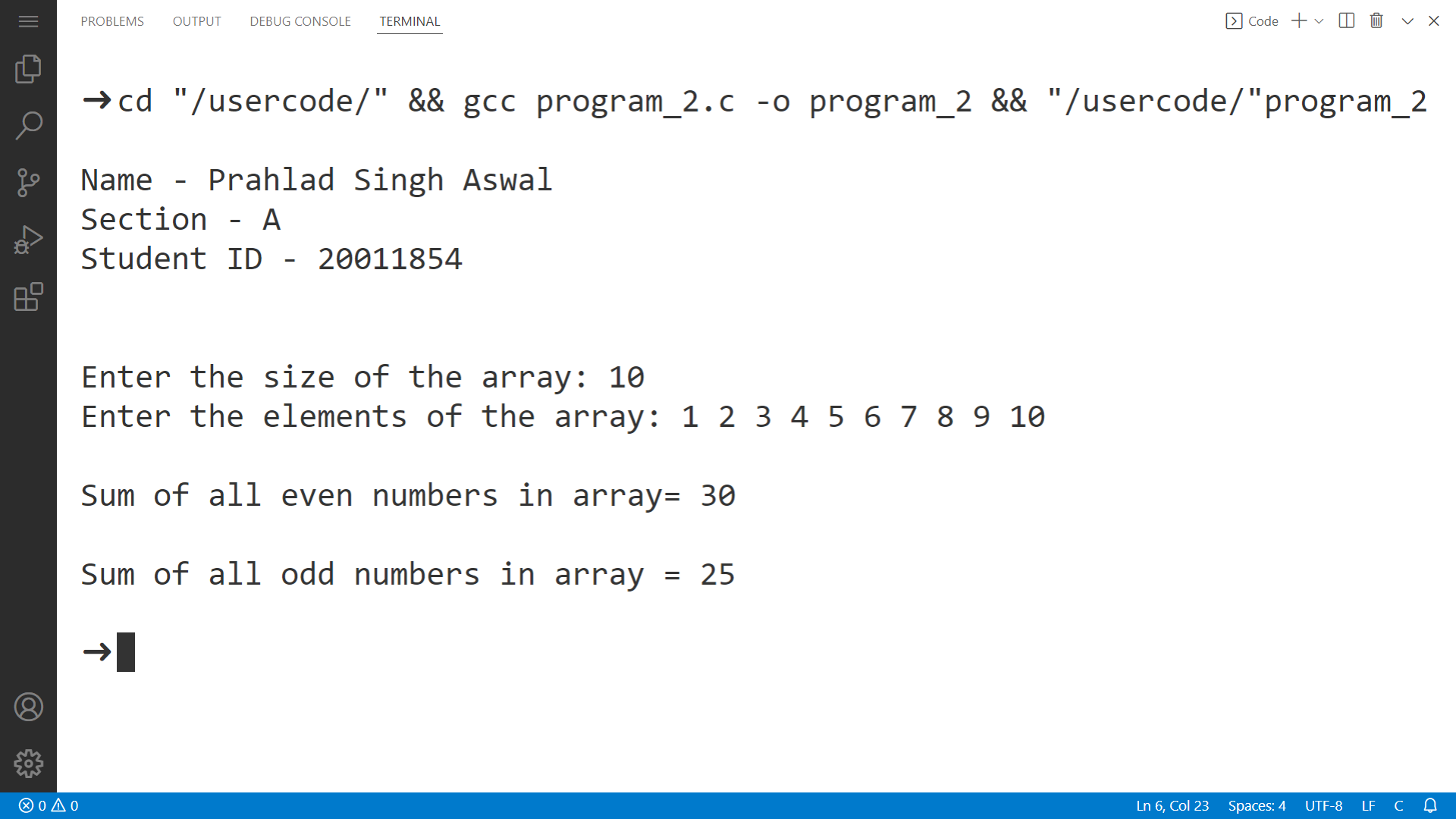
printf("\nSum of all even numbers in array= %d\n", e\_sum);

}

return 0;

}

**Output**

****

**PRACTICAL 3**

**Question:** Write a C program to demonstrate Zombie Process.

**About Zombie Process:**

A zombie process is a process in its terminated state. This usually happens in a program that has parent-child functions. After a child function has finished execution, it sends an exit status to its parent function.

**Source Code:-**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main()

{

printf("\nName - Prahlad Singh Aswal\nSection - A\n");

printf("Student id - 20011854\n\n");

printf("\nZombie Process\n");

int pid = fork();

// PARENT PROCESS

if (pid > 0)

{

sleep(10);

printf("\nParent Process\n");

printf("PID of Parent process is: %d\n", getpid());

}

// CHILD PROCESS

else

{

printf("\nChild Process\n");

printf("PID of Child process is: %d\n", getpid());

printf("PID of the parent process is: %d\n", getppid());

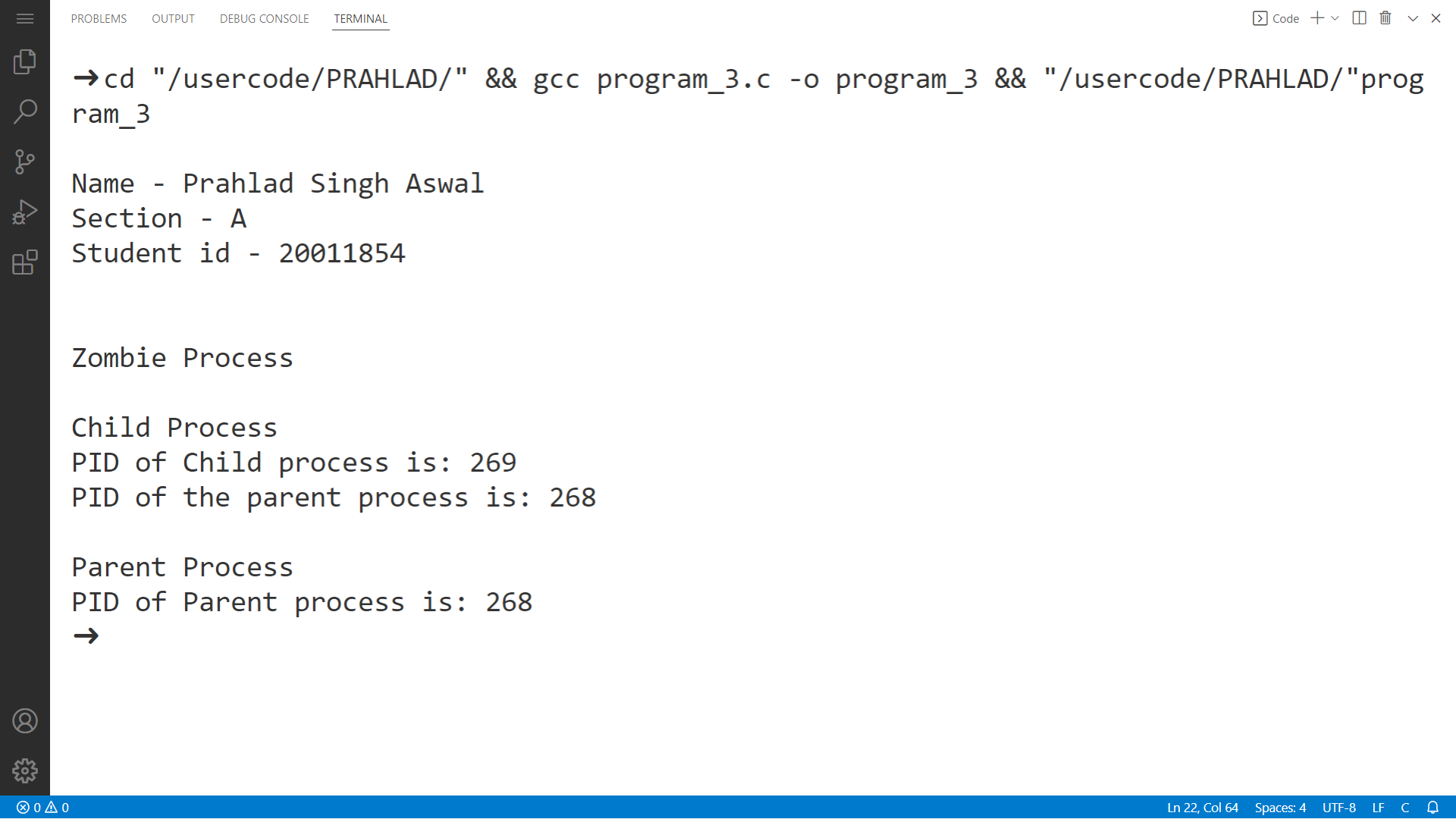
exit(0);

}

return 0;

}

**Output**

****

**PRACTICAL 4**

**Question:** Write a C program to demonstrate Orphan Process.

**About Orphan Process:**

A process whose parent process no more exists i.e. either finished or terminated without waiting for its child process to terminate is called an orphan process.  Orphan process can be orphaned intentionally or unintentionally.

**Source Code:-**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main()

{

printf("\nName - Prahlad Singh Aswal\nSection - A\n");

printf("Student id - 20011854\n\n");

printf("\nOrphan Process\n");

int pid = fork();

// PARENT PROCESS

if (pid > 0)

{

printf("\nParent Process\n");

printf("PID of process is: %d\n", getpid());

exit(0);

}

//CHILD PROCESS

else

{

sleep(10);

printf("\nChild Process\n");

printf("PID of process is: %d\n", getpid());

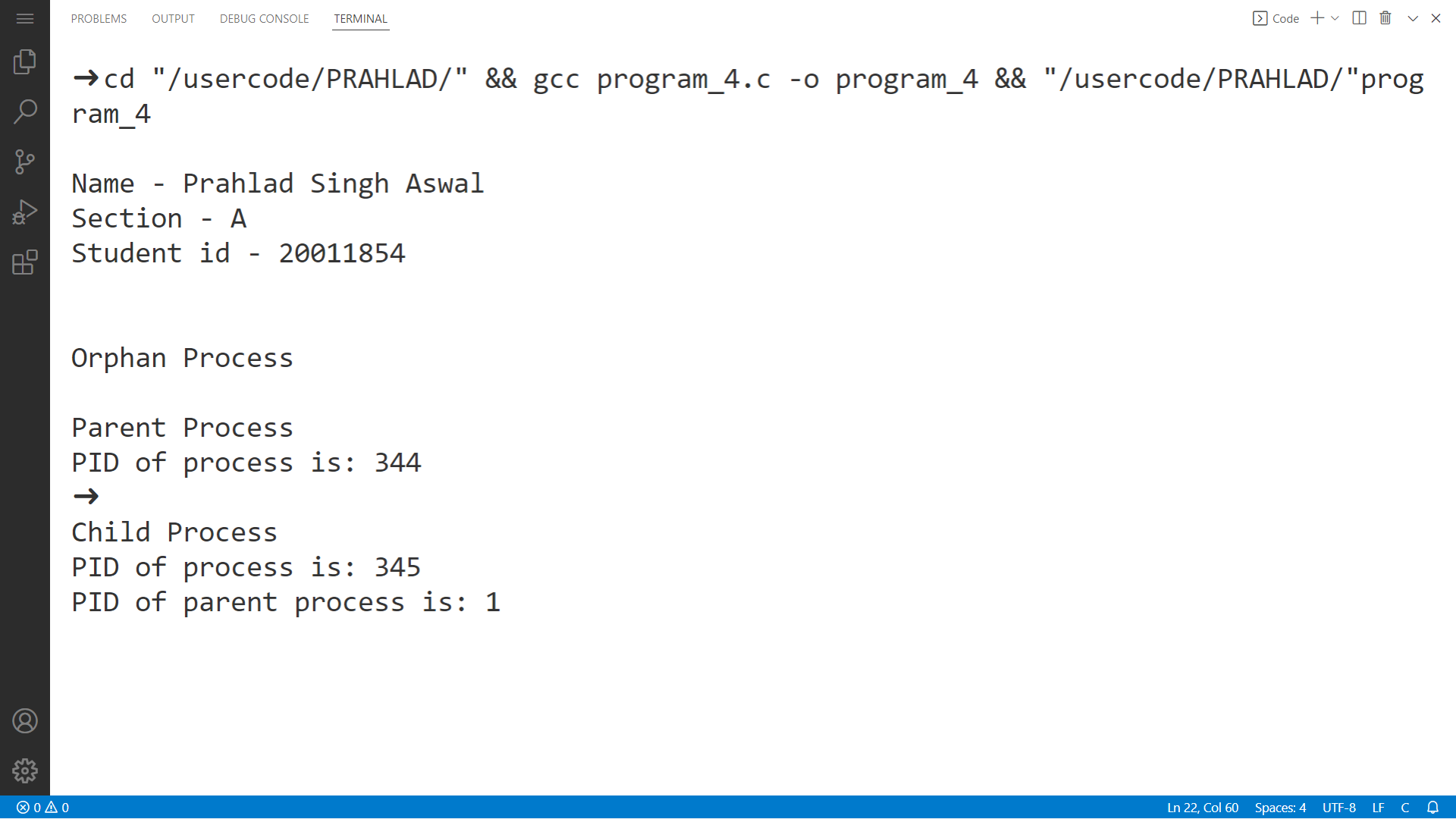
printf("PID of parent process is: %d\n",getppid());

}

return 0;

}

**Output**

****