FINAL OERATING SYSTEM WORKSHEET- 4

Student Name:- Neha Sharma **UID:-**20BCS4576
Branch: - (CSE_IOT) **Section/Group: -** 20BIT (A)
Semester: 3rd **Date of Performance: -** 7/12/2021
Subject Name:-Operating System Lab

A. Implement system calls – fork, exec, getpid, exit.

1. Aim/Overview of the practical:- . Implement system calls – fork, exec, getpid, exit.

2. Task to be done:- We have to implement the program using system calls: fork, exec, getpid, exit.

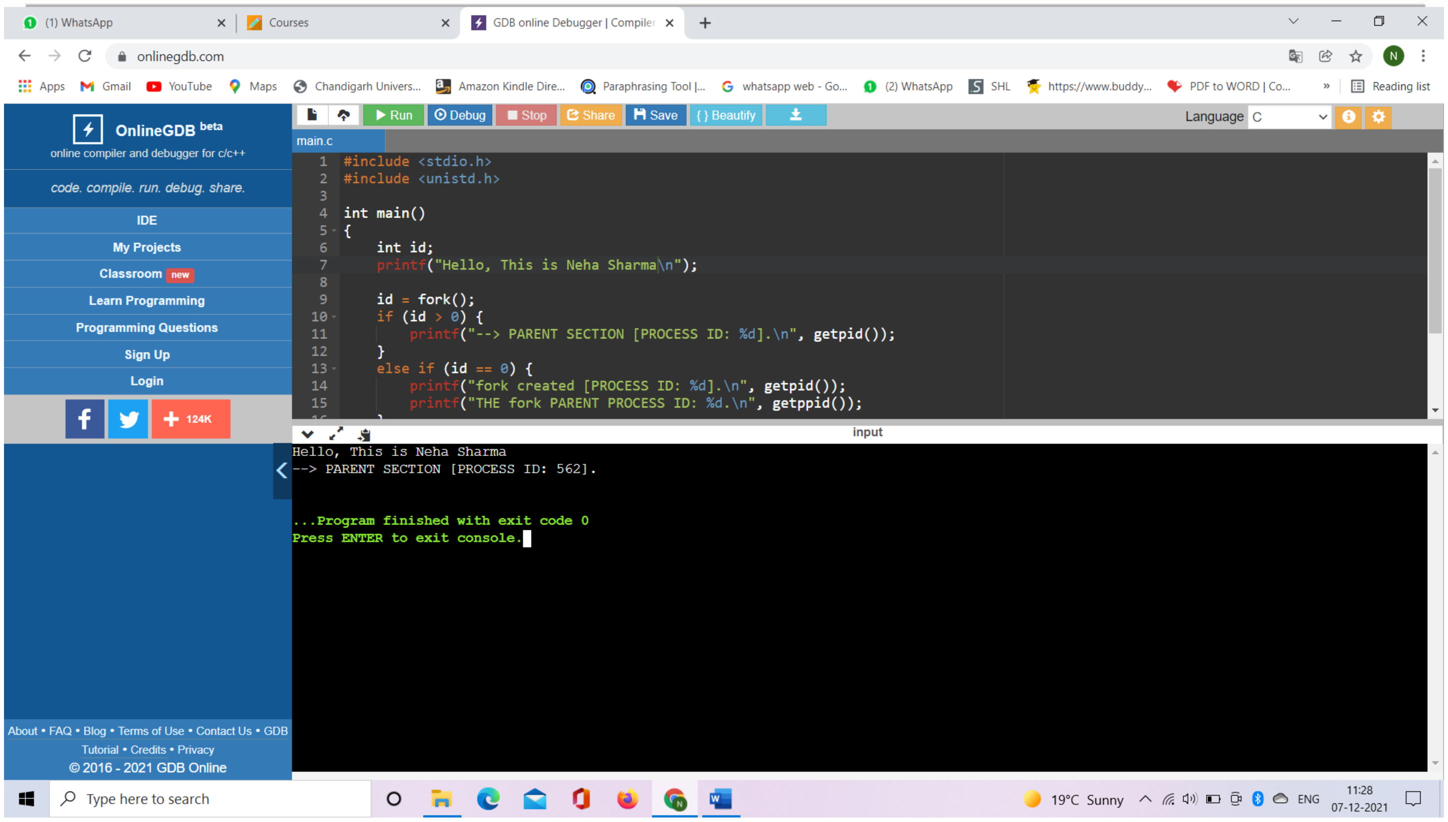
3. Apparatus:- PC , online Compiler

4. Program Code-

```
#include <stdio.h>

#include <unistd.h>


int main()

{

   int id;

   printf("Hello, This is Neha  Sharma\n");


   id = fork();

   if (id > 0) {

      printf("--> PARENT SECTION [PROCESS ID: %d].\n", getpid());

   }

   else if (id == 0) {

      printf("fork created [PROCESS ID: %d].\n", getpid());

      printf("THE fork PARENT PROCESS ID: %d.\n", getppid());

   }

   else {

      printf("ERROR fork CREATION FAILED\n");
```

```
    }

    return 0;
}
```

5. OUTPUT



```c
#include <stdio.h>
#include <unistd.h>

int main()
{
    int id;
    printf("Hello, This is Neha Sharma\n");

    id = fork();
    if (id > 0) {
        printf("--> PARENT SECTION [PROCESS ID: %d].\n", getpid());
    }
    else if (id == 0) {
        printf("fork created [PROCESS ID: %d].\n", getpid());
        printf("THE fork PARENT PROCESS ID: %d.\n", getppid());
```

```
Hello, This is Neha Sharma
--> PARENT SECTION [PROCESS ID: 562].


...Program finished with exit code 0
Press ENTER to exit console.
```

B. Write a program to implement the SJF with 5 processes out of which 2 process having same burst time.

1. Aim/Overview of the practical:- Simulation of shortest job first scheduling algorithm

2. Task to be done:- We have to implement the program shortest job first scheduling algorithm.

3. Algorithm/Flowchart (For programming based labs)**:**

1. Sort all the process according to the arrival time.
2. Then select that process which has minimum arrival time and minimum Burst time.
3. After completion of process make a pool of process which after till the completion of previous pro-cess and select that process among the pool which is having minimum Burst time.

4. Steps for experiment/practical:

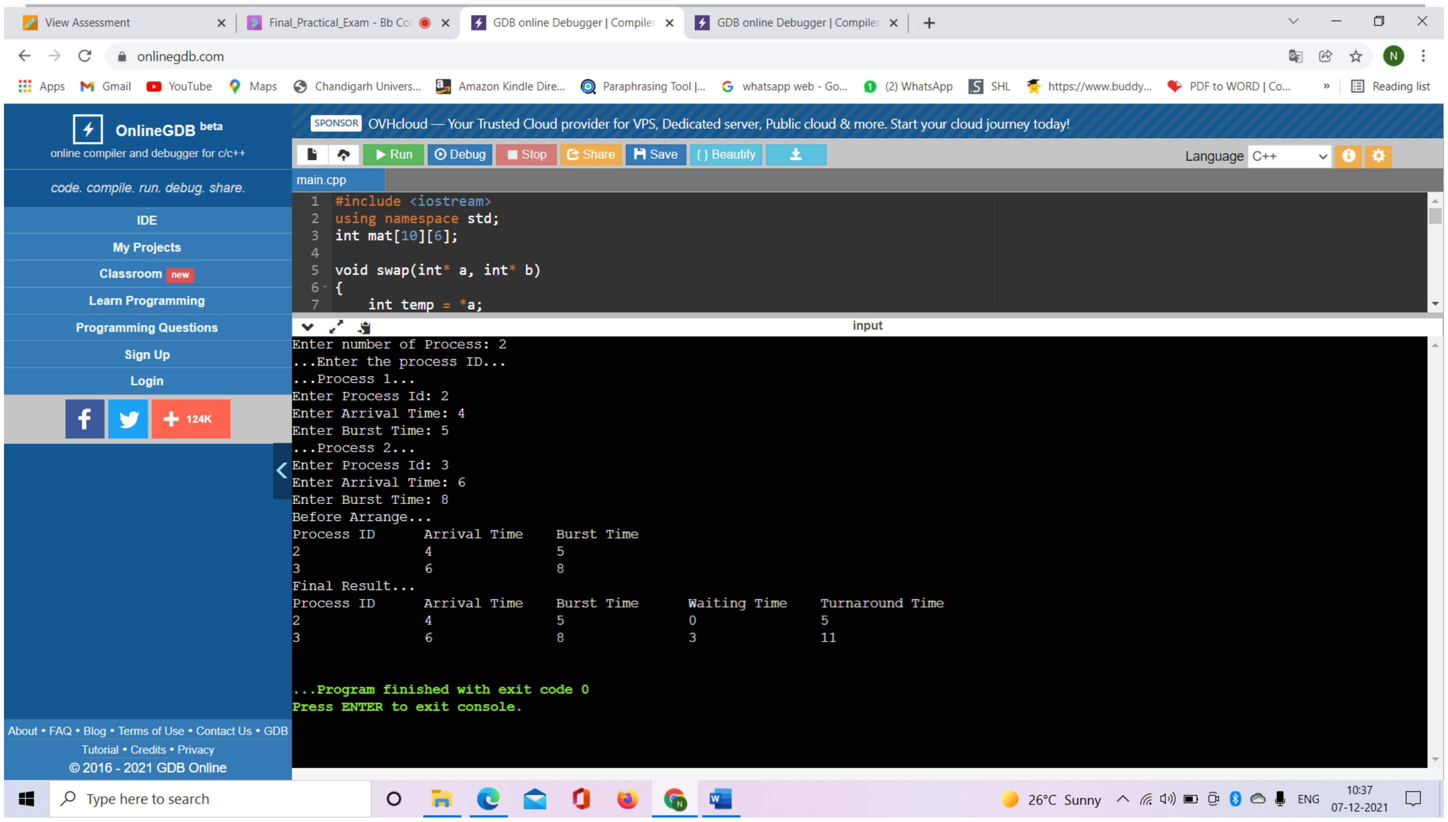
```
#include <iostream>
using namespace std;
int mat[10][6];

void swap(int* a, int* b)
{
	int temp = *a;
	*a = *b;
	*b = temp;
}

void arrangeArrival(int num, int mat[][6])
{
	for (int i = 0; i < num; i++) {
		for (int j = 0; j < num - i - 1; j++) {
			if (mat[j][1] > mat[j + 1][1]) {
				for (int k = 0; k < 5; k++) {
					swap(mat[j][k], mat[j + 1][k]);
				}
			}
		}
	}
}
```

```
void completionTime(int num, int mat[][6])
{
        int temp, val;
        mat[0][3] = mat[0][1] + mat[0][2];
        mat[0][5] = mat[0][3] - mat[0][1];
        mat[0][4] = mat[0][5] - mat[0][2];

        for (int i = 1; i < num; i++) {
                temp = mat[i - 1][3];
                int low = mat[i][2];
                for (int j = i; j < num; j++) {
                        if (temp >= mat[j][1] && low >= mat[j][2]) {
                                low = mat[j][2];
                                val = j;
                        }
                }
                mat[val][3] = temp + mat[val][2];
                mat[val][5] = mat[val][3] - mat[val][1];
                mat[val][4] = mat[val][5] - mat[val][2];
                for (int k = 0; k < 6; k++) {
                        swap(mat[val][k], mat[i][k]);
                }
        }
}

int main()
{
        int num, temp;

        cout << "Enter number of Process: ";
        cin >> num;
        cout << "...Enter the process ID...\n";
        for (int i = 0; i < num; i++) {
                cout << "...Process " << i + 1 << "...\n";
                cout << "Enter Process Id: ";
                cin >> mat[i][0];
                cout << "Enter Arrival Time: ";
                cin >> mat[i][1];
                cout << "Enter Burst Time: ";
                cin >> mat[i][2];
        }

        cout << "Before Arrange...\n";
        cout << "Process ID\tArrival Time\tBurst Time\n";
        for (int i = 0; i < num; i++) {
                cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"
                        << mat[i][2] << "\n";
        }

        arrangeArrival(num, mat);
        completionTime(num, mat);
        cout << "Final Result...\n";
        cout << "Process ID\tArrival Time\tBurst Time\tWaiting "
                        "Time\tTurnaround Time\n";
        for (int i = 0; i < num; i++) {
                cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"
                        << mat[i][2] << "\t\t" << mat[i][4] << "\t\t"
                        << mat[i][5] << "\n";
        }
}
```

5. Observations/ Discussions (For applied/experimental sciences/materials based labs):

1. Find **waiting time** for all other processes i.e. for Process i -> wt[i] = bt[i-1] + wt[i-1] .
2. Find **turnaround time** = waiting time + burst time for all processes.
3. Find **average waiting time** = total waiting time / no of processes.
4. Similarly, find **average turnaround time** = total turnaround time / no of processes.

6. Output



```
#include <iostream>
using namespace std;
int mat[10][6];

void swap(int* a, int* b)
{
    int temp = *a;
```

```
Enter number of Process: 2
...Enter the process ID...
...Process 1...
Enter Process Id: 2
Enter Arrival Time: 4
Enter Burst Time: 5
...Process 2...
Enter Process Id: 3
Enter Arrival Time: 6
Enter Burst Time: 8
Before Arrange...
Process ID      Arrival Time    Burst Time
2               4               5
3               6               8
Final Result...
Process ID      Arrival Time    Burst Time      Waiting Time    Turnaround Time
2               4               5               0               5
3               6               8               3               11


...Program finished with exit code 0
Press ENTER to exit console.
```

Learning outcomes (What I have learnt):

1. Learned about shortest job first algorithm.

2. Learned about how to calculate Waiting time, turnaround time, average waiting time, and average turnaround time.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks

1.			
2.			
3.			