Experiment 9

Student Name: V.Sri Surya Prakash Reddy **UID: 21BCS9133**
Branch: CSE-AML **Section/Group: 21AIML-12(A)**
Semester: 3rd **Date of Performance: 11-10-2022**
Subject Name: Operating System **Subject Code:21CSH-242**

1. Aim/Overview of the practical:

Simulation of Shortest Job First CPU Scheduling Algorithm

2. Task to be done:

Write Program to calculate Average turnaround time and Average waiting time using SJF Scheduling Algorithm

Shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next. SJN is a non-preemptive algorithm.

• Shortest Job first has the advantage of having a minimum average waiting time among all scheduling algorithms.

• It is a Greedy Algorithm.

• It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.

• It is practically infeasible as Operating System may not know burst time and therefore may not sort them. While it is not possible to predict execution time, several methods can be used to estimate the execution time for a job, such as a weighted average of previous execution times. SJF can be used in specialized environments where accurate estimates of running time are available.

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

Step 1 : Sort all the process according to the arrival time.

Step 2 : Then select that process which has minimum arrival time and minimum Burst time.

}

Step 3 : After completion of process make a pool of process which after till the completion of

previous process and select that process among the pool which is having minimum Burst time.

4. Steps for practical/code:

```
#include <iostream>

using namespace std;

int mat[10][6];

void swap(int* a, int* b)

{

int temp = *a;

*a = *b;

*b = temp;

}

void sort_it(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 CT(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";

}

sort_it(num, mat);

CT(num, mat);
```

cout << "Final Result...\n";

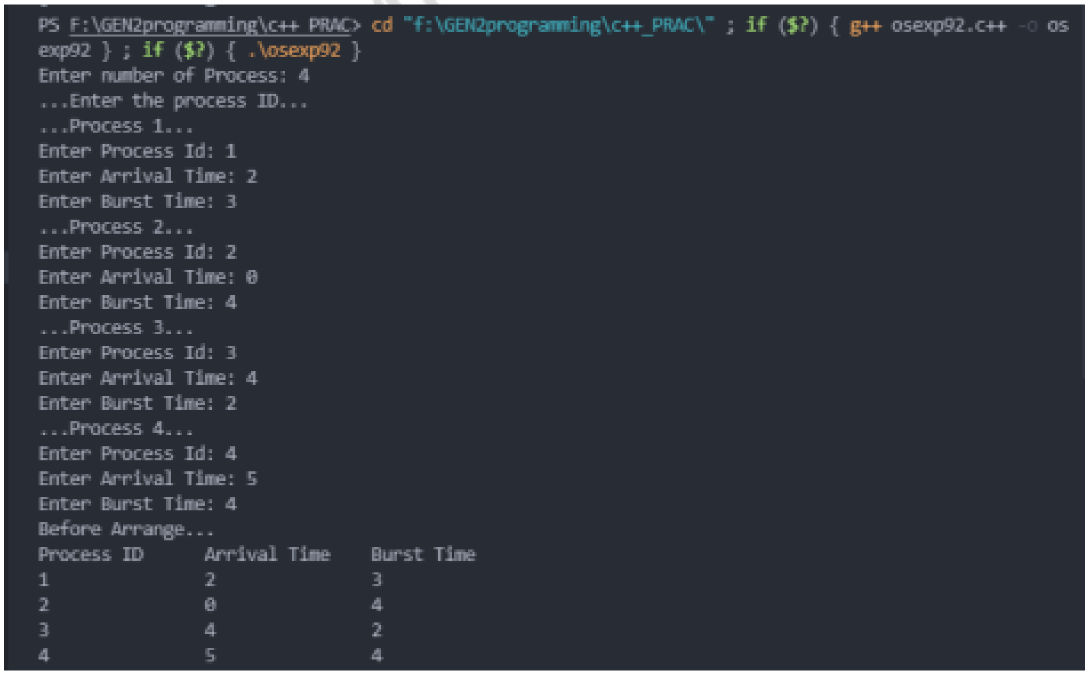
cout << "Process ID\tArrival Time\tBurst Time\tWaiting ""Time\tTurnaround Time\n";

for (int i = 0; i < num; i++) {

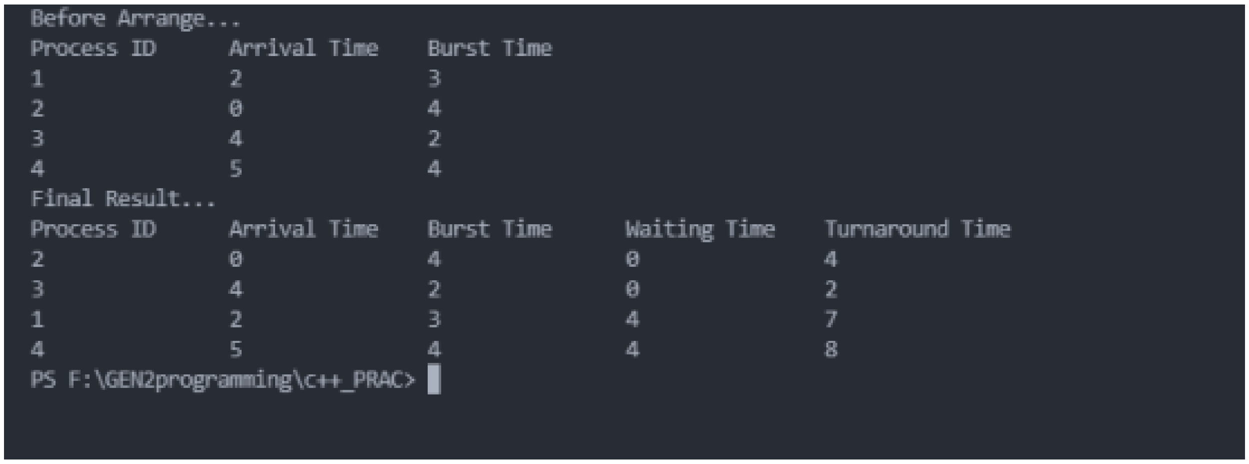
cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"<< mat[i][2] << "\t\t" << mat[i][4] << "\t\t"<< mat[i][5] <<\n";

}

5. Result/Output:



```
PS F:\GEN2programming\c++ PRAC> cd "f:\GEN2programming\c++_PRAC\" ; if ($?) { g++ osexp92.c++ -o os
exp92 } ; if ($?) { .\osexp92 }
Enter number of Process: 4
...Enter the process ID...
...Process 1...
Enter Process Id: 1
Enter Arrival Time: 2
Enter Burst Time: 3
...Process 2...
Enter Process Id: 2
Enter Arrival Time: 0
Enter Burst Time: 4
...Process 3...
Enter Process Id: 3
Enter Arrival Time: 4
Enter Burst Time: 2
...Process 4...
Enter Process Id: 4
Enter Arrival Time: 5
Enter Burst Time: 4
Before Arrange...
Process ID      Arrival Time    Burst Time
1               2               3
2               0               4
3               4               2
4               5               4
```



```
Before Arrange...
Process ID      Arrival Time    Burst Time
1               2               3
2               0               4
3               4               2
4               5               4
Final Result...
Process ID      Arrival Time    Burst Time      Waiting Time    Turnaround Time
2               0               4               0               4
3               4               2               0               2
1               2               3               4               7
4               5               4               4               8
PS F:\GEN2programming\c++_PRAC>
```

6. Learning outcomes (What I have learnt):

1. learned about the concept of scheduling in OS

2. learned about different scheduling algorithms

3. learned and understood the working of SJF scheduling

4. implemented SJF scheduling using c++ program

5. learned about WT , TAT , CT.

Evaluation Grid :

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Student Performance (Conduct of experiment) objectives/Outcomes.		12
2.	Viva Voce		10
3.	Submission of Work Sheet (Record)		8
	Total		30