Experiment 03

To write a program for the implementation of the SJF scheduling algorithm

<u>Learning Objective:</u> Students should be able to understand the SJF algorithm by using different coding language C/C++/Java/Python.

Tools: Online compiler

Theory:

Shortest Job First (SJF) is a CPU scheduling algorithm that selects the process with the shortest execution time to run next. It can be pre-emptive (Shortest Remaining Time First) or non-pre-emptive.

Working Principle:

- 1. The scheduler selects the process with the smallest burst time (execution time).
- 2. If two processes have the same burst time, selection is made based on arrival time.
- 3. The process runs until completion (non-pre-emptive) or until a new shorter job arrives (pre-emptive).
- 4. This continues until all processes are executed.

Types of SJF:

- Non-Pre-emptive SJF: Once a process starts, it runs until completion.
- Pre-emptive SJF (SRTF): A new process with a shorter burst time can interrupt an ongoing process.

Advantages:

- Minimizes average waiting time.
- Reduces average turnaround time.
- More efficient for batch processing.

Disadvantages:

- Can lead to starvation of longer processes.
- Difficult to implement in real-time as future burst times are not always known.

Class: SE CSE Name: Rishabh Hadagali Roll No: 11

TCET



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (CYBER SECURITY

Y) tcet

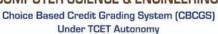
Choice Based Credit Grading System (CBCGS)
Under TCET Autonomy

Code:

```
print("\nSHORTEST JOB FIRST SCHEDULLING\n")
n = int(input('Enter no of processes: '))
bt = [0] * (n + 1)
at = [0] * (n + 1)
abt = [0] * (n + 1)
for i in range(n):
    abt[i] = int(input('\nEnter the burst time for process {} : '.format(i + 1)))
   at[i] = int(input('Enter the arrival time for process {} : '.format(i + 1)))
    bt[i] = [abt[i], at[i], i]
bt.pop(-1)
sumbt = 0
i = 0
11 = []
for i in range(0, sum(abt)):
    l = [j \text{ for } j \text{ in bt } if j[1] <= i]
    1.sort(key=1ambda x: x[0])
    bt[bt.index(1[0])][0] -= 1
    for k in bt:
        if k[0] == 0:
            t = bt.pop(bt.index(k))
            ll.append([k, i + 1])
ct = [0] * (n + 1)
tat = [0] * (n + 1)
wt = [0] * (n + 1)
for i in ll:
    ct[i[0][2]] = i[1]
for i in range(len(ct)):
    tat[i] = ct[i] - at[i]
    wt[i] = tat[i] - abt[i]
ct.pop(-1)
wt.pop(-1)
tat.pop(-1)
abt.pop(-1)
at.pop(-1)
print(f'\nBT\tAT\tCT\tTAT\tWT')
for i in range(len(ct)):
    print("{}\t{}\t{}\t{}\n".format(abt[i], at[i], ct[i], tat[i], wt[i]))
print(f'\nAverage Waiting Time = ', sum(wt)/len(wt))
print('Average Turnaround Time = ', sum(tat)/len(tat), f'\n')
```



TCET DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (CYBER SECURITY)





Output Screenshot:

SHORTEST JOB FIRST SCHEDULLING
Enter no of processes: 5
Enter the burst time for process 1 : 6 Enter the arrival time for process 1 : 2
Enter the burst time for process 2 : 2 Enter the arrival time for process 2 : 5
Enter the burst time for process 3 : 8 Enter the arrival time for process 3 : 1
Enter the burst time for process 4 : 3 Enter the arrival time for process 4 : 0
Enter the burst time for process 5 : 4 Enter the arrival time for process 5 : 4

BT 6	AT 2	CT 15	TAT 13	WT 7
2	5	7	2	0
8	1	23	22	14
3	0	3	3	0
4	4	10	6	2
		; Time = ound Time		

Learning Outcomes:

The student should have the ability to:

LO 2.1 Outline various compilers for different language

LO 2.2 Understood the SJF algorithm

LO 2.3 Choose an appropriate compiler to solve the algorithm.

<u>Course Outcomes:</u> Upon completion of the course students will be able to learn about operating systems and security concepts.

Conclusion:

Class: SE CSE Name: Rishabh Hadagali Roll No: 11

Viva Questions:

1. Find the average waiting time and average waiting time for SJF algorithm.

Process	Burst Time	Arrival Time
P1	6 ms	2 ms
P2	2 ms	5 ms
P3	8 ms	1 ms
P4	3 ms	0 ms
P5	4 ms	4 ms

For Faculty Use:

Correction	Formative	Timely	Attendance/
Parameters	Assessment	completion	Learning
	[40%]	of Practical	Attitude
		[40%]	[20%]
Marks			
Obtained			

Class: SE CSE Name: Rishabh Hadagali Roll No: 11