



**Department of Computer Science Fall 2023**

**CSC 203 – Operating Systems**

**Lab #8**

**Objective:**

**CPU Scheduling – FCFS and SJF**

Implementation of CPU scheduling algorithms; FCFS and SJF.

| Name of Student | maha rehan |
| --- | --- |
| Student ID | 22sp-018-se |
| Date of Lab Conducted |  |

**Objective:** Implementation of CPU scheduling algorithms; FCFS and SJF.

**THEORY**

**FCFS CPU SCHEDULING ALGORITHM**

**First Come First Serve** is a Non-Preemptive Scheduling algorithm where each process is executed according to its arrival time.

**Step 1**: Input the number of processes required to be scheduled using FCFS, burst time for each process and its arrival time.

**Step 2**: Using enhanced bubble sort technique, sort the all given processes in ascending order according to arrival time in a ready queue.

**Step 3**: Calculate the Finish Time, Turn Around Time and Waiting Time for each process which in turn help to calculate Average Waiting Time and Average Turn Around Time required by CPU to schedule given set of process using FCFS.

**Step 4**: Process with less arrival time comes first and gets scheduled first by the CPU.

**Step 5**: Calculate the Average Waiting Time and Average Turn Around Time.

**Step 6**: Stop.

**Sample Run:**

Enter total number of processes (maximum 20): 3

Enter Process Arrival Time

P[1]:0

P[2]:0

P[3]:0

Enter Process Burst Time

P[1]:24

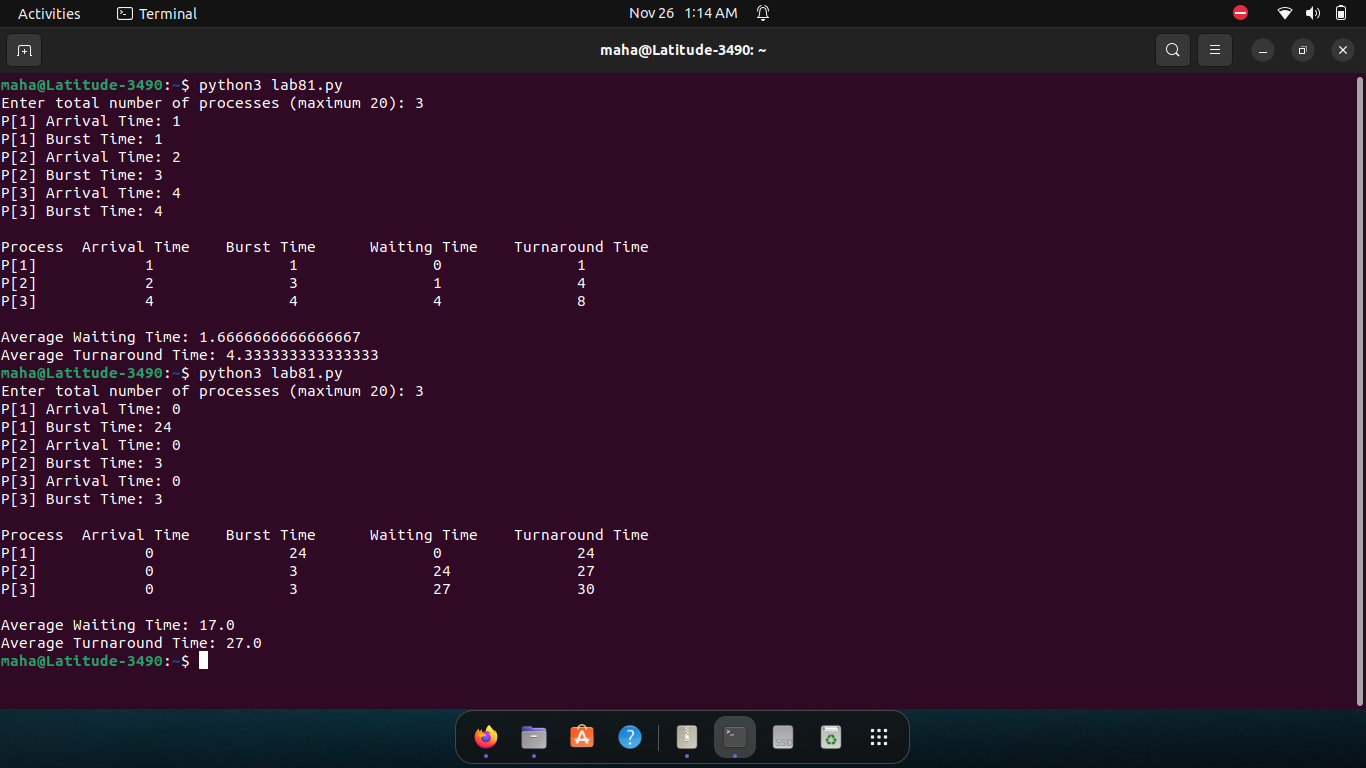
P[2]:3

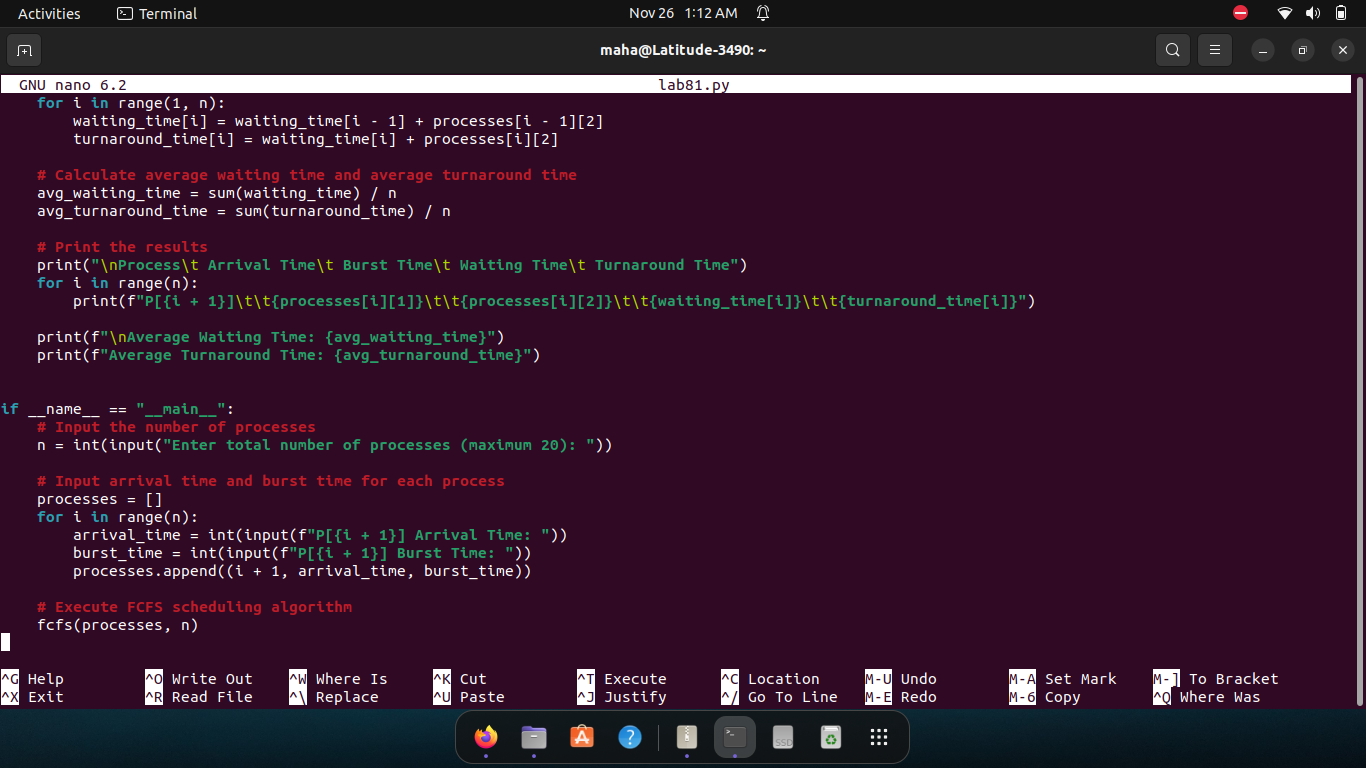
P[3]:3

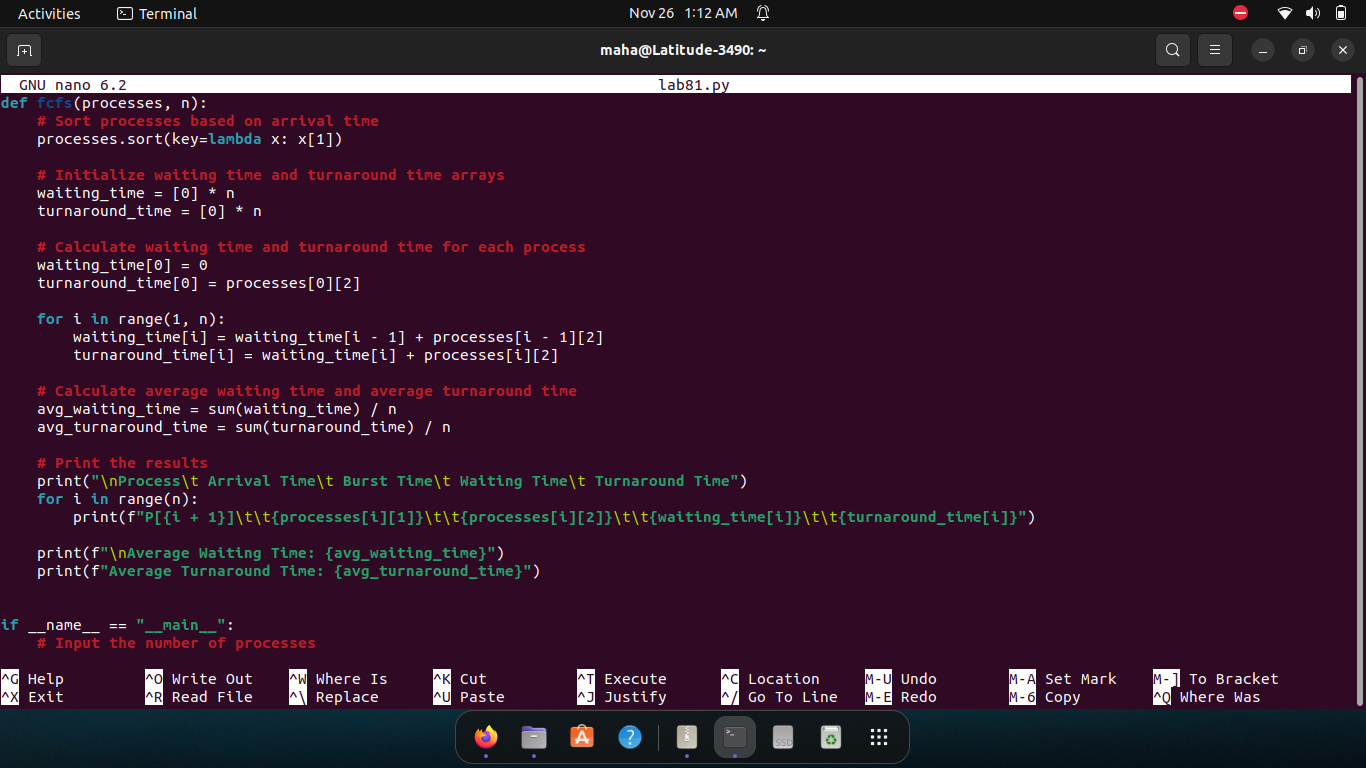
| **Process ID** | **Arrival Time** | **Burst Time** | **Waiting Time** | **Turnaround Time** |
| --- | --- | --- | --- | --- |
| P[1] | 0 | 24 | 0 | 24 |
| P[2] | 0 | 3 | 24 | 27 |
| P[3] | 0 | 3 | 27 | 30 |

Average Waiting Time: 17

Average Turnaround Time: 27







**SJF CPU 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.

**Step 1**: Input the number of processes required to be scheduled using SJF, burst time for each process and its arrival time.

**Step 2**: Using selection sort technique, sort the all given processes in ascending order according to burst time in ascending order.

**Step 3**: Calculate the Finish Time, Turn Around Time and Waiting Time for each process which in turn help to calculate Average Waiting Time and Average Turn Around Time required by CPU to schedule given set of process

**Step 4**: Process with less Burst and arrival time comes first and gets scheduled first by the CPU.

**Step 5**: Calculate the Average Waiting Time and Average Turn Around Time.

**Step 6**: Stop

**Sample Run:**

Enter number of process: 4

Enter Burst Time:

P[1]:4

P[2]:8

P[3]:3

P[4]:7

Enter Process Arrival Time

P[1]:0

P[2]:0

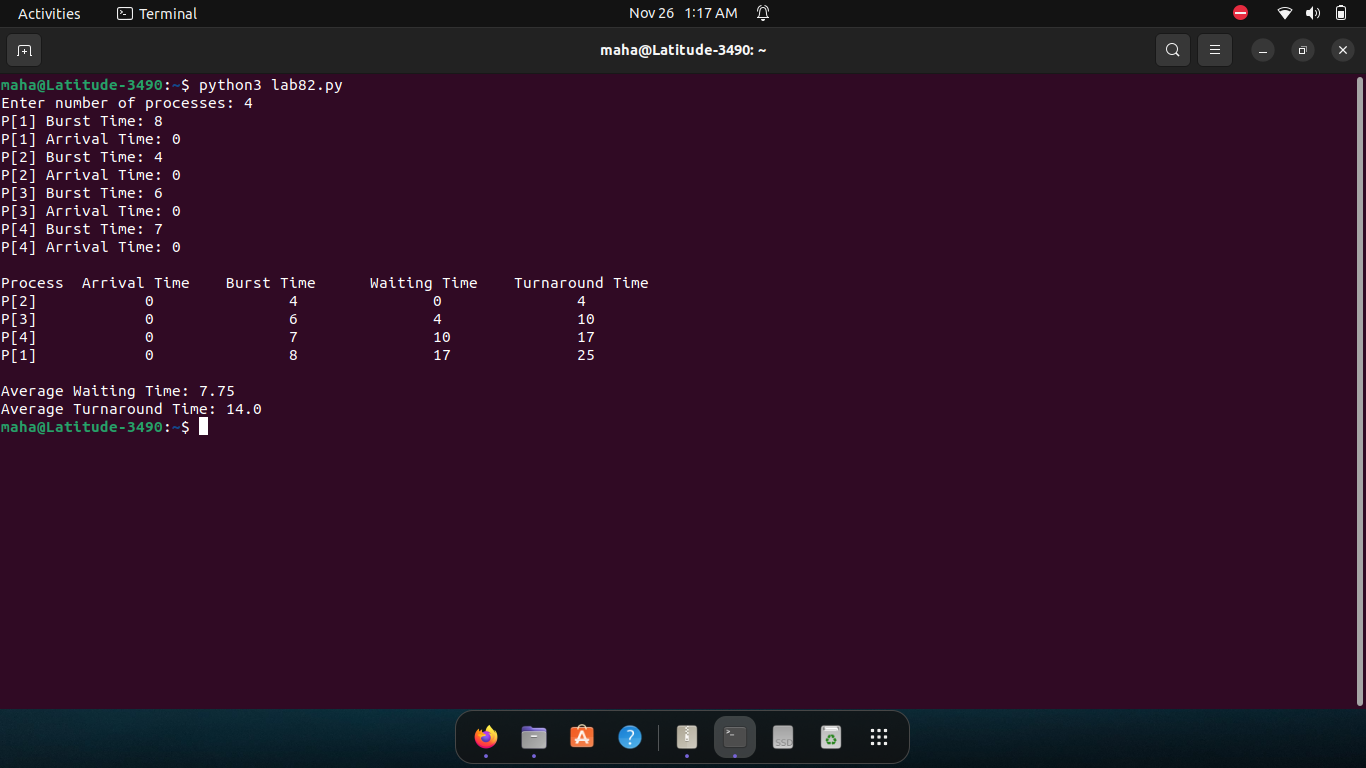
P[3]:0

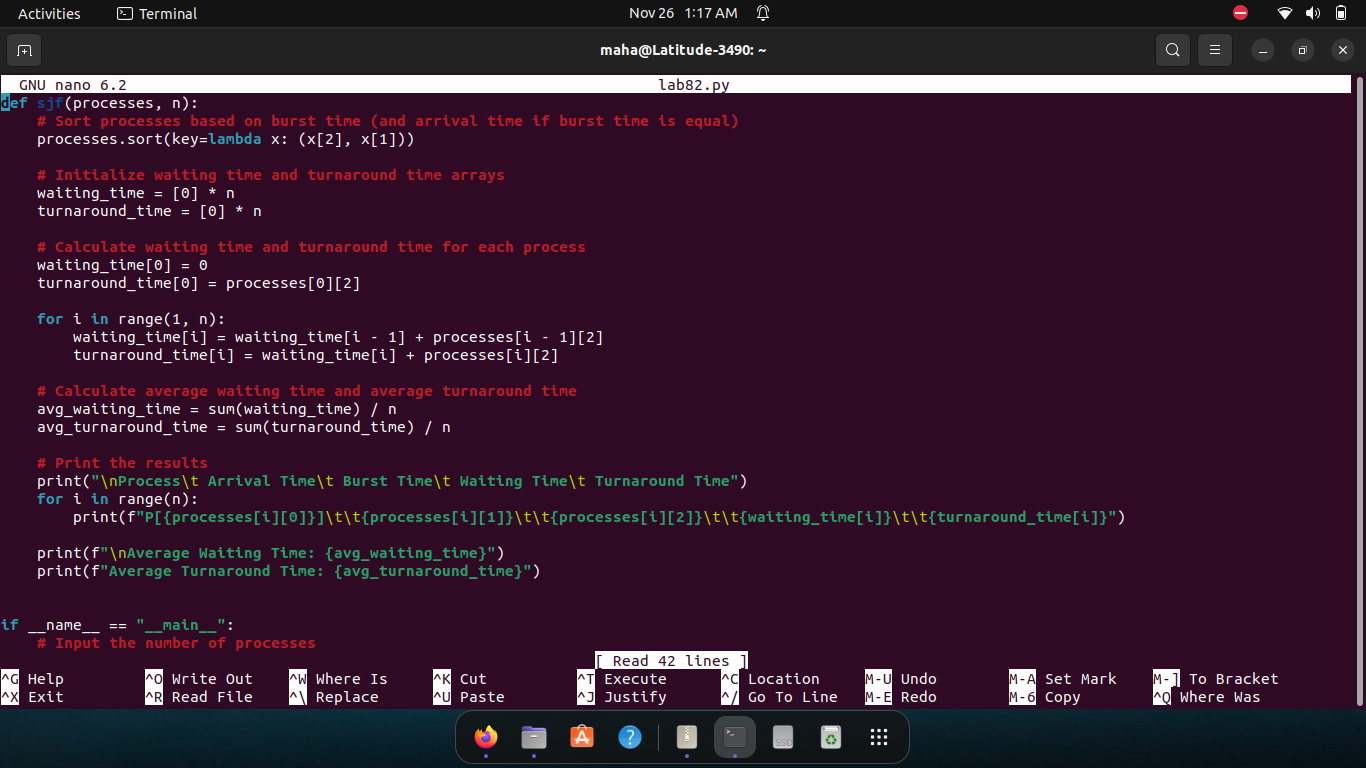
P[4]:0

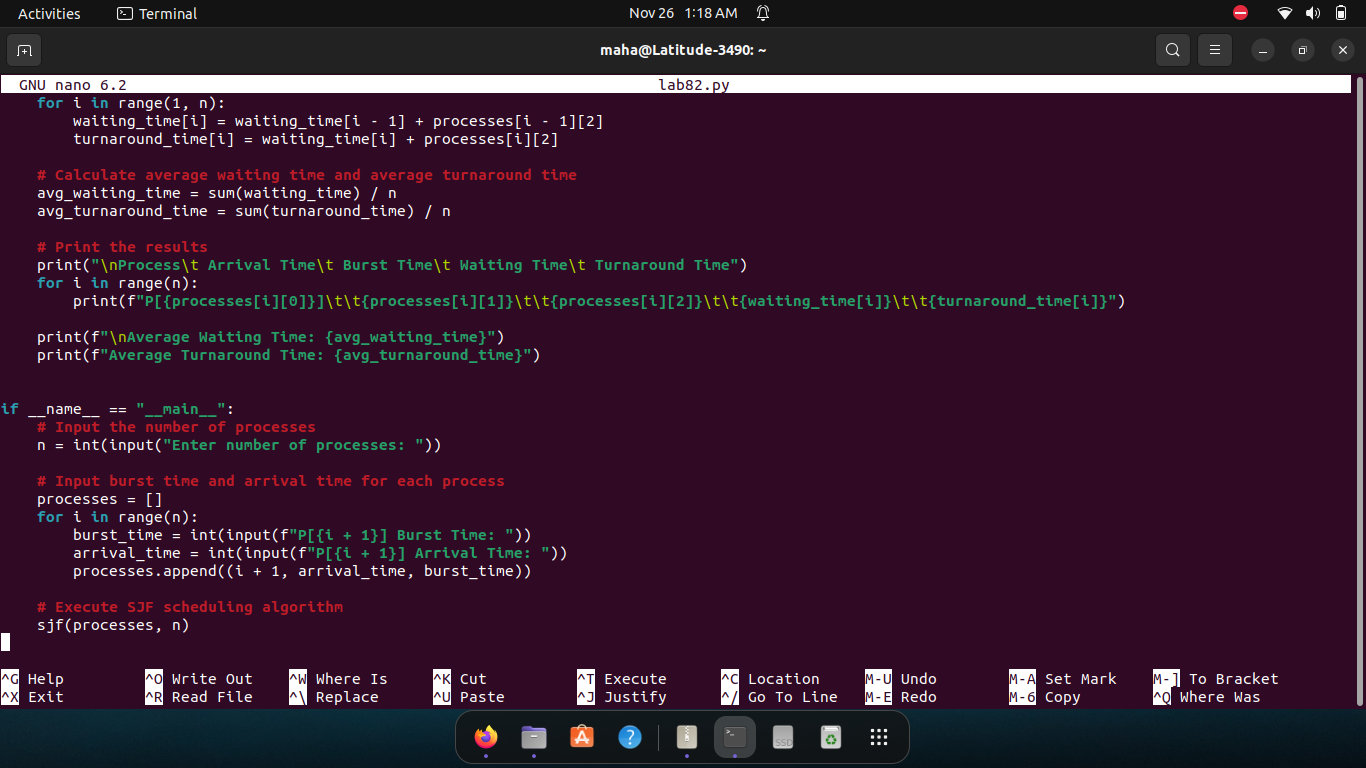
| **Process ID** | **Arrival Time** | **Burst Time** | **Waiting Time** | **Turnaround Time** |
| --- | --- | --- | --- | --- |
| P[3] | 0 | 3 | 0 | 3 |
| P[1] | 0 | 4 | 3 | 7 |
| P[4] | 0 | 7 | 7 | 14 |
| P[2] | 0 | 8 | 14 | 22 |

Average Waiting Time = 6.000000

Average Turnaround Time = 11.500000







**Terms and formulas used in above scheduling algorithms:**

Completion Time (CT): Time at which process completes its execution.

Turn Around Time (TAT): Time Difference between completion time and arrival time.

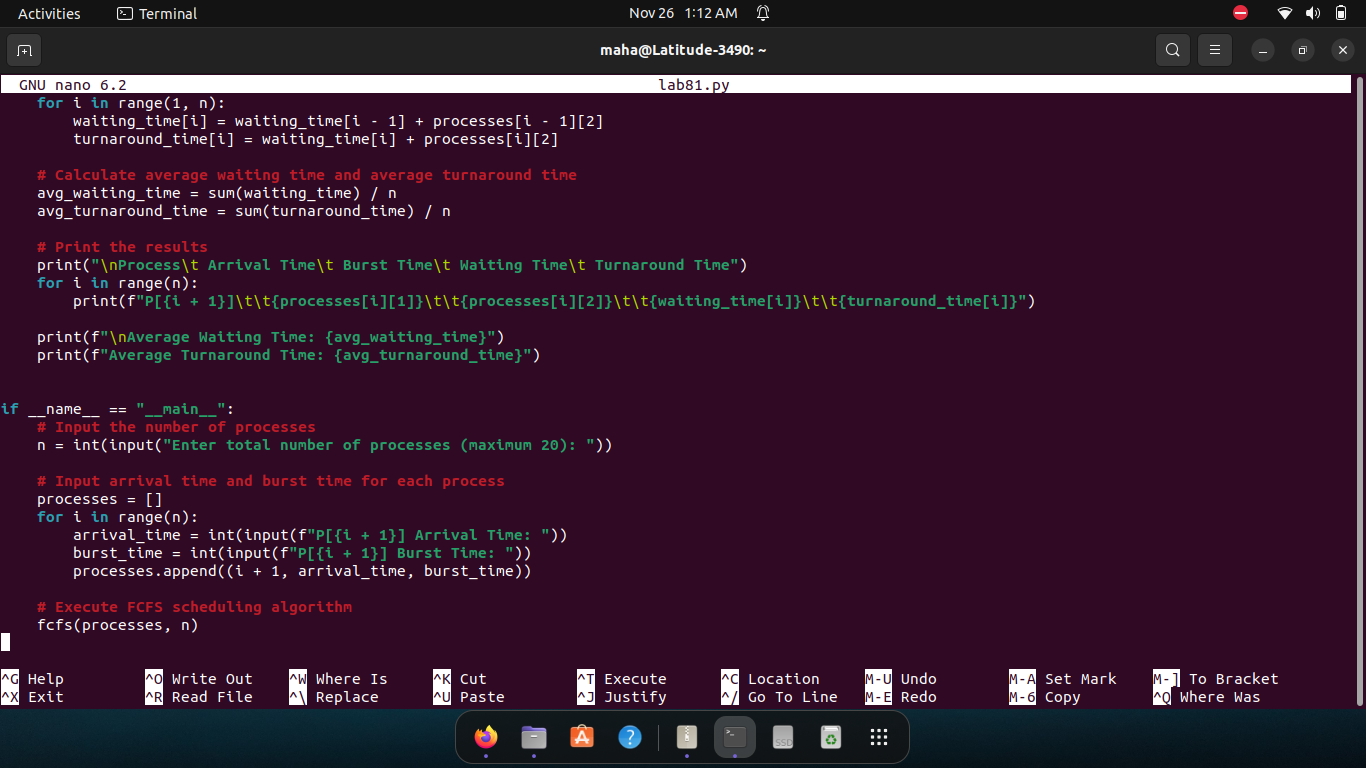
**Turn Around Time** = Completion Time – Arrival Time

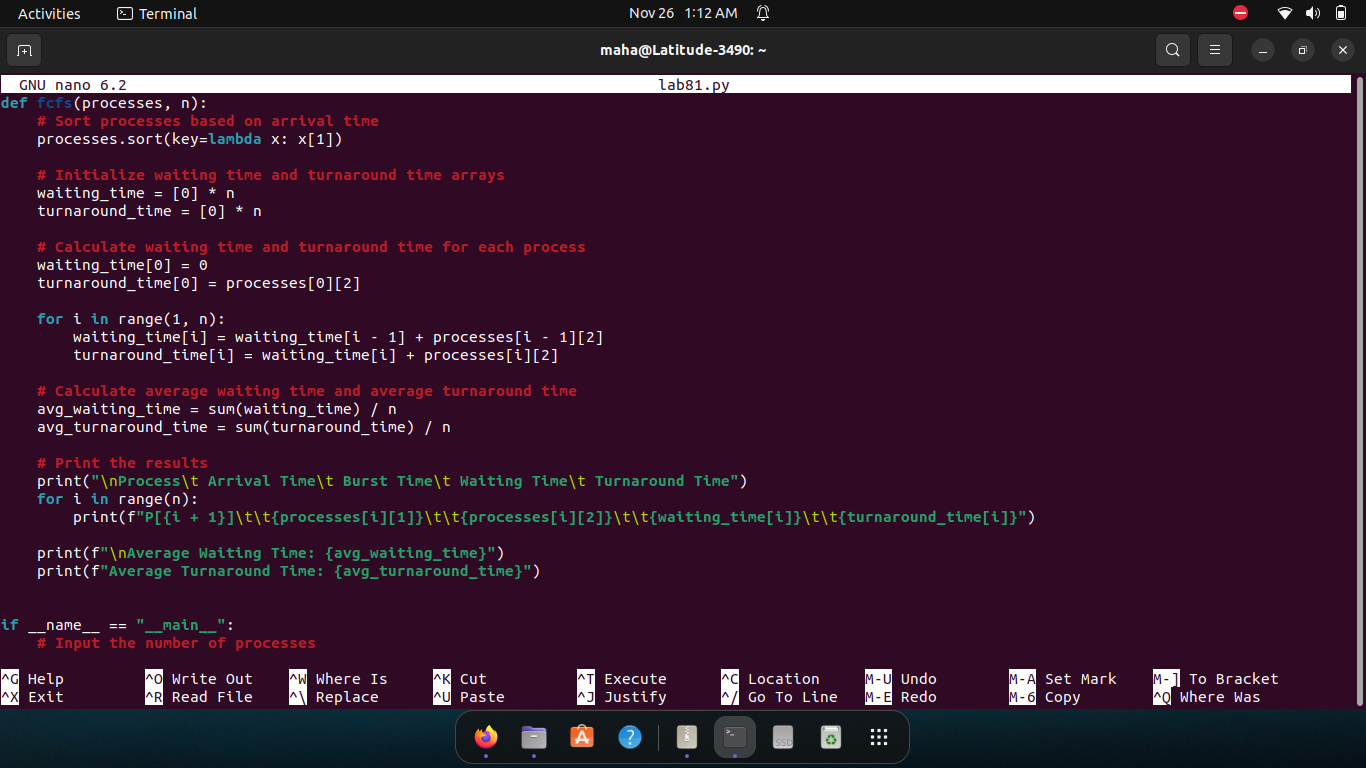
Waiting Time (WT): Time Difference between turnaround time and burst time.

**Waiting Time** = Turn Around Time – Burst Time

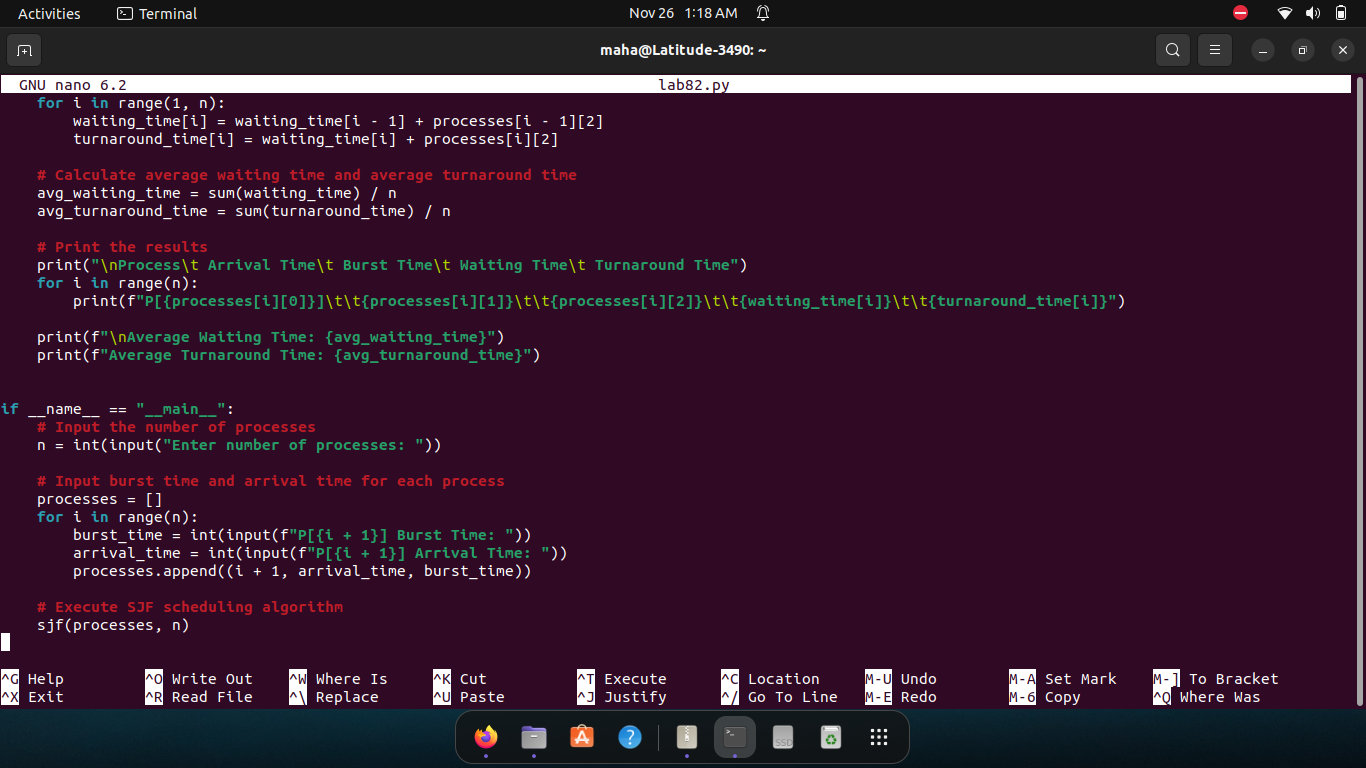
**Lab Exercise(s):**

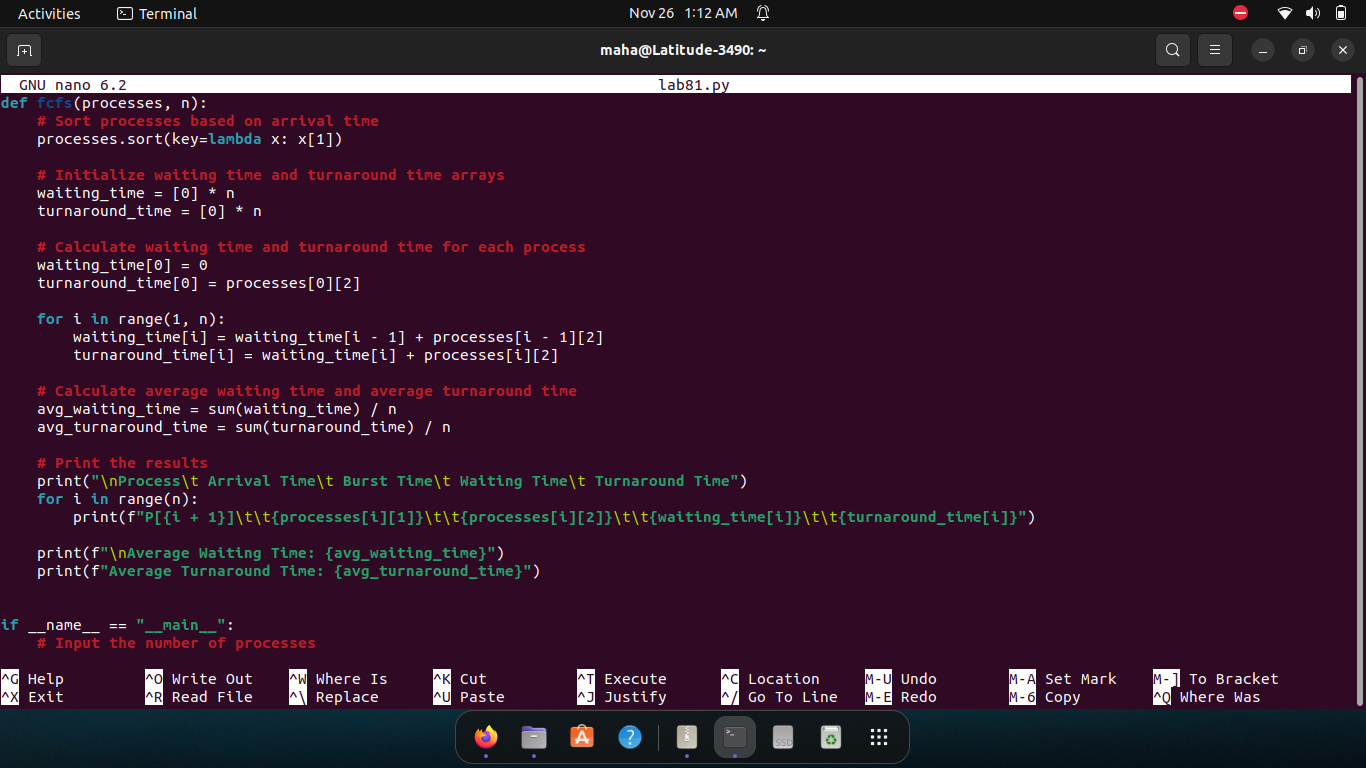
1. Write a Python program to implement and simulate the FCFS Algorithm.



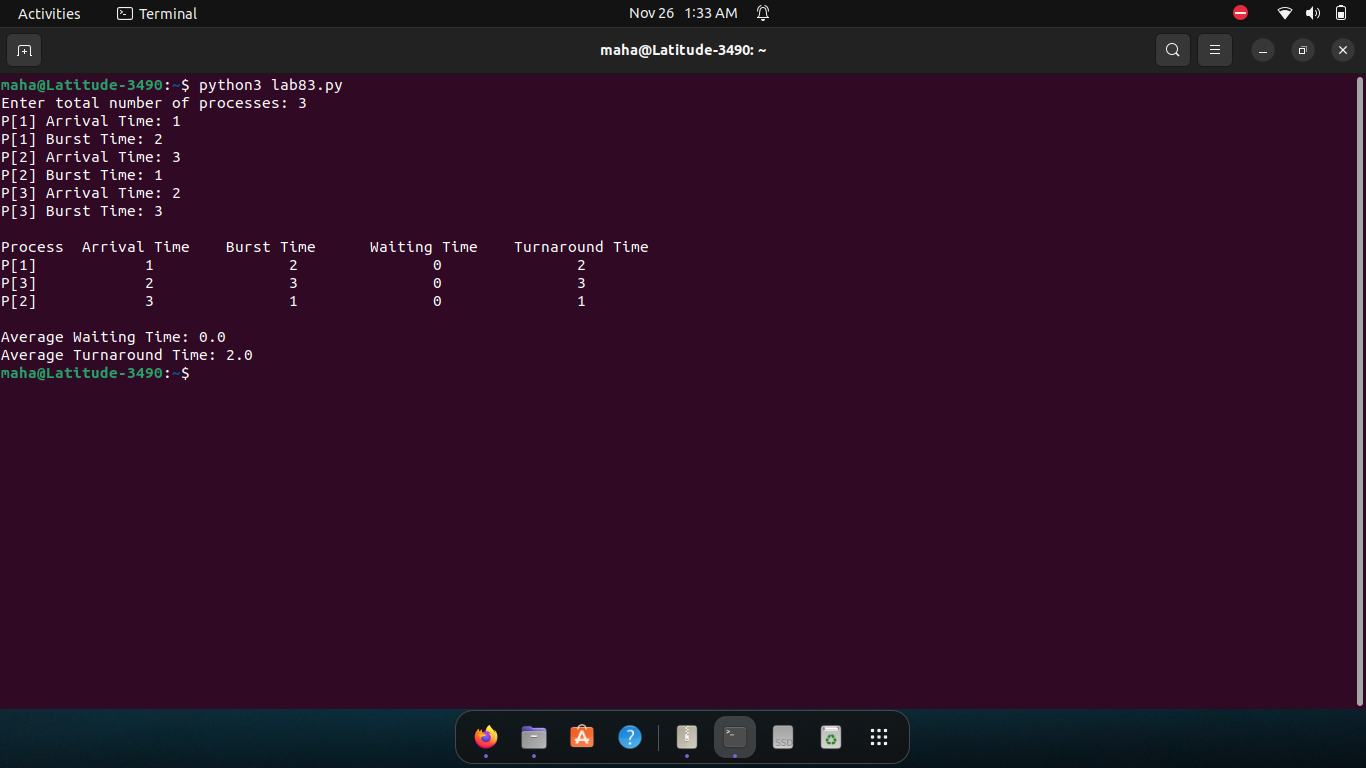


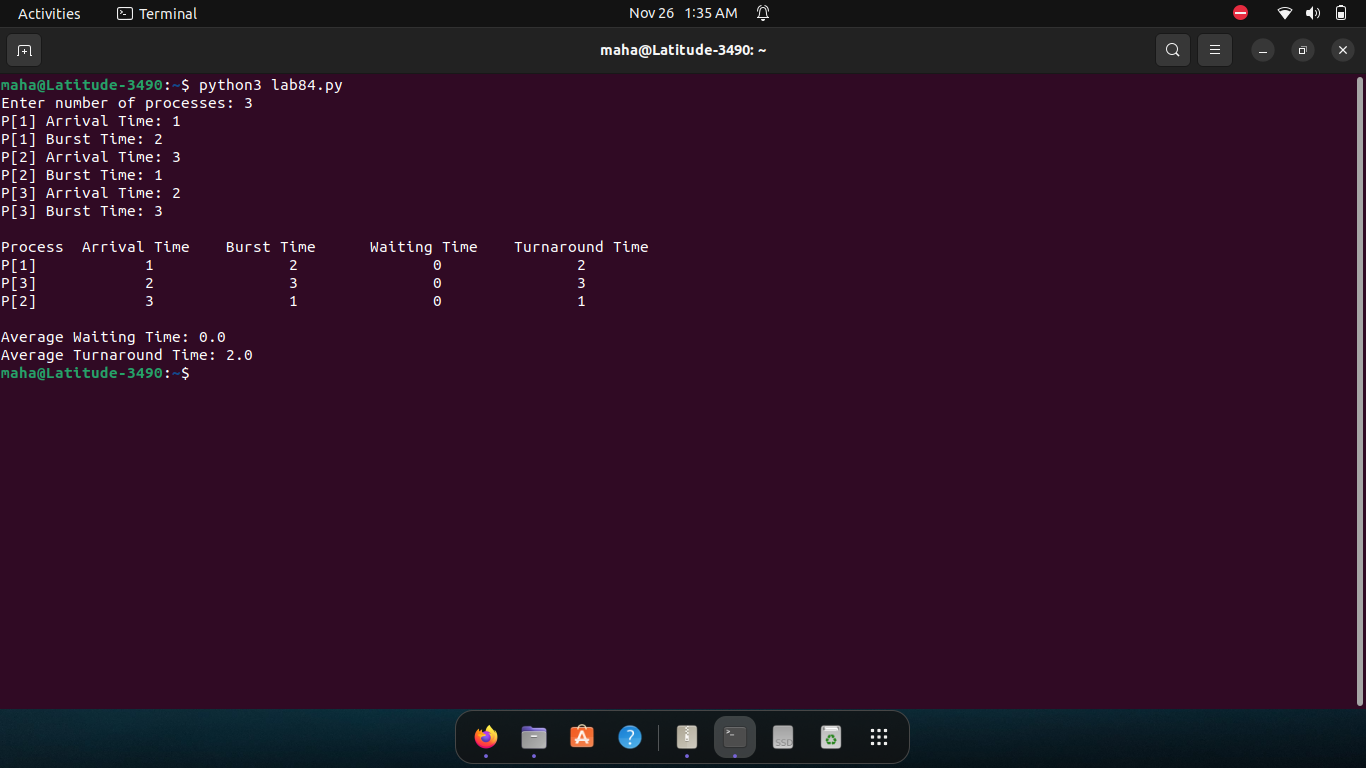
1. Write a Python program to implement and simulate the SJF Algorithm.



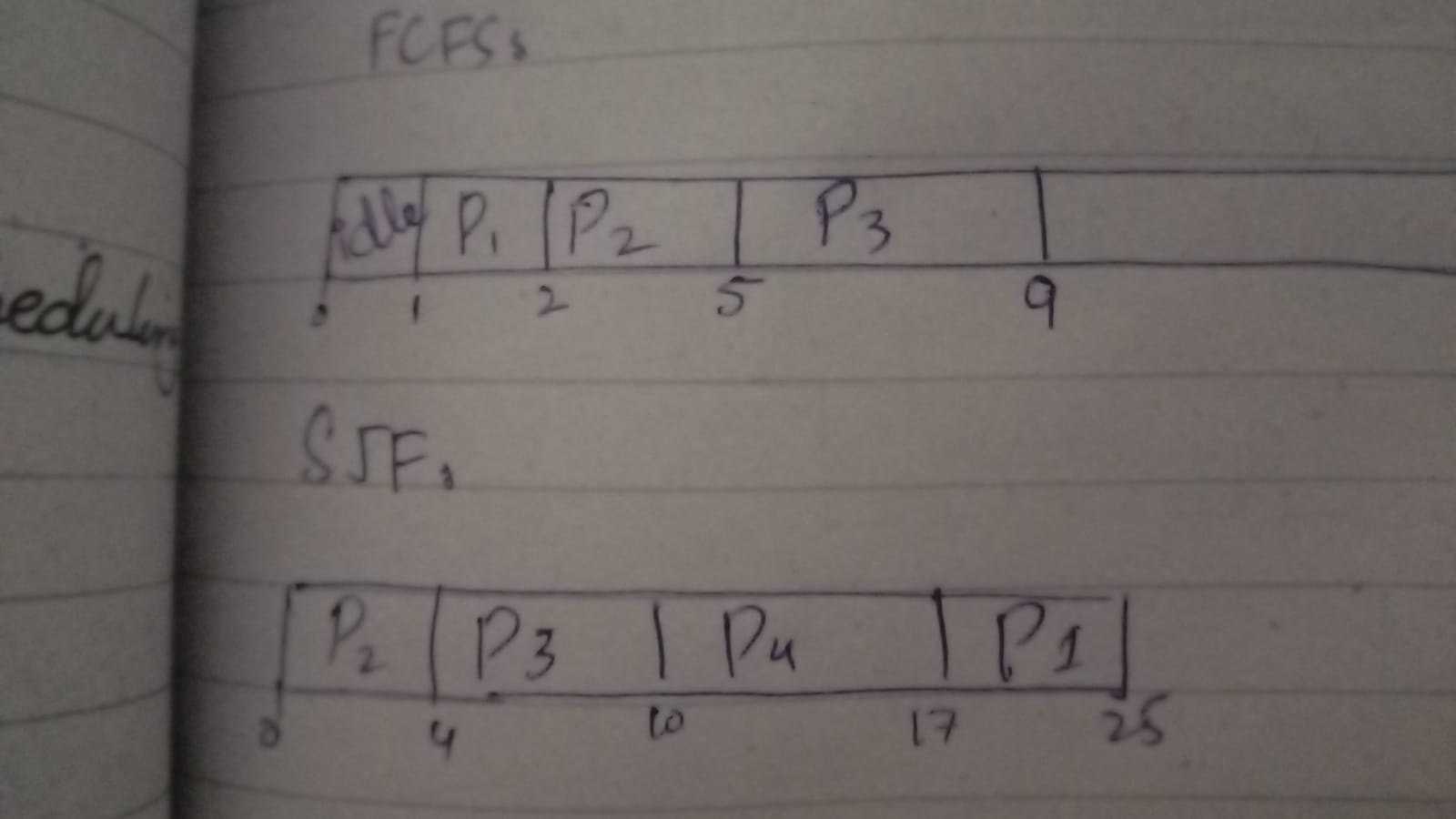


1. Modify both algorithms for the different arrival time.





1. Draw Gantt chart of the result for both algorithms.



1. Show results for both algorithms in the table form with average waiting time and average turnaround time.

