**Department of Computer Science**



**Fall 2023**

**CSC 203 – Operating Systems**

**Lab #9**

**Objective:**

**CPU Scheduling – Round Robin and Priority**

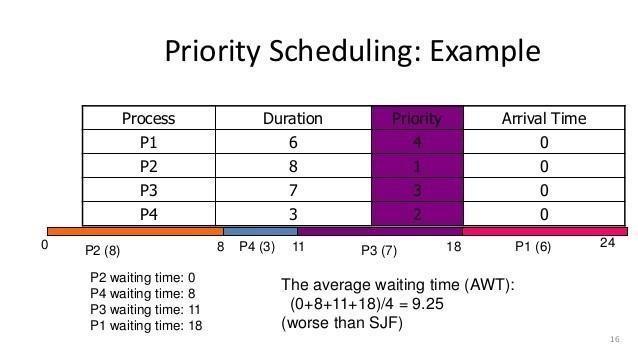
: Implementation of CPU scheduling algorithms; Priority and Round Robin.

| Name of Student | Maha Rehan |
| --- | --- |
| Student ID | 22sp-018-se |
| Date of Lab Conducted |  |

**Objective:** Implementation of CPU scheduling algorithms; Priority and Round Robin.

**THEORY**

**PRIORITY SCHEDULING ALGORITHM**

In priority scheduling algorithm each process has a priority associated with it and as each process hits the queue, it is stored in based on its priority so that process with higher priority is dealt first. It should be noted that equal priority processes are scheduled in FCFS order.

**Implementation –**

1. First input the processes with their arrival time, burst time and priority.

2. Sort the processes, according to arrival time if two process arrival time is same then sort according process priority if two process priority are same then sort according to process number.

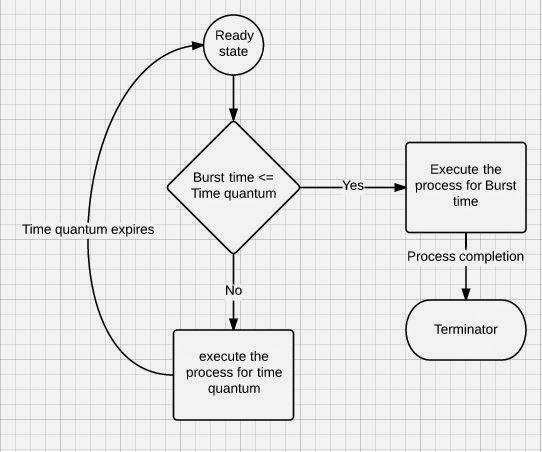
3. Now

simply apply FCFS algorithm.

Each process will be executed according to its priority. Calculate the waiting time and turnaround time of each of the processes accordingly.

**Round Robin Scheduling Algorithm**

Round Robin scheduling algorithm is one of the most popular scheduling algorithms which can actually be implemented in most of the operating systems. The Algorithm focuses on Time Sharing. In this algorithm, every process gets executed in a **cyclic way**. A certain time slice is defined in the system which is called time **quantum**. Each process present in the ready queue is assigned the CPU for that time quantum, if the execution of the process is completed during that time, then the process will **terminate** else the process will go back to the **ready queue** and waits for the next turn to complete the execution.



**Implementation**

For round robin scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times, and the size of the time slice. Time slices are assigned to each process in equal portions and in circular order, handling all processes execution. This allows every process to get an equal chance. Calculate the waiting time and turnaround time of each of the processes accordingly.

**Steps to find Completion times of all processes:**

1. Create an array **rem\_bt[]** to keep track of remaining burst time of processes. This array is initially a copy of bt[] (burst times array)
2. Create another array **ct[]** to store completion times of processes. Initialize this array as 0.
3. Initialize time : t = 0
4. Keep traversing the all processes while all processes are not done. Do following for ith process if it is not done yet.

a- if rem\_bt[i] > quantum

(i) t = t + quantum

(ii) bt\_rem[i] -= quantum;

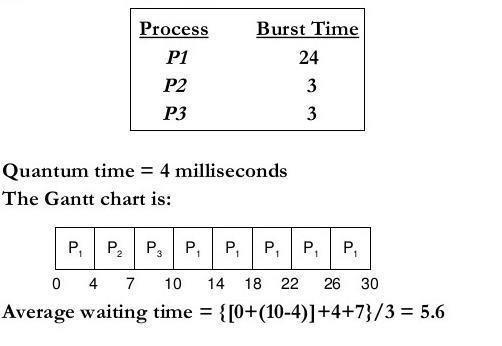
c- else // Last cycle for this process

(i) t = t + bt\_rem[i];

(ii) ct[i]=t;

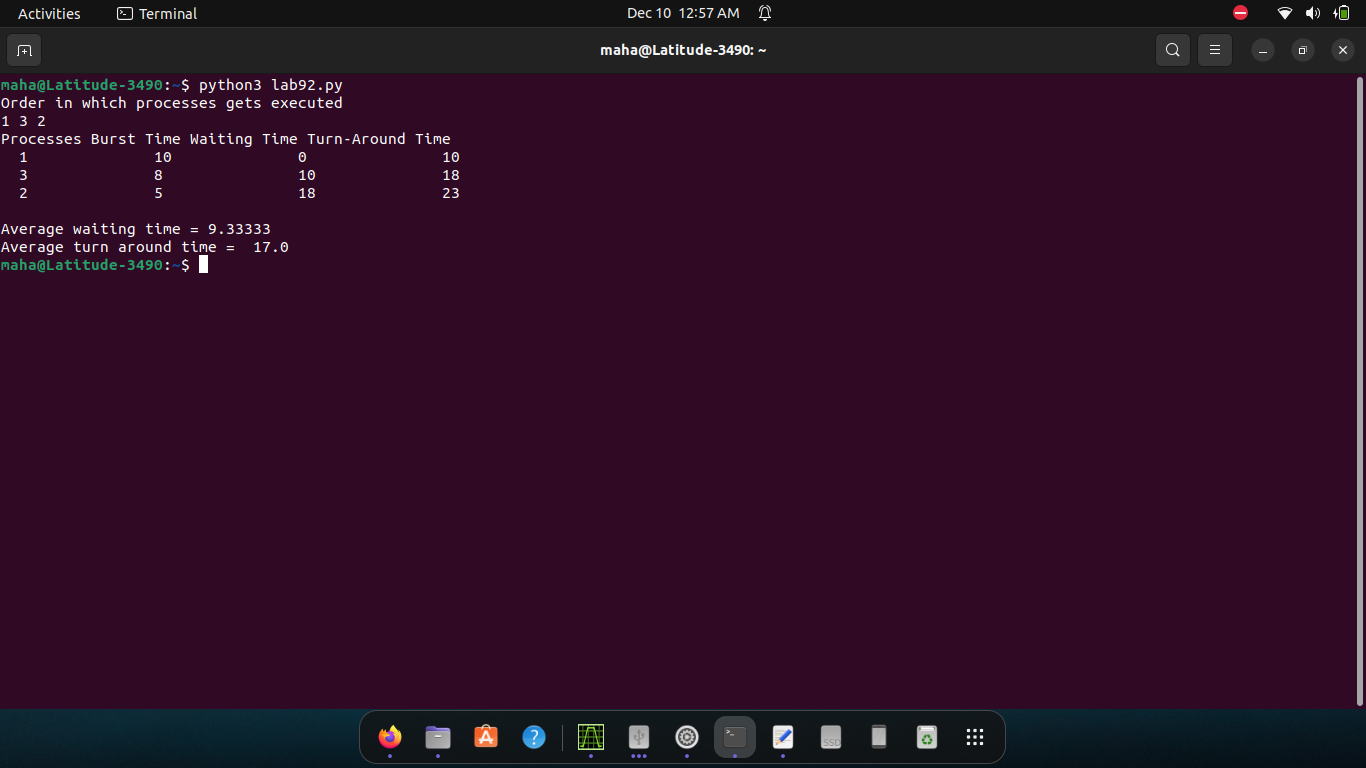
(iii) bt\_rem[i] = 0; // This process is over

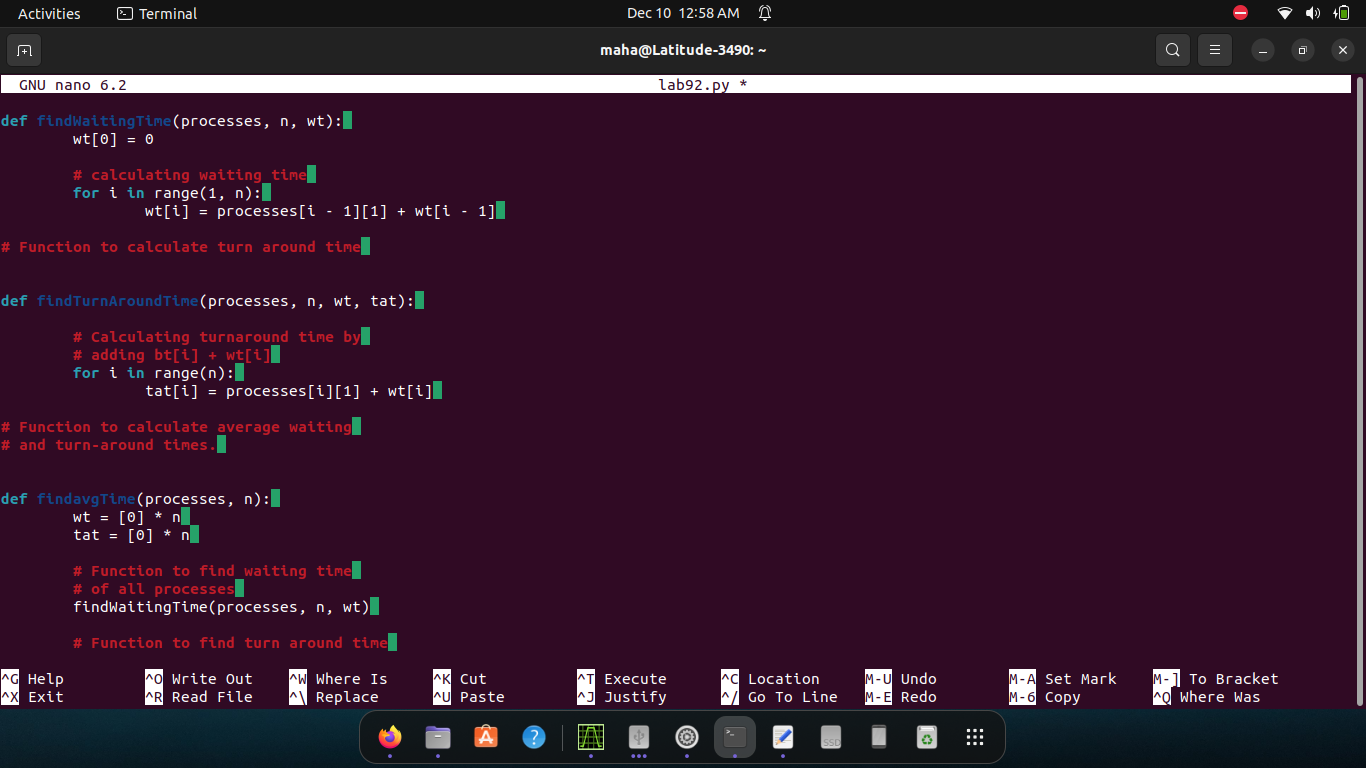
**Round Robin Scheduling Example**

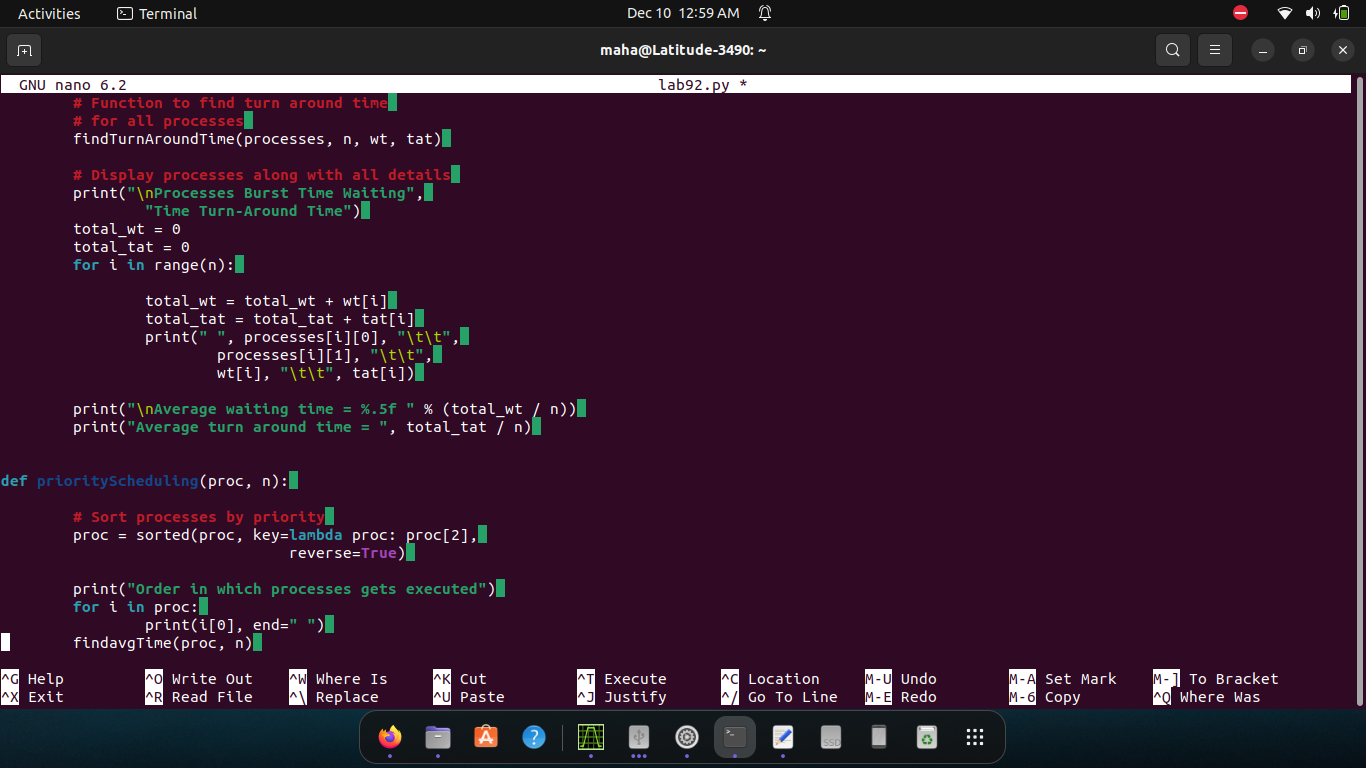


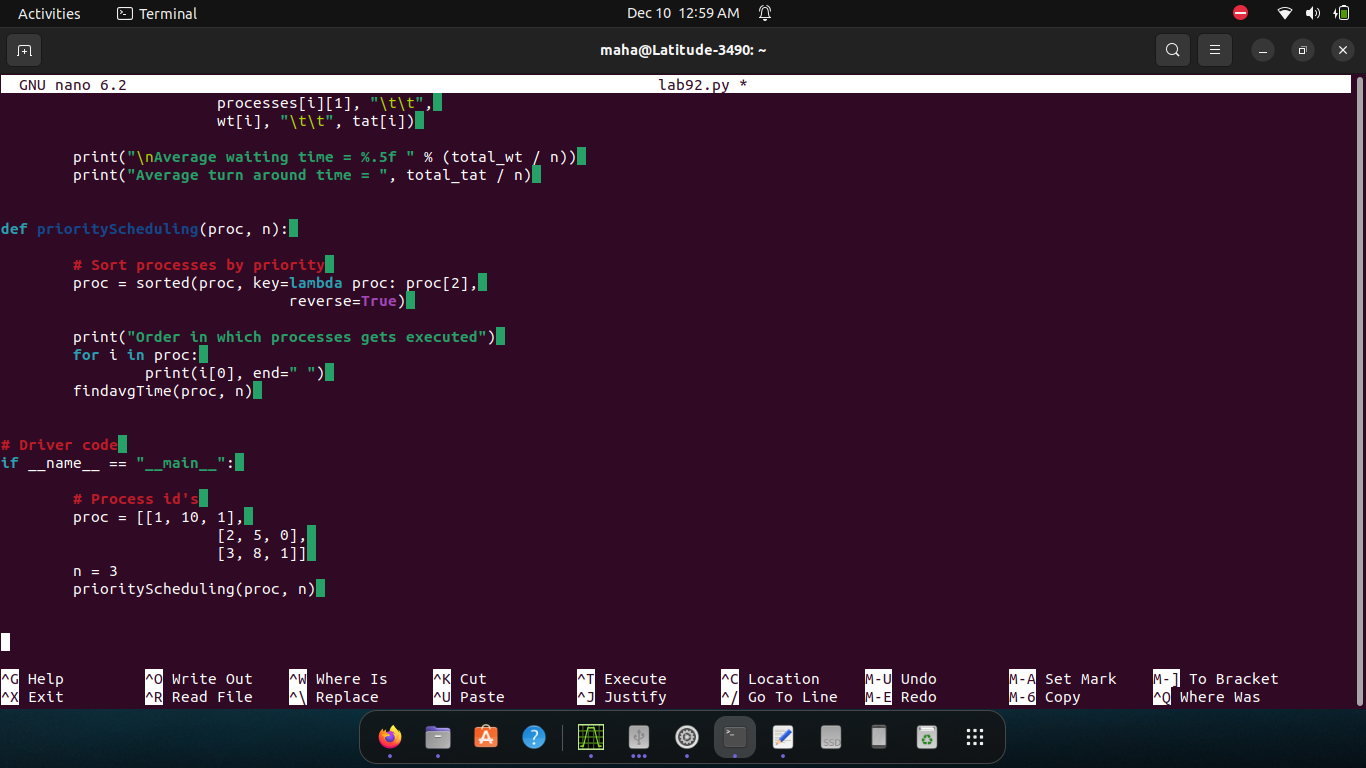
**Lab Exercise(s):**

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

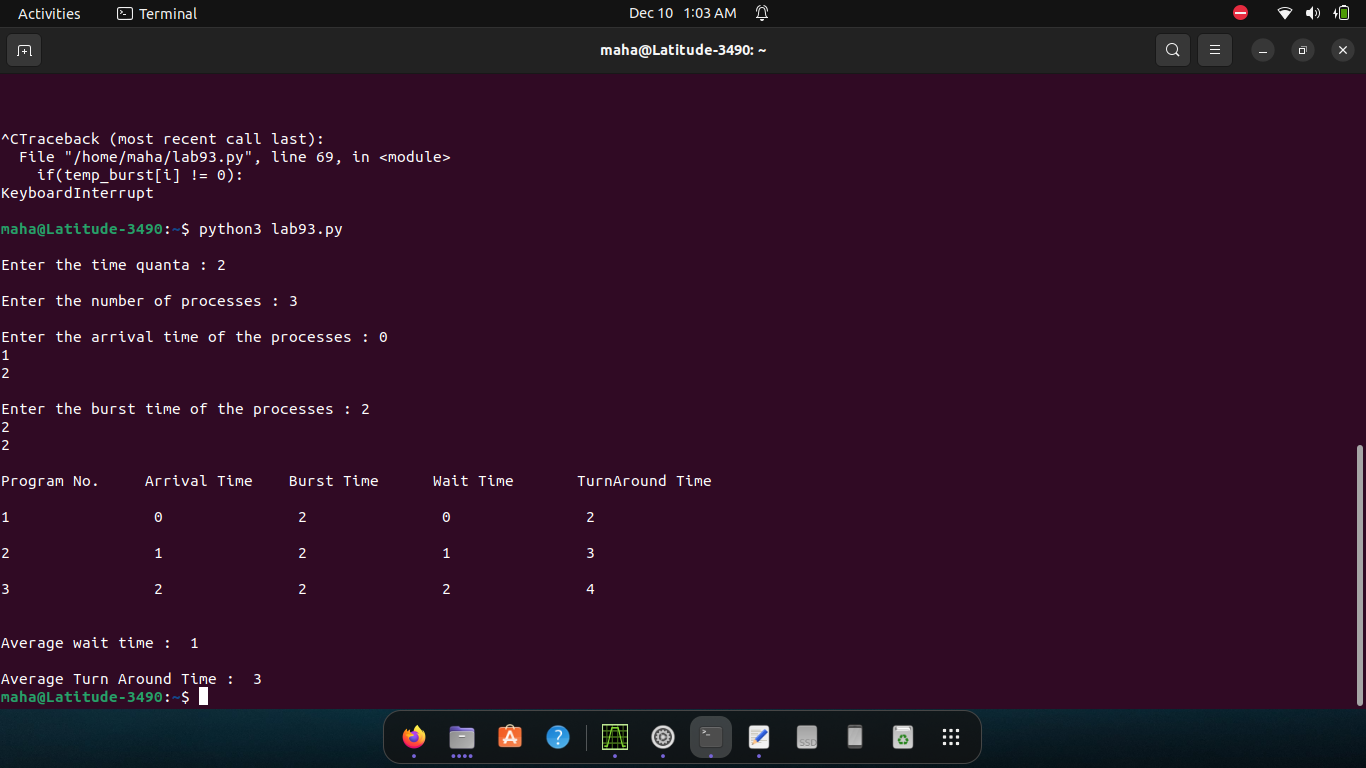


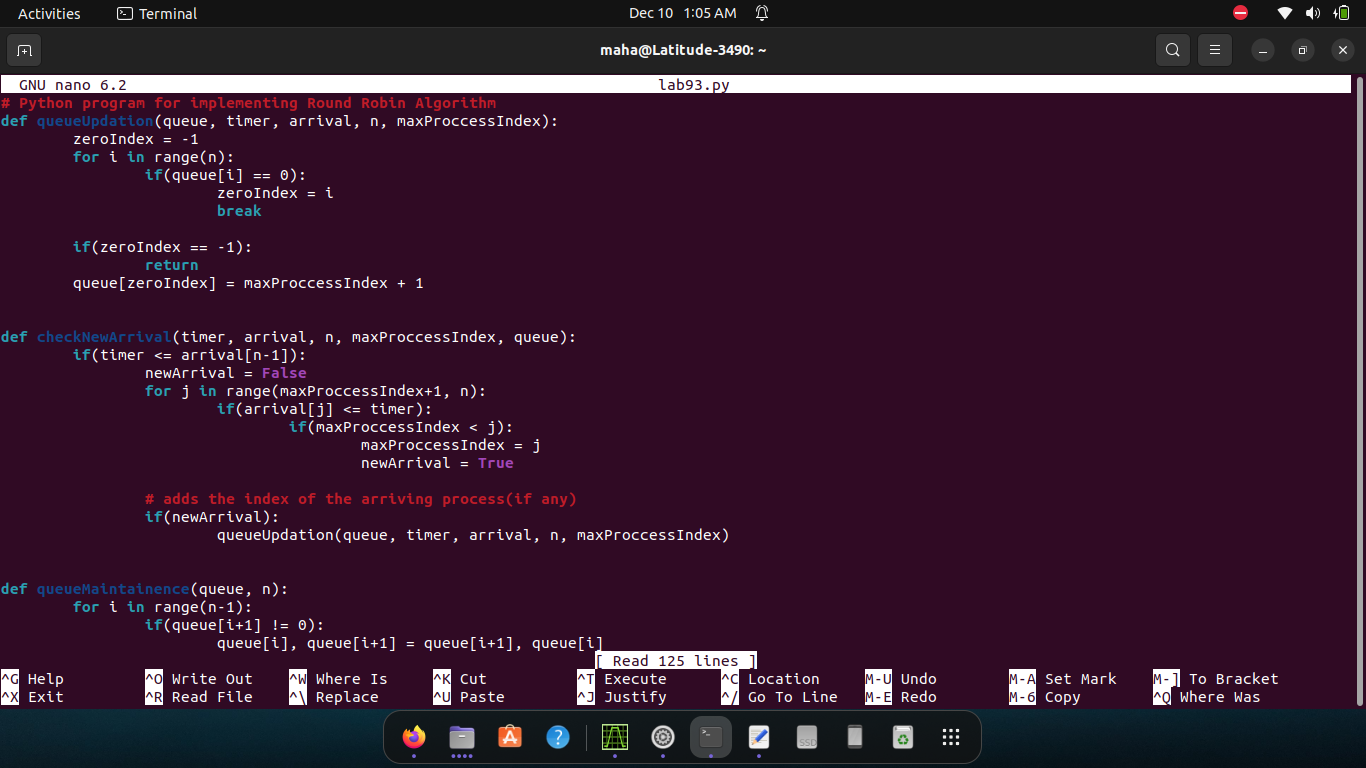


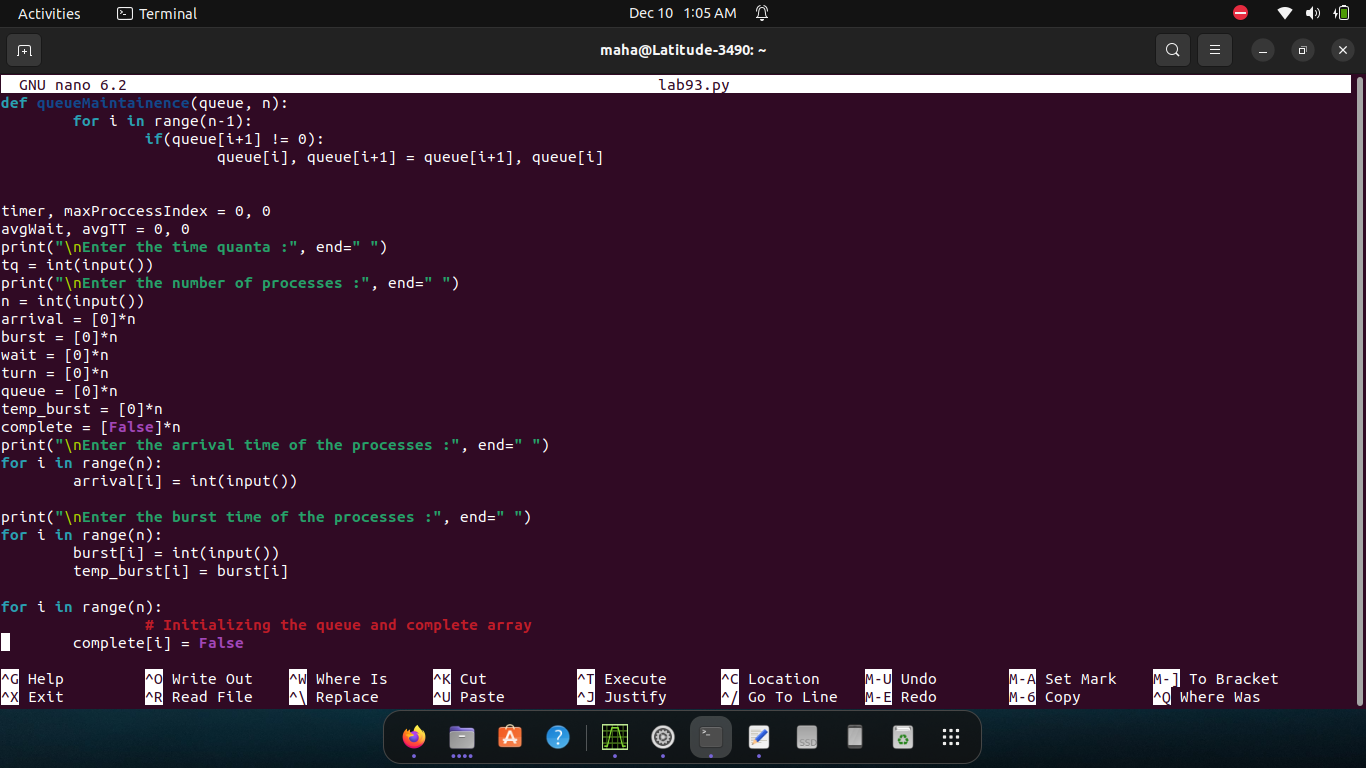


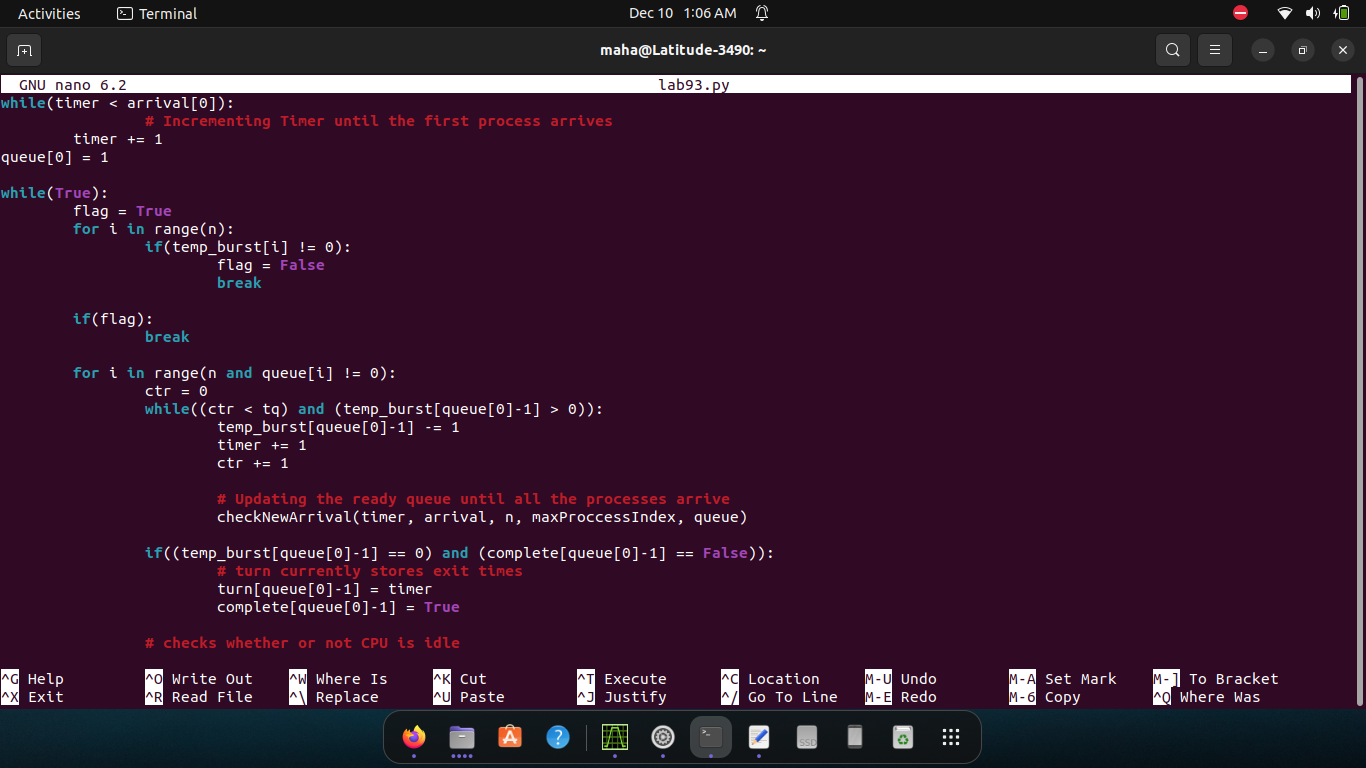


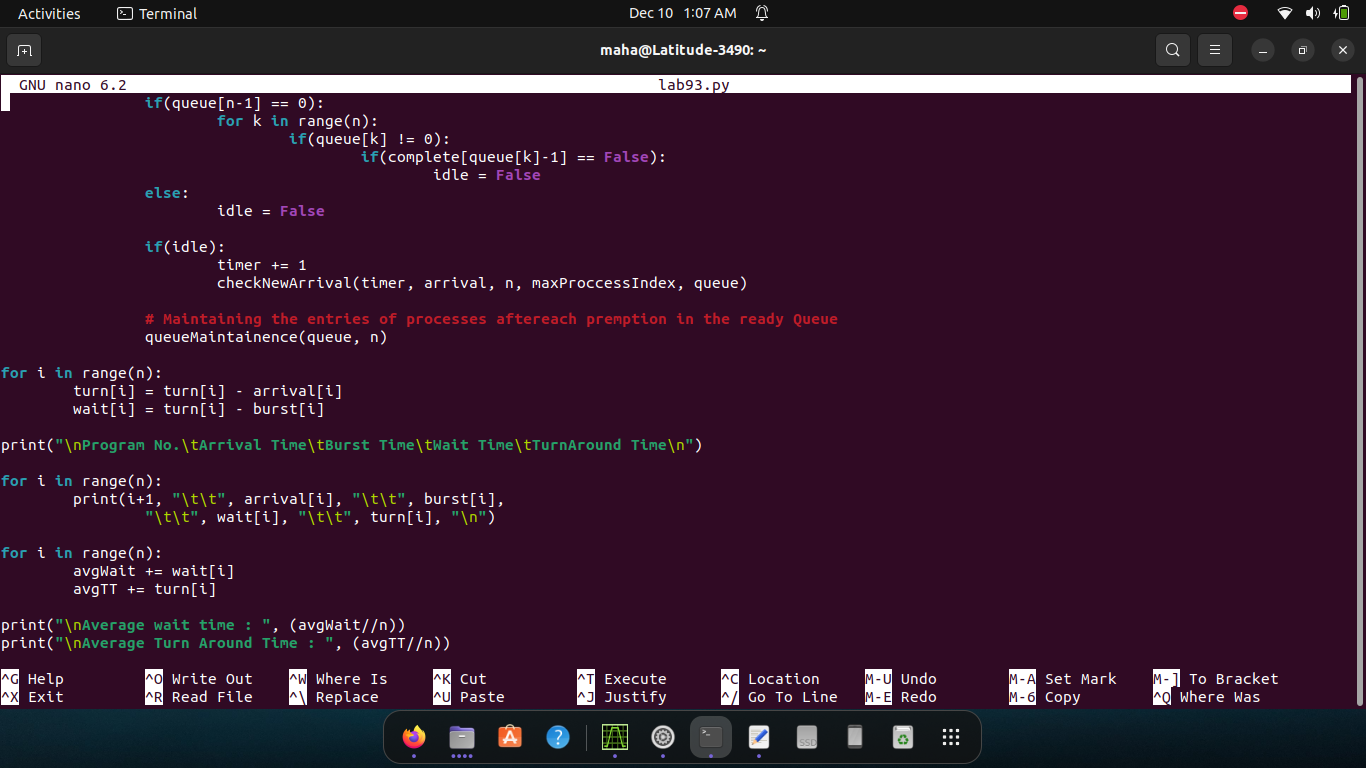
1. Write a Python program to implement and simulate the Round Robin Algorithm.



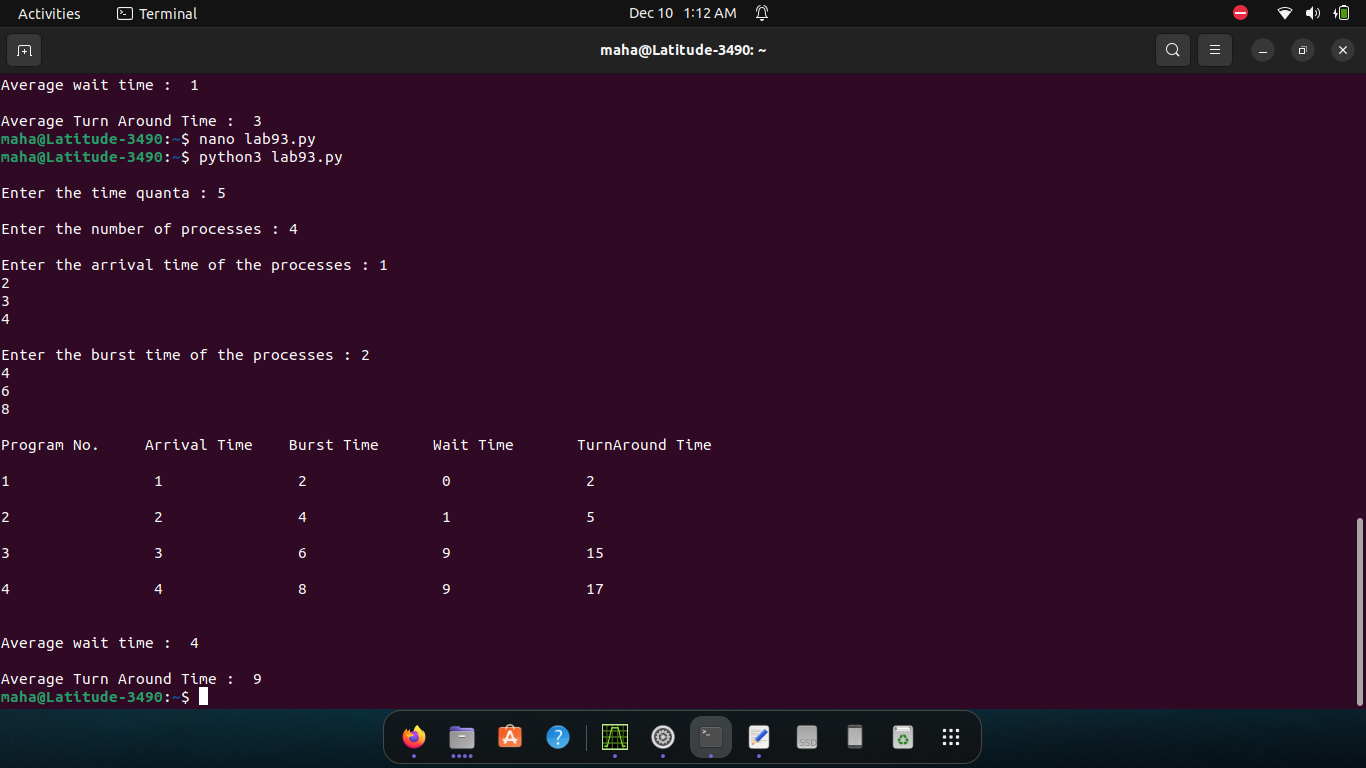




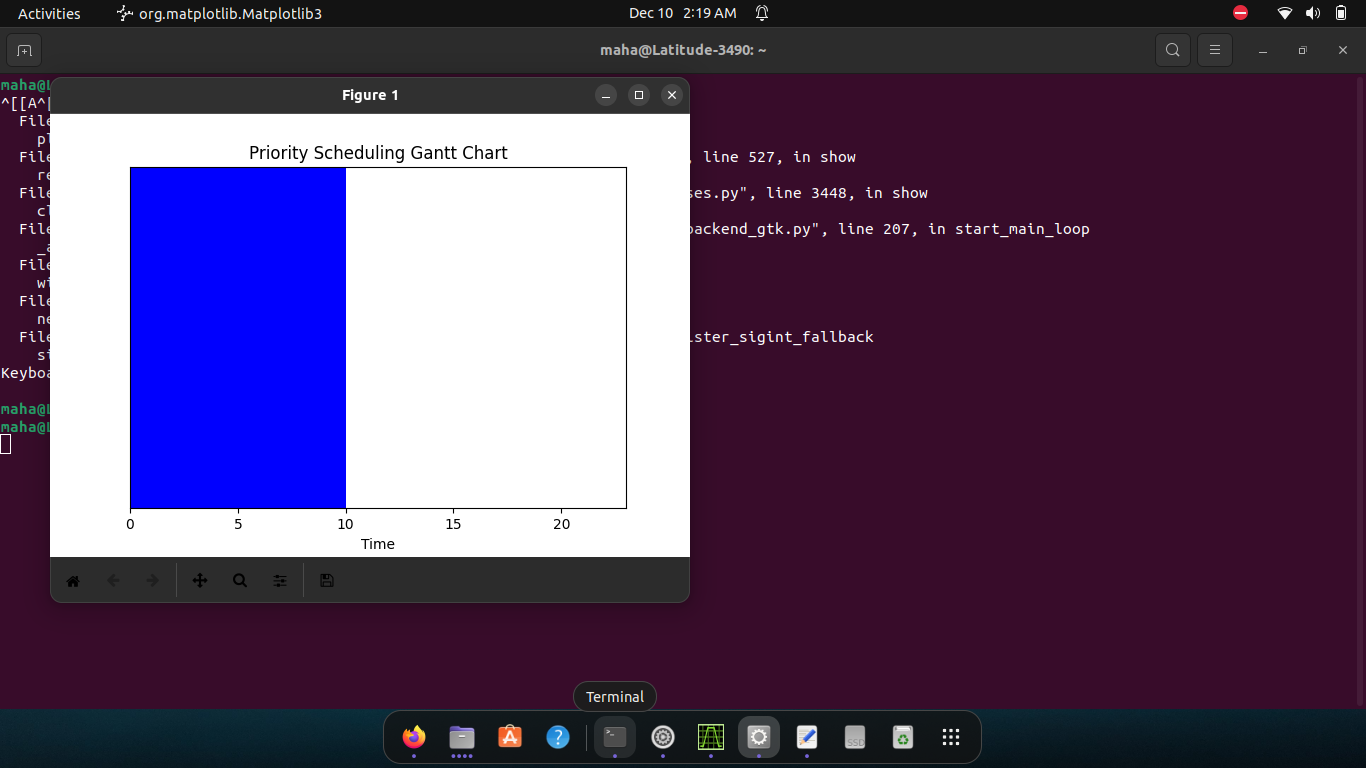


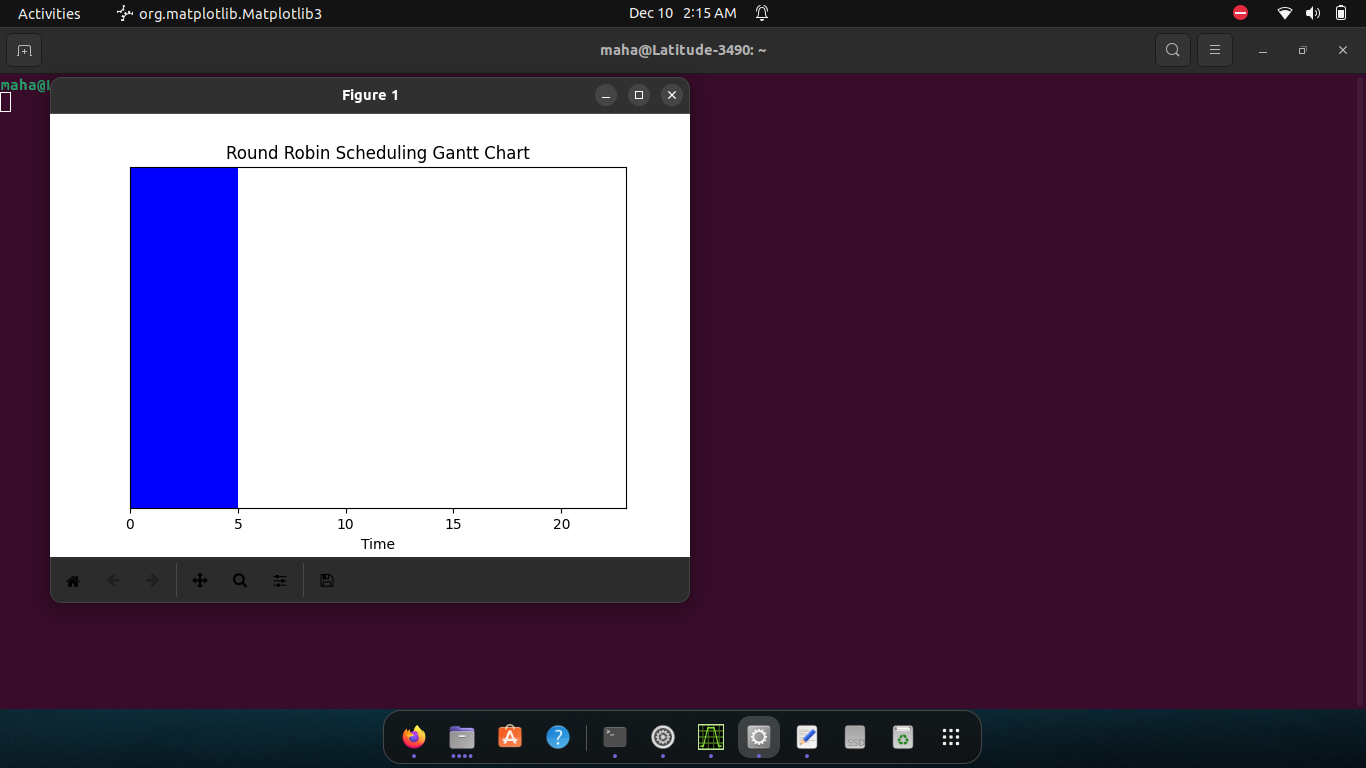


1. Modify both algorithms for the different arrival time.



1. Modify programs to generate 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.

