National University of Modern Languages

Department of Computer Sciences

**Subject:** Data Structures and Algorithm

**Lab Assignment:** 2

**Instructor:** Zainab Malik

**Class:** BSCS 3BE

**Student Roll No:2883**

**Total Marks:** 30

**Muhammad Awais**

**Assigned Task:**

# Q1. Study following process scheduling techniques and suggest appropriate data structures for their implementation along with proper justification/reason. (CLO-3, Marks 15)

A program that invoked for execution is known as a process. For execution, process enters a queue first (i.e. READY queue) and from this READY queue, process gets dequeued and utilize CPU for the execution of respective task. Process may have different attributes such as process ID, priority, arrival time, completion time, execution time, wait time and turn-around time.

* **Process ID**: Unique identifier that differentiate one process from the other. e.g. p1, p2 and p3 etc.
* **Arrival Time**: Time at which the process arrives in the ready queue.
* **Completion Time**: Time at which process completes its execution.
* **Process Execution Time**: Time a process takes for executing its task.
* **Turn Around Time:** Time difference between completion time and arrival time
* **Process Wait Time/Service Time**: Time a process waits in queue due to the execution of other process in the queue or Time difference between turnaround time and execution time.
* **Priority**: Priority is associated with each process that determines the order of execution in

*priority scheduling only*.

For understanding, different scheduling techniques, consider following set of processes along with the associated priority, arrival and execution time *(provided by the user).*

**Table 1: List of Processes**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Process ID** | **Priority** | **Arrival Time** | **Execution Time (milli sec)** | **Wait Time (milli sec)** | **Completion Time** | **Turnaround Time** |
| **P0** | **2** | **0** | **5** |  |  |  |
| **P1** | **0** | **1** | **3** |  |  |  |
| **P2** | **3** | **2** | **8** |  |  |  |
| **P3** | **1** | **3** | **6** |  |  |  |

# First-Come-First-Served Scheduling

As name suggests that the process coming first should be served/ or executed first. e.g. there are 4 process P0, P1, P2 and P3. According to the arrival time (mentioned in the table 1), p0 came first, therefore it should be executed first for 5 sec as per its execution time. As it was the first one so, its service time/wait time is 0. P1 came at 1 (arrival time), so, it should be executed at second number for 3 sec as per the execution time provided by the user. As P1 was executed after P0 whose execution time was 5 sec, that means P1 waited for 5sec before it execution. Similarly, p3, will be executed at the end therefore, its waiting time should be 5+3+8=16 sec.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Process ID** | **Priority** | **Arrival Time** | **Execution Time (milli sec)** | **Wait Time (milli sec)** | **Completion Time** | **Turnaround Time** |
| P0 | 2 | 0 | 5 | 0 | 5 | 5 |
| P1 | 0 | 1 | 3 | 5 | 8 | 7 |
| P2 | 3 | 2 | 8 | 8 | 16 | 14 |
| P3 | 1 | 3 | 6 | 16 | 22 | 19 |

**Suggested Data Structure and why?**

To achieved the **First come first served** behavior in the processing of above scheduling we used a **Linked list base Queue.**

**Justification:**

Because to achieved FCFS implementation we follow the Queue which allows to execute the process in First come First Served.

# Shortest-Job-First Scheduling

The process with shortest execution time should be served/ or executed first. From table 1, as per the execution time process should be executed in following sequence: p1, p0, p3 and p2. However, wait time, completion time and turnaround time should be calculated accordingly.

**Suggested Data Structure and why?**

To achieved the **Shortest Job First** implementation in the processing of above scheduling we used a **Priority Queue.**

we

**Justification:**

**Shortest Job First algorithm** is one of scheduling technique which allow the **shortest** **burst time** **(Takes shortest time to execute)** process to execute first. So, to implement **SJF** we need to priorities the tasks with burst time. To implement the **SJF** we used **Priority Queue. Here** the **highest priority the one which burst time is shortest / minimum** than all and follow this rule till end.

# Longest-Job-First Scheduling

The process with longest execution time should be served/ or executed first. From table 1, as per the execution time process should be executed in following sequence: p2, p3, p0 and p1. However, wait time, completion time and turnaround time should be calculated accordingly.

**Suggested Data Structure and why?**

To achieved the **Longest Job First** implementation in the processing of above scheduling we used a **Priority Queue.**

we

**Justification:**

**Longest Job First algorithm** is one of scheduling technique which allow the **longest** **burst time (Takes highest time to execute)** process to execute first. So, to implement **LJF** we need to priorities the tasks with **burst** **time**. To implement the **LJF** we used **Priority Queue. Here** the **highest priority the one which burst time is longest / maximum** than all and follow this rule till end.

# Priority Scheduling

The process with highest Priority should be processed first/ or executed first. For example from table 1, as per the associated priorities process should be executed in following sequence: p1, p3, p0 and p2. However, wait time, completion time and turnaround time should be calculated accordingly.

**Suggested Data Structure and why?**

To achieved the **Priority Scheduling** implementation, we used a **Priority Queue.**

**Justification:**

**Priority algorithm** is one of scheduling technique which allow the **highest** **priority (here Highest priority means smallest in number integer)** process to execute first.

1. **Round Robin Scheduling** Each process is provided with a fix time to execute; it is called a quantum. Once a process is executed for a given time period, it re-enter the ready **queue** (if it is incomplete) with reduced execution time and again wait for it turns. The wait time, completion time and turnaround time should be calculated accordingly. (explore more over the internet)

**Suggested Data Structure and why?**

To implement **Round Robin Scheduling**, we used a **circular Queue.**

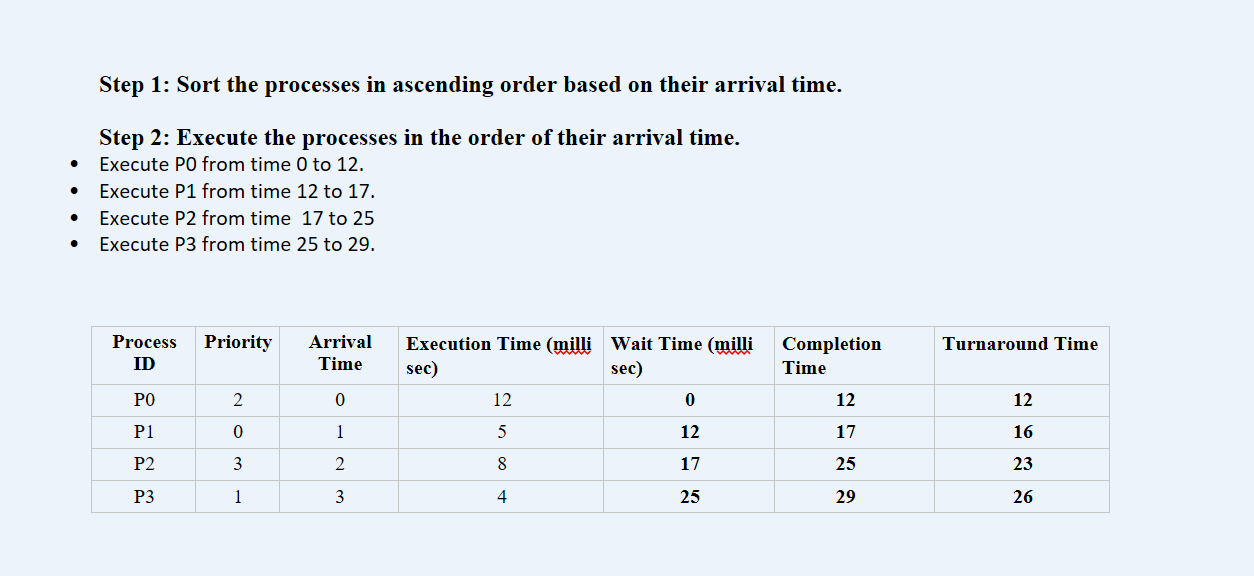
**Justification:**

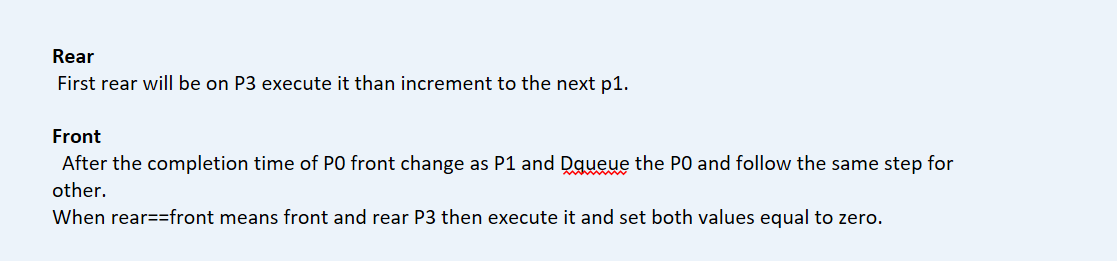
**Round Robin Scheduling algorithm** is one of scheduling technique which follow First come First Served under a time Quantum **(A specific time which share the execution process among tasks. This defined time is called Time Quantum).** If the task does not complete in the **time Quantum** than the process will **interrupted by CPU and proceed next in the Queue.** Once all tasks in Queue gets a given time period to execute incomplete tasks will execute again with **Time Quantum**, So the process work in a **circular behavior**.

# Q2. Demonstrate the working of suggested data structure for each scenario using the following input. For round robin you may consider quantum time as 3 sec. (CLO-2, Marks 25)

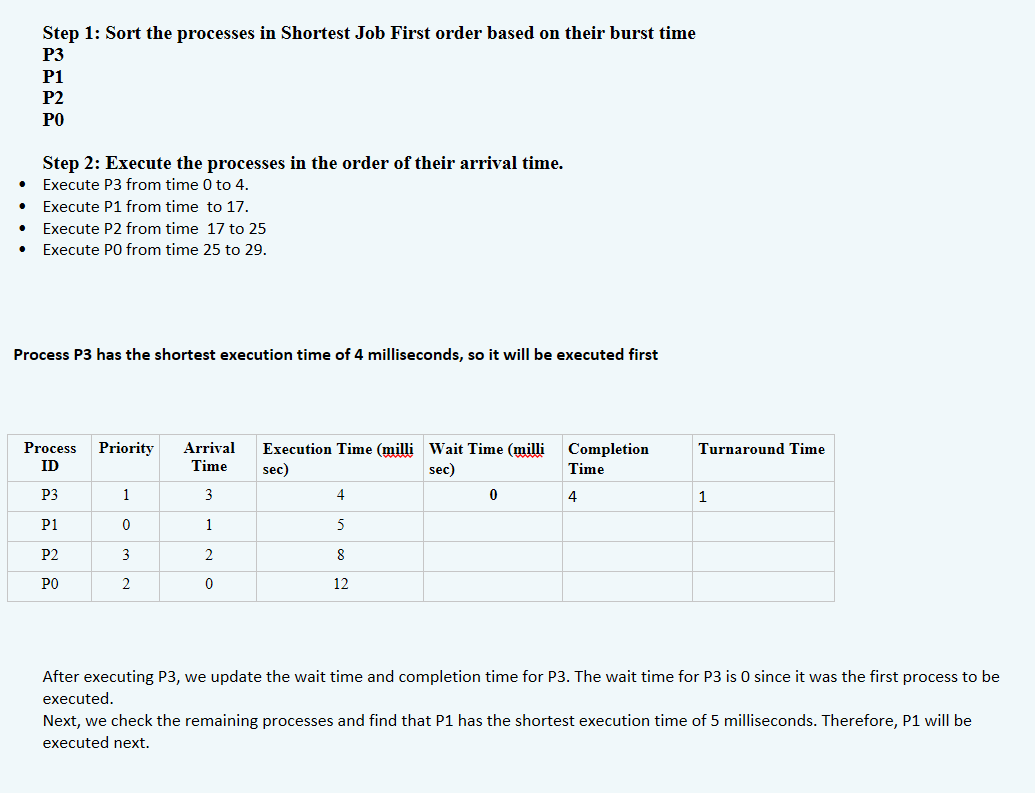
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Process ID** | **Priority** | **Arrival Time** | **Execution Time (milli sec)** | **Wait Time (milli sec)** | **Completion Time** | **Turnaround Time** |
| P0 | 2 | 0 | 12sec |  |  |  |
| P1 | 0 | 1 | 5sec |  |  |  |
| P2 | 3 | 2 | 8sec |  |  |  |
| P3 | 1 | 3 | 4sec |  |  |  |

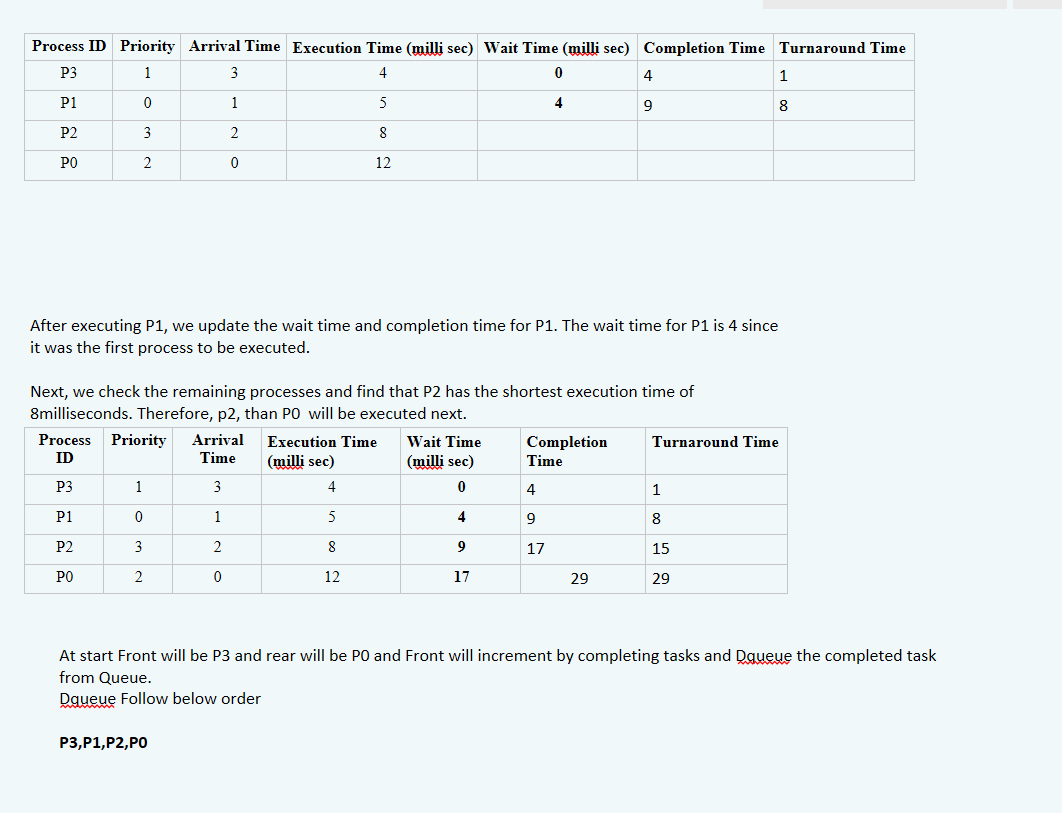
**First come first served**



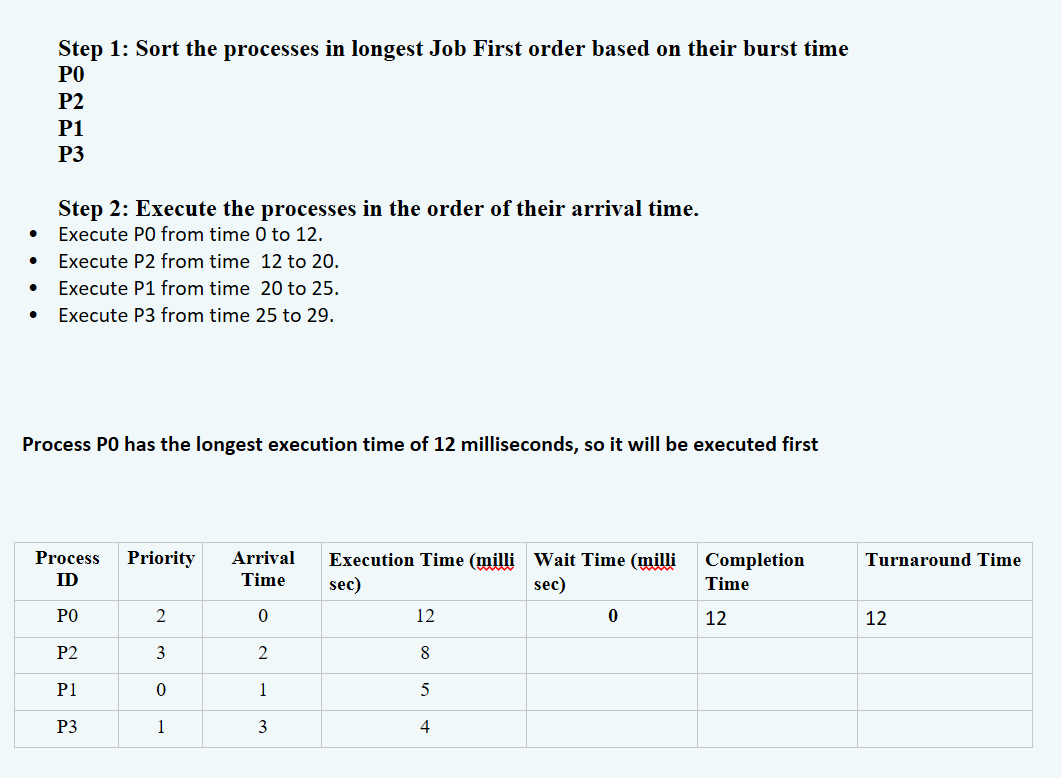


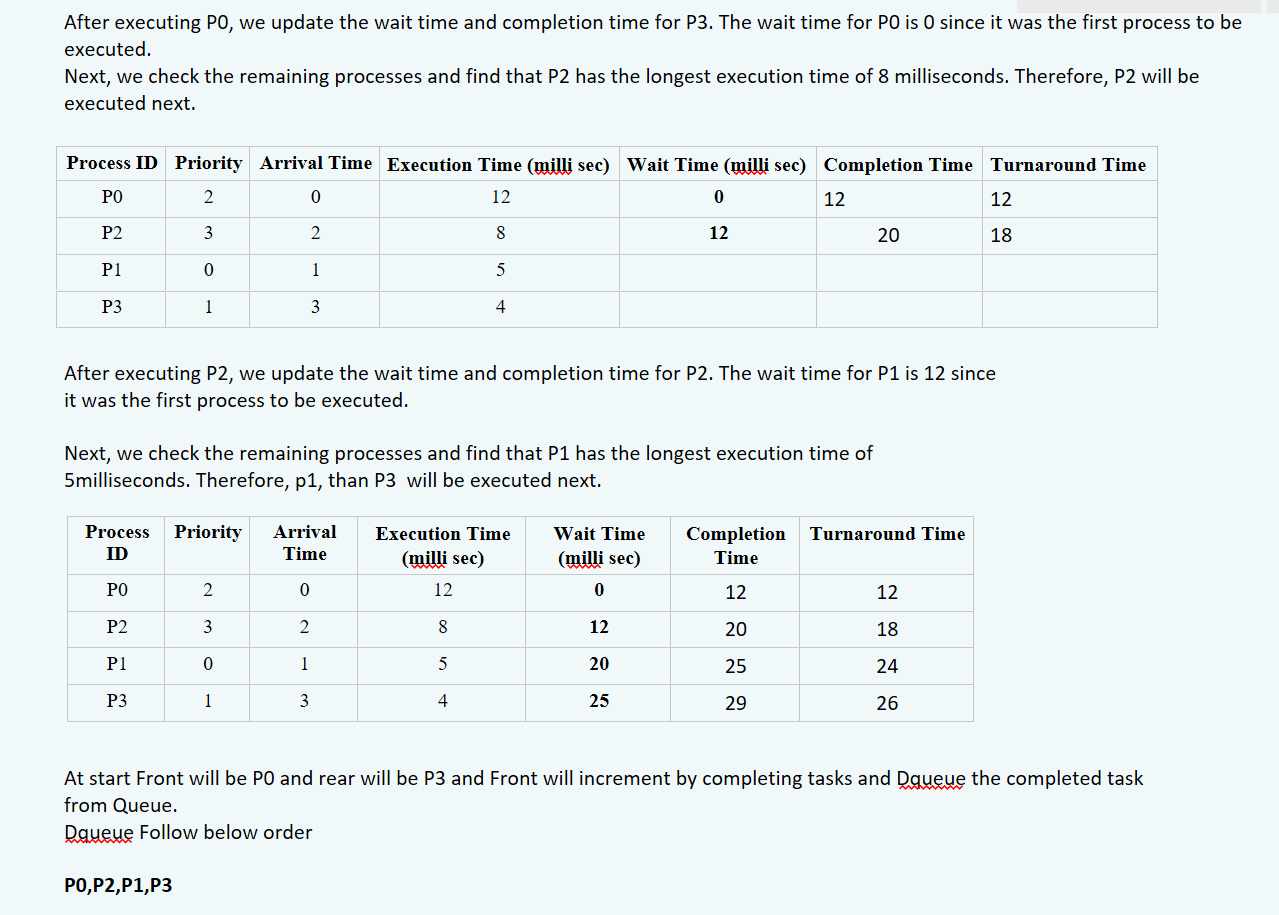
**Shortest Job First**



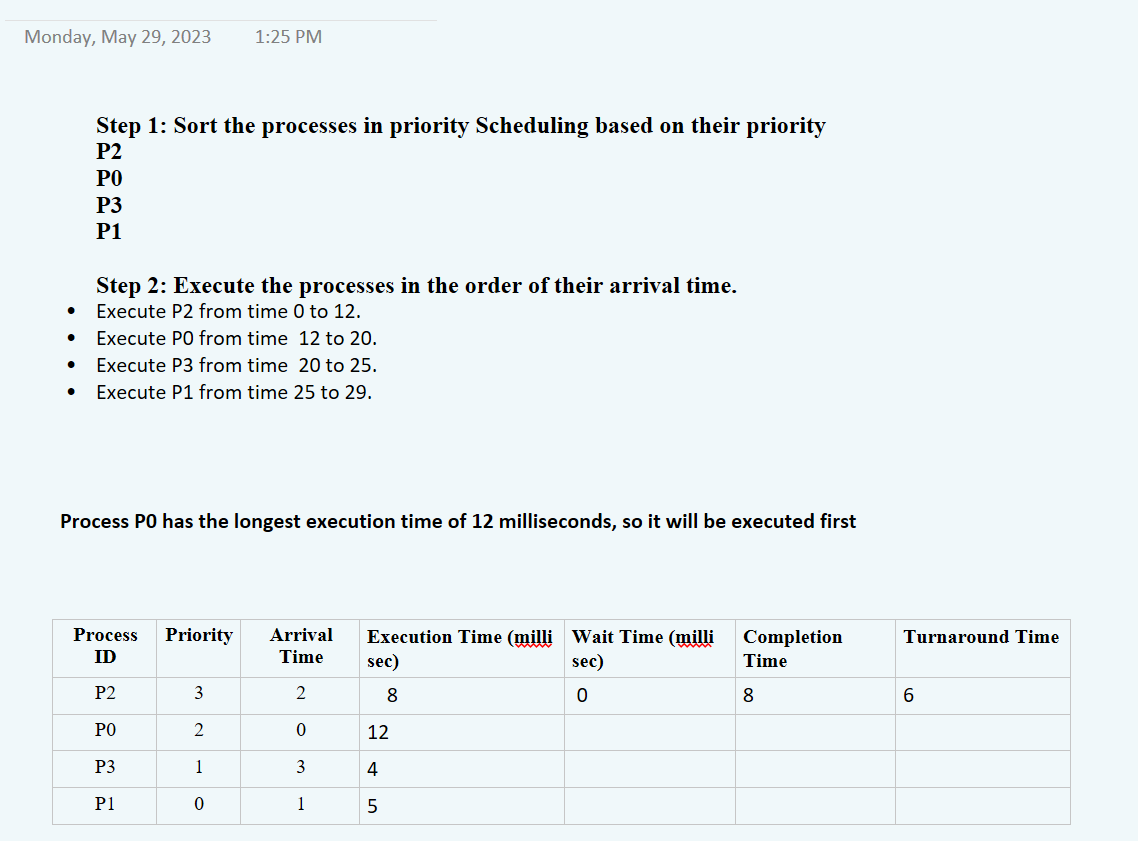


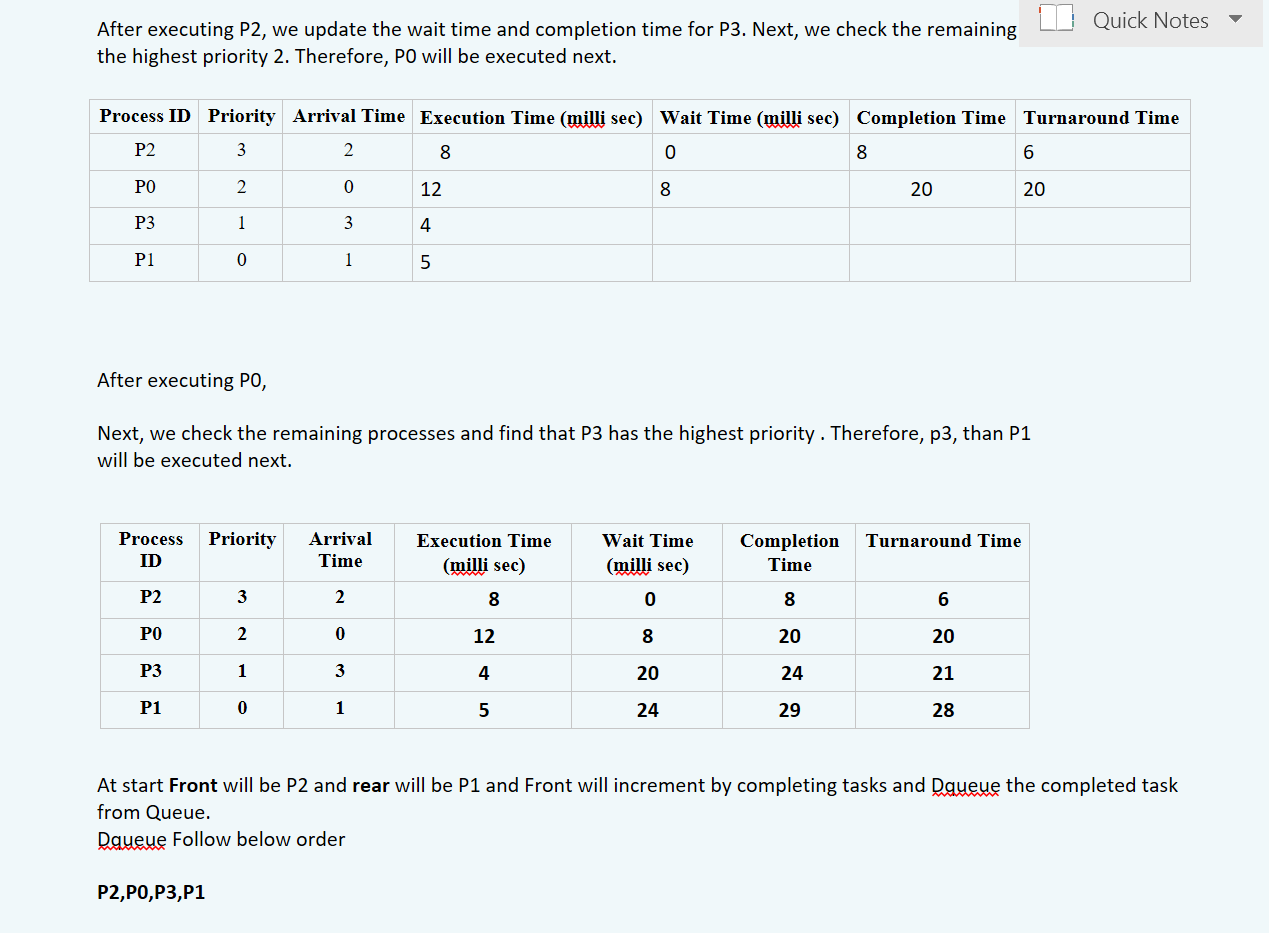
**Longest Job First**



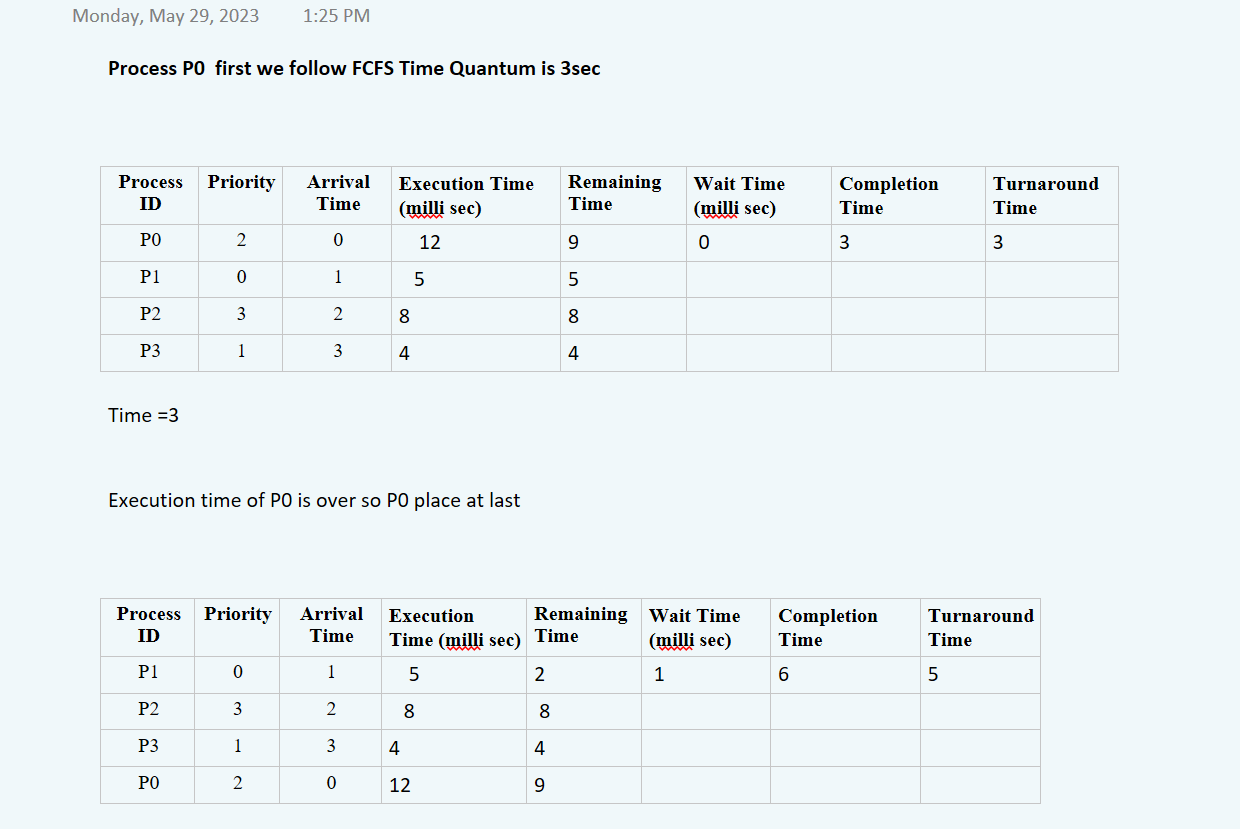


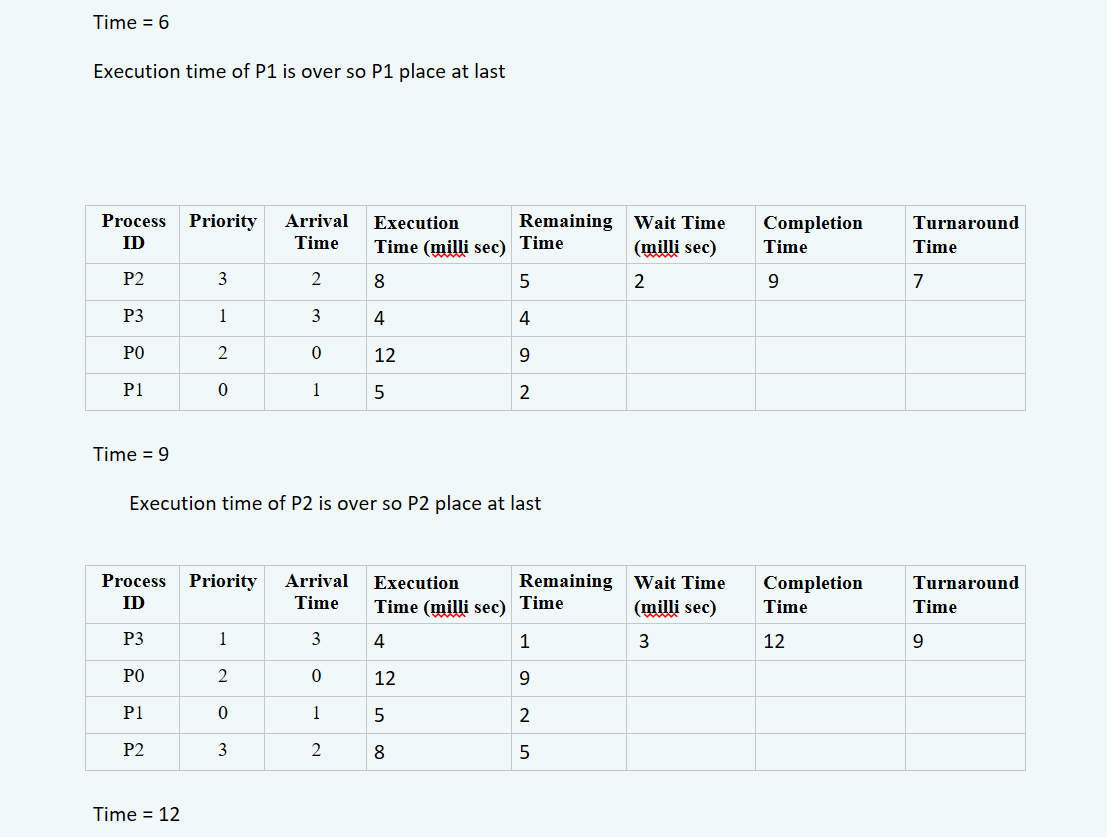
**Priority Scheduling**

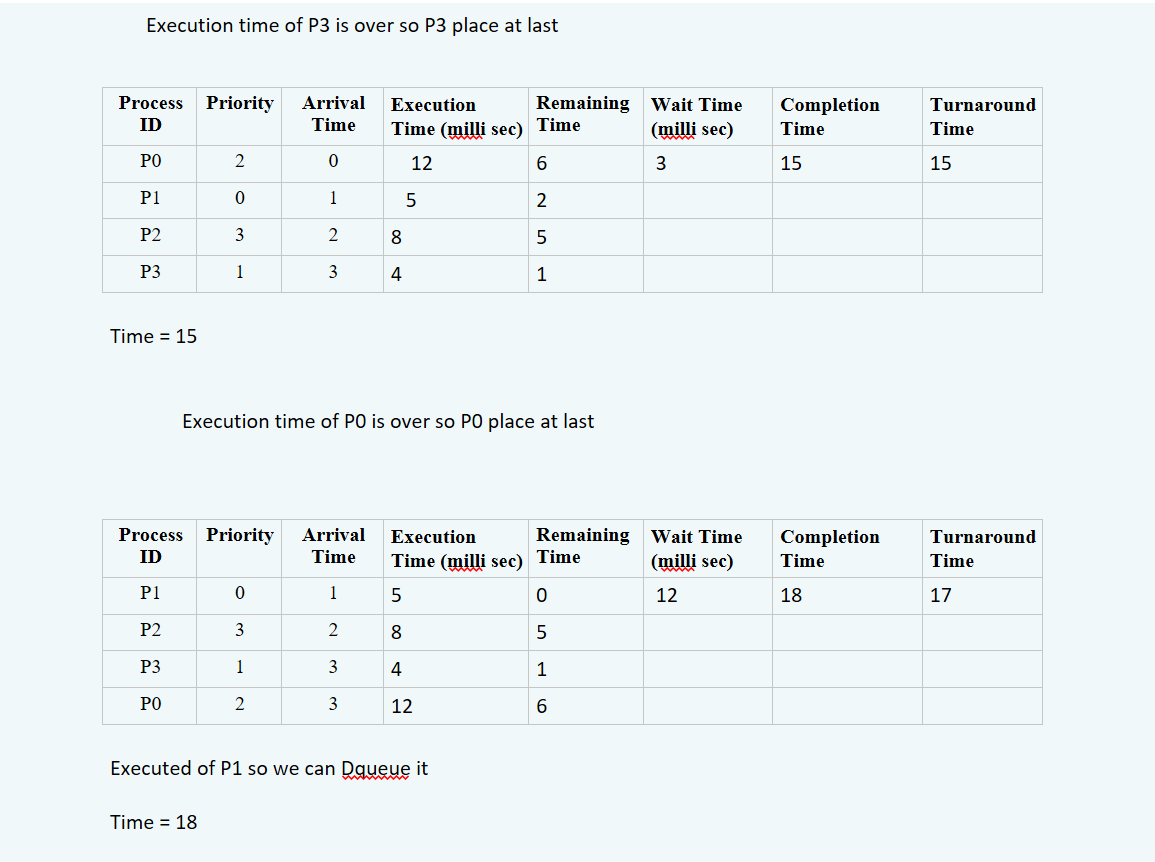
****

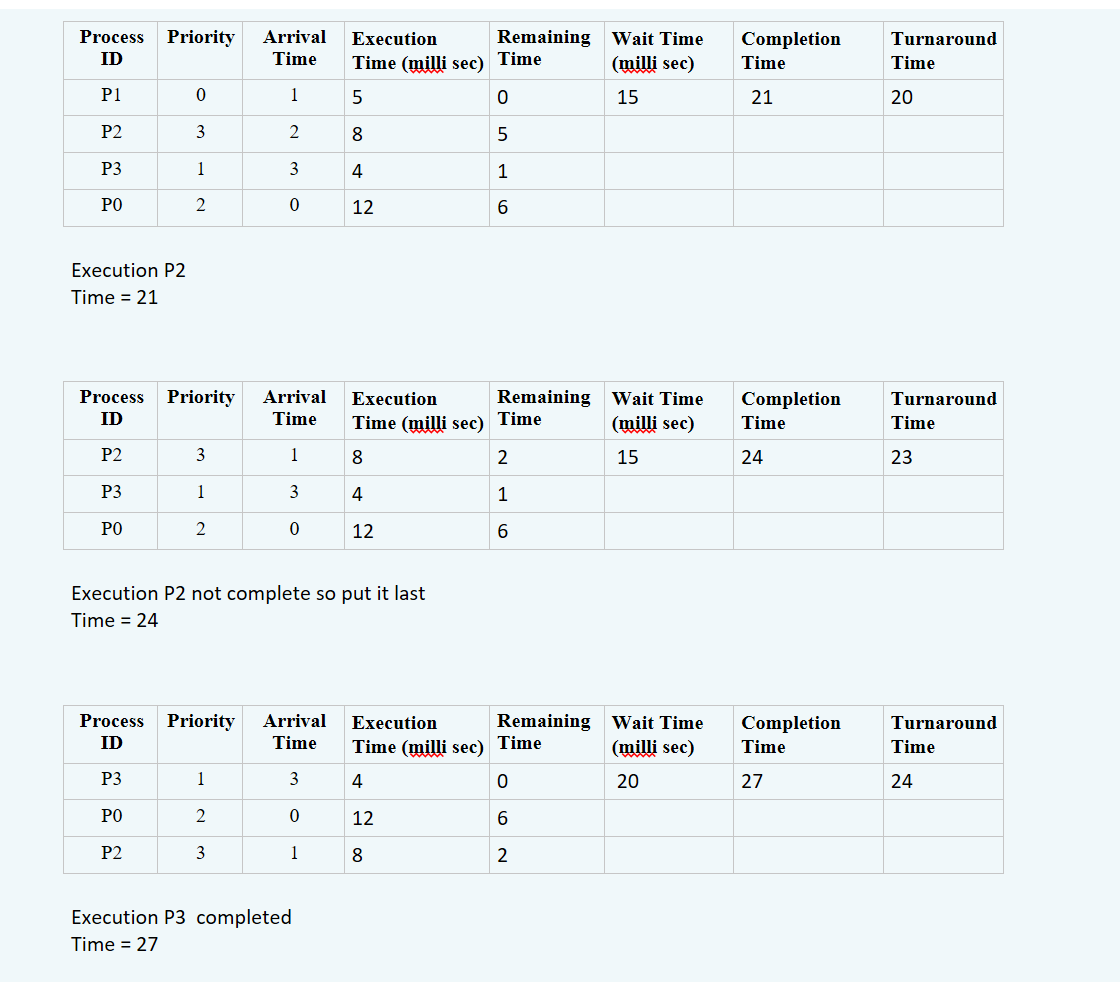
****

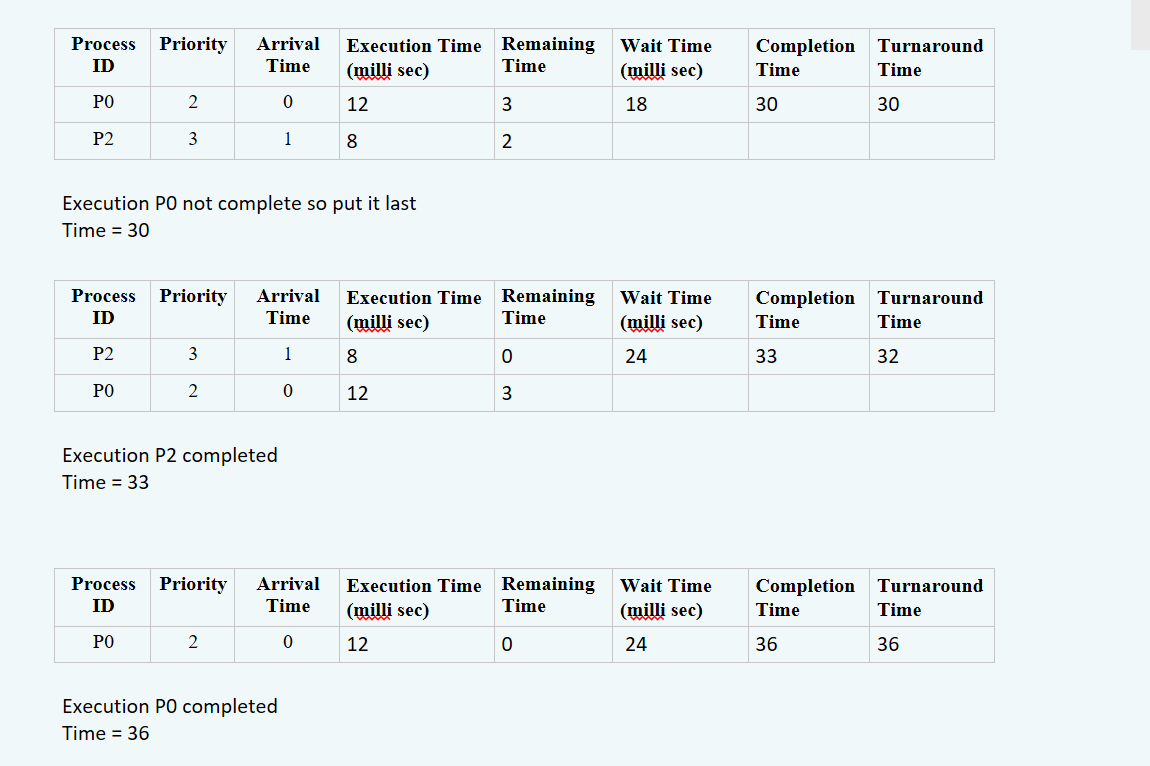
**Round Robin Scheduling**

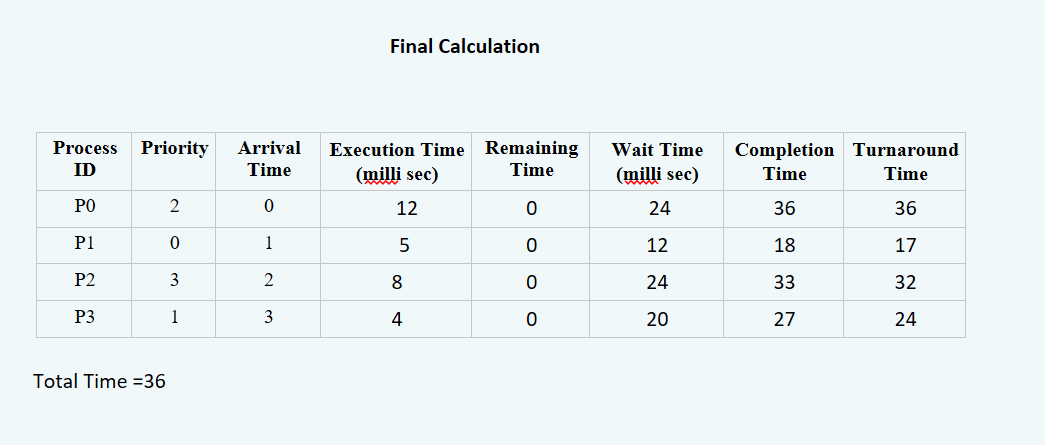
****

****

****

****

****

****

The End