**Part E**

1. Consider the following processes with arrival times and burst times:

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Burst Time |
| P1 | 0 | 5 |
| P2 | 1 | 3 |
| P3 | 2 | 6 |
|  |  |  |

Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.

Solution.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Response Time | Waiting Time | TAT |
| P1 | 0 | 5 | 0 | 0 | 5 |
| P2 | 1 | 3 | 5 | 4 | 7 |
| P3 | 2 | 6 | 8 | 6 | 12 |
| Average | | | 4.333333333 | 3.333333333 | 8 |
|  |  |  |  |  |  |
|  |  | P1 | P2 | P3 |  |
|  | Gantt Time | 0 | 5 | 8 | 14 |

**Average Turnaround Time (TAT) =** (5 + 7 + 12) / 3 = **8**  
**Average Waiting Time (WT) =** (0 + 4 + 6) / 3 = **3.33**

1. Consider the following processes with arrival times and burst times:

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Burst Time |
| P1 | 0 | 3 |
| P2 | 1 | 5 |
| P3 | 2 | 1 |
| P4 | 3 | 4 |

Calculate the average turnaround time using Shortest Job First (SJF) scheduling

Solution:

SJT with preemptive

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Response Time | Waiting Time | TAT |  |  |
| P1 | 0 | 3 | 0 | 1 | 4 |  |  |
| P2 | 1 | 5 | 8 | 7 | 12 |  |  |
| P3 | 2 | 1 | 2 | 0 | 1 |  |  |
| P4 | 3 | 4 | 4 | 1 | 5 |  |  |
| Average | | | 3.5 | 2.25 | 5.5 |  |  |
|  |  |  |  |  |  |  |  |
|  |  | P1 | P3 | P1 | P4 | P2 |  |
|  | Gantt Time | 0 | 2 | 3 | 4 | 8 | 13 |

SJT with Non preemptive :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Response Time | Waiting Time | TAT |  |
| P1 | 0 | 3 | 0 | 0 | 3 |  |
| P2 | 1 | 5 | 8 | 7 | 12 |  |
| P3 | 2 | 1 | 2 | 1 | 2 |  |
| P4 | 3 | 4 | 4 | 1 | 5 |  |
| Average | | | 3.5 | 2.25 | 5.5 |  |
|  |  |  |  |  |  |  |
|  |  | P1 | P3 | P4 | P2 |  |
|  | Gantt Time | 0 | 3 | 4 | 8 | 13 |

1. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority)

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Priority |
| P1 | 0 | 6 | 3 |
| P2 | 1 | 4 | 1 |
| P3 | 2 | 7 | 4 |
| P4 | 3 | 2 | 2 |

Solution:

Priority Scheduling(Preemptive Scheduling)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Priority | Response Time | Waiting Time | TAT |
| P1 | 0 | 6 | 3 | 0 | 6 | 12 |
| P2 | 1 | 4 | 1 | 1 | 0 | 4 |
| P3 | 2 | 7 | 4 | 12 | 10 | 17 |
| P4 | 3 | 2 | 2 | 5 | 2 | 4 |
|  |  |  | Average | 4.5 | 4.5 | 9.25 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Gantt Chart | P1 | P2 | P4 | P1 | P3 |  |
|  | 0 | 1 | 5 | 7 | 12 | 19 |

Priority Scheduling(Non Preemptive Scheduling)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Priority | Response Time | Waiting Time | TAT |
| P1 | 0 | 6 | 3 | 0 | 0 | 6 |
| P2 | 1 | 4 | 1 | 1 | 5 | 9 |
| P3 | 2 | 7 | 4 | 12 | 10 | 17 |
| P4 | 3 | 2 | 2 | 5 | 7 | 9 |
|  |  |  | Average | 4.5 | 5.5 | 10.25 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Gantt Chart | P1 | P2 | P4 | P3 |  |  |
|  | 0 | 6 | 10 | 12 | 19 |  |

1. Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Burst Time |
| P1 | 0 | 4 |
| P2 | 1 | 5 |
| P3 | 2 | 2 |
| P4 | 3 | 3 |

Solution:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Response Time | Waiting Time | TAT |  |  |  |  |
| P1 | 0 | 4 | 0 | 6 | 10 |  |  |  |  |
| P2 | 1 | 5 | 2 | 8 | 13 |  |  |  |  |
| P3 | 2 | 2 | 4 | 2 | 4 |  |  |  |  |
| P4 | 3 | 3 | 6 | 7 | 10 |  |  |  |  |
| Average | | | 3 | 5.75 | 9.25 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Gantt Chart | P1 | P2 | P3 | P4 | P1 | P2 | P4 | P2 |  |
|  | 0 | 2 | 4 | 6 | 8 | 10 | 12 | 13 | 14 |

1. Consider a program that uses the fork() system call to create a child process. Initially, the parent process has a variable x with a value of 5. After forking, both the parent and child processes increment the value of x by 1.

What will be the final values of x in the parent and child processes after the fork() call?

Ans.

* The parent process has a variable **x = 5**.
* After calling fork(), **both parent and child processes increment x by 1**.
* We need to determine the final values of x in both processes.

1. Before fork() is called
   1. The parent process has x = 5.
2. When fork() is called
   1. A new child process is created.
   2. The child process gets a copy of the parent’s memory (including x = 5).
3. After fork() Execution
   1. Both parent and child have their own separate copies of x.
   2. Parent process increments x: x = 6
   3. Child process increments x: x = 6
   4. Changes in one process do not affect the other because each has its own copy of the variable.

| 1. **Process** | **Final Value of x** |
| --- | --- |

|  |  |
| --- | --- |
| **Parent Process** | x = 6 |

|  |  |
| --- | --- |
| **Child Process** | x = 6 |