**Experiment 1 – FCFS CPU Scheduling**

1. Aim:

To Simulate FCFS CPU scheduling algorithm to find turnaround time and waiting time.

1. Algorithm:
2. Start
3. Declare nop,burst[50],completion[50],waitingT[50],turnAroundT[50], totalTurnAroundT=0, totalWaitingT=0
4. Read the number of Processes to be inserted in nop
5. Read the Burst Time of Processes in array burstT
6. For each Process index as i, goto step 5.1
   1. Calculate completion[i] = burstT[i] + completionT[i-1]
   2. turnAroundT[i]=completionT[i]
   3. waitingT[i]=turnAroundT[i] - burstT[i];
   4. totalTurnAroundT= totalTurnAroundT + turnAroundT[i];
   5. totalWaitingT= totalWaitingT + waitingT[i];
7. Display the Values
8. Stop
9. URL:

<https://ide.geeksforgeeks.org/teE2xkdUMD>

1. Result:

The program for FCFS was executed and verified successfully.

**Experiment 2 – Shortest Job First CPU Scheduling**

1. Aim:

To Simulate SJF CPU scheduling algorithm to find average turnaround time and average waiting time.

1. Algorithm:
   1. Start
   2. Declare a struct Process with elements id, burstT, completionT , turnAroundT , waitingT
   3. Declare totalTurnAroundT=0, totalWaitingT=0, struct Process p[50]
   4. Read the number of Processes to be inserted in nop
   5. Read the Burst Time of each Processes
   6. Sort the Processes according to its burst time in ascending order
   7. For each Process index as i, goto step 7.1
      1. Calculate p[i].completionT = p[i].burstT + p[i-1].completionT
      2. p[i].turnAroundT = p[i].completionT
      3. p[i].waitingT= p[i].turnAroundT - p[i].burstT;
      4. totalTurnAroundT= totalTurnAroundT + p[i]. turnAroundT;
      5. totalWaitingT= totalWaitingT + p[i]. waitingT;
   8. Display the Values
   9. Stop
2. URL:

<https://ide.geeksforgeeks.org/oTTJVXPjWQ>

1. Result:

The program for SJF was executed and verified successfully.

**Experiment 3 – Non-Preemptive Priority CPU Scheduling**

1. Aim:

To Simulate Priority CPU scheduling algorithm to find average turnaround time and average waiting time.

1. Algorithm:
   1. Start
   2. Declare a struct Process with elements id, priority, burstT, completionT , turnAroundT , waitingT
   3. Declare totalTurnAroundT=0, totalWaitingT=0, struct Process p[50]
   4. Read the number of Processes to be inserted in nop.
   5. Read the Priority and Burst Time of each Processes.
   6. Sort the Processes according to its priority in ascending order
   7. For each Process index as I, goto step 7.1
      1. Calculate p[i].completionT = p[i].burstT + p[i-1].completionT
      2. p[i].turnAroundT = p[i].completionT
      3. p[i].waitingT= p[i].turnAroundT - p[i].burstT;
      4. totalTurnAroundT= totalTurnAroundT + p[i]. turnAroundT;
      5. totalWaitingT= totalWaitingT + p[i]. waitingT;
   8. Display the Values
   9. Stop
2. URL:

<https://ide.geeksforgeeks.org/LFlOZYEQsP>

1. Result:

The program for Non-preemptive Priority Scheduling was executed and verified successfully.

**Experiment 4 –Round Robin CPU Scheduling Algorithm**

1. Aim:

To Simulate Round Robin CPU scheduling algorithm to find average turnaround time and average waiting time.

1. Algorithm:

// to be completed

1. URL:

<https://ide.geeksforgeeks.org/LFlOZYEQsP>

1. Result:

The program for Non-preemptive Priority Scheduling was executed and verified successfully.