ASSIGNMENT NO.5

(Scheduling Algorithms)

PROGRAM:-

```
import java.util.*;
class Process {
   int id;
   int burstTime;
   int arrivalTime;
   int priority;
   int remainingTime;
   Process(int id, int burstTime, int arrivalTime, int priority) {
      this.id = id;
      this.burstTime = burstTime;
      this.arrivalTime = arrivalTime;
      this.priority = priority;
      this.remainingTime = burstTime;
   }
}
public class SchedulerProgram {
   public static void main(String[] args) {
     String continueChoice;
      Scanner scanner = new Scanner(System.in);
      ArrayList<Process> processes = new ArrayList<>();
      while (true) {
         System.out.print("Enter the number of processes: ");
         int n = scanner.nextInt();
         processes.clear(); // Clear the previous process list
         for (int i = 0; i < n; i++) {
            System.out.print("Enter arrival time for process " + (i + 1) + ": ");
            int arrivalTime = scanner.nextInt();
            System.out.print("Enter burst time for process " + (i + 1) + ": ");
            int burstTime = scanner.nextInt();
            System.out.print("Enter priority for process " + (i + 1) + ": ");
            int priority = scanner.nextInt();
            processes.add(new Process(i + 1, burstTime, arrivalTime, priority));
         }
      do
          {
         System.out.println("Choose a scheduling algorithm:");
         System.out.println("1. FCFS\n2. SJF\n3. SRTF\n4. Priority\n5. Round Robin");
             System.out.println("\nEnter your Choice:");
         int choice = scanner.nextInt();
```

```
        System.out.println("Process Data:");
        displayProcessData(processes);

        switch (choice) {
            case 1:
                runFCFS(new ArrayList<>(processes)); // Create a copy to avoid modifying the
original list
                break;
            case 2:
                runSJF(new ArrayList<>(processes));
                break;
            case 3:
                runSRTF(new ArrayList<>(processes));
                break;
            case 4:
                runPriority(new ArrayList<>(processes));
                break;
            case 5:
                System.out.print("Enter time quantum for Round Robin: ");
                int timeQuantum = scanner.nextInt();
                runRoundRobin(new ArrayList<>(processes), timeQuantum);
                break;
            default:
                System.out.println("Invalid choice.");
        }
        System.out.print("Do you want to continue (yes/no)? ");
        continueChoice = scanner.next().toLowerCase();
         }while(continueChoice.equals("yes"));break;  }
    }
    private static void displayProcessData(ArrayList<Process> processes) {
        System.out.println("Process\t\tBurst Time\t\tArrival Time\t\tPriority");
        for (Process process : processes) {
            System.out.println(process.id + "\t\t" + process.burstTime + "\t\t\t" +
                process.arrivalTime + "\t\t\t" + process.priority); }
    }
 private static void runFCFS(ArrayList<Process> processes) {
 processes.sort(Comparator.comparingInt(p -> p.arrivalTime));
 executeProcesses(processes);
 }
 private static void runSJF(ArrayList<Process> processes) {
 processes.sort(Comparator.comparingInt(p -> p.burstTime));
 executeProcesses(processes);
 }
 private static void runSRTF(ArrayList<Process> processes) {
```

```
 processes.sort(Comparator.comparingInt(p -> p.arrivalTime));
 executeSRTF(processes);
 }
 private static void runPriority(ArrayList<Process> processes) {
 processes.sort(Comparator.comparingInt(p -> p.priority));
 executeProcesses(processes);
 }
 private static void runRoundRobin(ArrayList<Process> processes, int timeQuantum) {
 executeRoundRobin(processes, timeQuantum);
 }
 private static void executeProcesses(ArrayList<Process> processes) {
 int currentTime = 0;
 int totalWaitingTime = 0;
int totalturnaroundTime=0;
 System.out.println("\nProcess\t\tBurst Time\t\tArrival Time\t\tWaiting Time\t\tTurnaround
Time");
 for (Process process : processes) {
 int waitingTime = Math.max(0, currentTime - process.arrivalTime);
 int turnaroundTime = waitingTime + process.burstTime;
 System.out.println(process.id + "\t\t" + process.burstTime + "\t\t\t" +
 process.arrivalTime + "\t\t\t" + waitingTime + "\t\t\t" + turnaroundTime);
 currentTime += process.burstTime;
 totalWaitingTime += waitingTime;
totalturnaroundTime += turnaroundTime;
 }
 double averageWaitingTime = (double) totalWaitingTime / processes.size();
 System.out.println("Average Waiting Time: " + averageWaitingTime);
 double averageturnaroundTime = (double) totalturnaroundTime / processes.size();
 System.out.println("Average Turnaround Time: " + averageturnaroundTime);
 }
 private static void executeSRTF(ArrayList<Process> processes) {
 int currentTime = 0;
 int totalWaitingTime = 0;
int totalTurnaroundTime = 0;
 ArrayList<Process> remainingProcesses = new ArrayList<>(processes);
 System.out.println("\nProcess\t\tBurst Time\t\tArrival Time\t\tWaiting Time\t\tTurnaround
Time");
 while (!remainingProcesses.isEmpty()) {
 Process shortestProcess = remainingProcesses.get(0);
 for (Process process : remainingProcesses) {
 if (process.remainingTime < shortestProcess.remainingTime) {
 shortestProcess = process;
 }
 }
```

```
 int waitingTime = Math.max(0, currentTime - shortestProcess.arrivalTime);
 int turnaroundTime = waitingTime + shortestProcess.burstTime;
 System.out.println(shortestProcess.id + "\t\t" + shortestProcess.burstTime + "\t\t\t" +
 shortestProcess.arrivalTime + "\t\t\t" + waitingTime + "\t\t\t" + turnaroundTime);
 currentTime += shortestProcess.burstTime;
 totalWaitingTime += waitingTime;
totalTurnaroundTime +=turnaroundTime;
 remainingProcesses.remove(shortestProcess);
 }
 double averageWaitingTime = (double) totalWaitingTime / processes.size();
 System.out.println("Average Waiting Time: " + averageWaitingTime);
double averageTurnaroundTime = (double) totalTurnaroundTime / processes.size();
 System.out.println("Average Turnaround Time: " + averageTurnaroundTime);
 }
 private static void executeRoundRobin(ArrayList<Process> processes, int timeQuantum) {
 Queue<Process> queue = new LinkedList<>(processes);
 int currentTime = 0;
 int totalWaitingTime = 0;
int totalTurnaroundTime = 0;
 System.out.println("\nProcess\t\tBurst Time\t\tArrival Time\t\tWaiting Time\t\tTurnaround
Time");
 while (!queue.isEmpty()) {
 Process currentProcess = queue.poll();
 int remainingTime = currentProcess.remainingTime - timeQuantum;
 if (remainingTime <= 0) {
 currentTime += currentProcess.remainingTime;
 int waitingTime = Math.max(0, currentTime - currentProcess.arrivalTime -
currentProcess.burstTime);
 int turnaroundTime = waitingTime + currentProcess.burstTime;
 System.out.println(currentProcess.id + "\t\t" + currentProcess.burstTime + "\t\t\t" +
 currentProcess.arrivalTime + "\t\t\t" + waitingTime + "\t\t\t" + turnaroundTime);
 totalWaitingTime += waitingTime;
totalTurnaroundTime +=turnaroundTime;
 } else {
 currentTime += timeQuantum;
 currentProcess.remainingTime = remainingTime;
 queue.offer(currentProcess);
 }
 }
 double averageWaitingTime = (double) totalWaitingTime / processes.size();
 System.out.println("Average Waiting Time: " + averageWaitingTime);
double averageTurnaroundTime = (double) totalTurnaroundTime / processes.size();
 System.out.println("Average Turnaround Time: " + averageTurnaroundTime);
 }
```

}

OUTPUT:-

```
Enter the number of processes: 4
Enter arrival time for process 1: 0
Enter burst time for process 1: 8
Enter priority for process 1: 1
Enter arrival time for process 2: 0
Enter burst time for process 2: 6
Enter priority for process 2: 2
Enter arrival time for process 3: 2
Enter burst time for process 3: 2
Enter priority for process 3: 3
Enter arrival time for process 4: 3
Enter burst time for process 4: 2
Enter priority for process 4: 0
Choose a scheduling algorithm:
1. FCFS
2. SJF
3. SRTF
4. Priority
5. Round Robin

Enter your Choice:
1
Process Data:
Process     Burst Time          Arrival Time          Priority
1           8                   0                     1
2           6                   0                     2
3           2                   2                     3
4           2                   3                     0

Process     Burst Time          Arrival Time          Waiting Time          Turnaround Time
1           8                   0                     0                      8
2           6                   0                     8                      14
3           2                   2                     12                      14
4           2                   3                     13                      15
Average Waiting Time: 8.25
Average Turnaround Time: 12.75

Do you want to continue (yes/no)? yes
Choose a scheduling algorithm:
1. FCFS
2. SJF
3. SRTF
4. Priority
5. Round Robin
Enter your Choice:
2
Process Data:
Process     Burst Time          Arrival Time          Priority
1           8                   0                     1
2           6                   0                     2
3           2                   2                     3
4           2                   3                     0
```

Process	Burst Time	Arrival Time	Waiting Time	Turnaround Time
3	2	2	0	2
4	2	3	0	2
2	6	0	4	10
1	8	0	10	18

Average Waiting Time: 3.5
Average Turnaround Time: 8.0

Do you want to continue (yes/no)? yes
Choose a scheduling algorithm:
1. FCFS
2. SJF
3. SRTF
4. Priority
5. Round Robin

Enter your Choice:
3
Process Data:

Process	Burst Time	Arrival Time	Priority
1	8	0	1
2	6	0	2
3	2	2	3
4	2	3	0

Process	Burst Time	Arrival Time	Waiting Time	Turnaround Time
3	2	2	0	2
4	2	3	0	2
2	6	0	4	10
1	8	0	10	18

Average Waiting Time: 3.5
Average Turnaround Time: 8.0

Do you want to continue (yes/no)? yes
Choose a scheduling algorithm:
1. FCFS
2. SJF
3. SRTF
4. Priority
5. Round Robin
Enter your Choice:
4
Process Data:

Process	Burst Time	Arrival Time	Priority
1	8	0	1
2	6	0	2
3	2	2	3
4	2	3	0

Process	Burst Time	Arrival Time	Waiting Time	Turnaround Time
4	2	3	0	2
1	8	0	2	10
2	6	0	10	16
3	2	2	14	16

Average Waiting Time: 6.5
Average Turnaround Time: 11.0
Do you want to continue (yes/no)? yes
Choose a scheduling algorithm:
1. FCFS
2. SJF
3. SRTF
4. Priority
5. Round Robin

Enter your Choice:
5
Process Data:

Process	Burst Time	Arrival Time	Priority
1	8	0	1
2	6	0	2
3	2	2	3
4	2	3	0

Enter time quantum for Round Robin: 20

Process	Burst Time	Arrival Time	Waiting Time	Turnaround Time
1	8	0	0	8
2	6	0	8	14
3	2	2	12	14
4	2	3	13	15

Average Waiting Time: 8.25
Average Turnaround Time: 12.75