LP Practical

Write a program to simulate CPU Scheduling Algorithms: FCFS, SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive).

1. FCFS

```
import java.io.*;
import java.util.Scanner;
public class FCFS
{
     public static void main(String args[])
     {
          int i,no_p,burst_time[],TT[],WT[];
          float avg_wait=0,avg_TT=0;
          burst_time=new int[50];
          TT=new int[50];
          WT=new int[50];
          WT[0]=0;
          Scanner s=new Scanner(System.in);
          System.out.println("Enter the number of process: ");
          no_p=s.nextInt();
          System.out.println("\nEnter Burst Time for
processes:");
          for(i=0;i<no_p;i++)
          {
               System.out.print("\tP"+(i+1)+":  ");
               burst_time[i]=s.nextInt();
          }

          for(i=1;i<no_p;i++)
          {
               WT[i]=WT[i-1]+burst_time[i-1];
               avg_wait+=WT[i];
          }
          avg_wait/=no_p;

          for(i=0;i<no_p;i++)
```

```
          {
               TT[i]=WT[i]+burst_time[i];
               avg_TT+=TT[i];
          }
          avg_TT/=no_p;

     System.out.println("\n**********************************
**************************");
          System.out.println("\tProcesses:");

     System.out.println("**********************************
**************************");
          System.out.println("    Process\tBurst Time\tWaiting
Time\tTurn Around Time");
          for(i=0;i<no_p;i++)
          {
               System.out.println("\tP"+(i+1)+"\t
"+burst_time[i]+"\t\t  "+WT[i]+"\t\t "+TT[i]);

          }
          System.out.println("\n-------------------------------
--------------------------------");
          System.out.println("\nAverage waiting time :
"+avg_wait);
          System.out.println("\nAverage Turn Around time :
"+avg_TT+"\n");
     }
}
```

```
/*Output:
Enter the number of process:
3

Enter Burst Time for processes:
     P1:  24
     P2:  3
     P3:  3

*****************************************************************
     Processes:
*****************************************************************
    Process    Burst Time     Waiting Time   Turn Around Time
     P1     24         0         24
     P2     3          24        27
     P3     3          27        30

-----------------------------------------------------------------
Average waiting time : 17.0
Average Turn Around time : 27.0  */
```

2. Round Robin(Preemptive):

```
import java.util.*;
import java.io.*;
class RoundR
{
     public static void main(String args[])
     {
          int Process[]=new int[10];
          int a[]=new int[10];
          int Arrival_time[]=new int[10];
          int Burst_time[]=new int[10];
          int WT[]=new int[10];
          int TAT[]=new int[10];
          int Pno,sum=0;;
          int TimeQuantum;

System.out.println("\nEnter the no. of Process::");
          Scanner sc=new Scanner(System.in);
          Pno=sc.nextInt();
          System.out.println("\nEnter each process::");
          for(int i=0;i<Pno;i++)
          {
               Process[i]=sc.nextInt();
          }

System.out.println("\nEnter the Burst Time of each process::");
          for(int i=0;i<Pno;i++)
          {
               Burst_time[i]=sc.nextInt();
          }
System.out.println("\nEnter the Time Quantum::");
TimeQuantum=sc.nextInt();
          do{
          for(int i=0;i<Pno;i++)
          {
                    if(Burst_time[i]>TimeQuantum)
                    {
                    Burst_time[i]-=TimeQuantum;
                         for(int j=0;j<Pno;j++)
                         {
                              if((j!=i)&&(Burst_time[j]!=0))
```

```
                    WT[j]+=TimeQuantum;
                }
                }
                else
                {
                    for(int j=0;j<Pno;j++)
                    {
                        if((j!=i)&&(Burst_time[j]!=0))
                        WT[j]+=Burst_time[i];
                    }
                    Burst_time[i]=0;
                 }
            }
                    sum=0;
                    for(int k=0;k<Pno;k++)
                    sum=sum+Burst_time[k];
        } while(sum!=0);

                for(int i=0;i<Pno;i++)
                       TAT[i]=WT[i]+a[i];

System.out.println("process\t\tBT\tWT\tTAT");
                    for(int i=0;i<Pno;i++)
                    {

System.out.println("process"+(i+1)+"\t"+a[i]+"\t"+WT[i]+"\t"+TAT
[i]);
                    }
                          float avg_wt=0;
                    float avg_tat=0;
                    for(int j=0;j<Pno;j++)
                    {
                            avg_wt+=WT[j];
                    }
                    for(int j=0;j<Pno;j++)
                    {
                            avg_tat+=TAT[j];
                    }
                      System.out.println("average waiting time
"+(avg_wt/Pno)+"\n Average turn around time"+(avg_tat/Pno));
```

```
     }
}

/*OUTPUT::
unix@unix-HP-280-G1-
MT:~/TEA33$ java RoundR
Enter the no. of Process::
5
Enter each process::
1
2
3
4
5

Enter the Burst Time of each process::
2
1
8
4
5
Enter the Time Quantum::
2
process        BT   WT   TAT
process1  0    0    0
process2  0    2    2
process3  0    12   12
process4  0    9    9
process5  0    13   13
average waiting time 7.2
Average turn around time7.2       */
```

3. SJF(Preemptive):

```
import java.util.Scanner;

class sjf_swap1{
public static void main(String args[])

{
int
burst_time[],process[],waiting_time[],tat[],arr_time[],completio
n_time[],i,j,n,total=0,total_comp=0,pos,temp;
float wait_avg,TAT_avg;
Scanner s = new Scanner(System.in);
 System.out.print("Enter number of process: ");
n = s.nextInt();
 process = new int[n];
burst_time = new int[n];
waiting_time = new int[n];
arr_time=new int[n];
tat = new int[n];
completion_time=new int[n];

//burst time
System.out.println("\nEnter Burst time:");
for(i=0;i<n;i++)
{
System.out.print("\nProcess["+(i+1)+"]: ");
burst_time[i] = s.nextInt();;
process[i]=i+1; //Process Number
}

//arrival time
System.out.println("\nEnter arrival time:");
for(i=0;i<n;i++)
{
System.out.print("\nProcess["+(i+1)+"]: ");
arr_time[i] = s.nextInt();;
process[i]=i+1; //Process Number
}

//Sorting
for(i=0;i<n;i++)
```

```
{
pos=i;
for(j=i+1;j<n;j++)
{
if(burst_time[j]<burst_time[pos])
pos=j;
}

temp=burst_time[i];
burst_time[i]=burst_time[pos];
burst_time[pos]=temp;

temp=process[i];
process[i]=process[pos];
process[pos]=temp;

System.out.println("process"+process[i]);
}
//completion
time new
for(i=1;i<n;i++)
{
completion_time[i]=0;
for(j=0;j<i;j++)
completion_time[i]+=burst_time[j];
 total_comp+=completion_time[i];
}

//First process has 0 waiting
time
waiting_time[0]=0;
//calculate

waiting time
for(i=1;i<n;i++)
{
waiting_time[i]=0;
for(j=0;j<i;j++)
waiting_time[i]+=burst_time[j];
total+=waiting_time[i];
}
```

```
//Calculating Average waiting time
wait_avg=(float)total/n;
total=0;

System.out.println("\nPro_number\t Burst Time
\tcompletion_time\tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
tat[i]=burst_time[i]+waiting_time[i];
 //Calculating Turnaround Time
total+=tat[i];
System.out.println("\n"+process[i]+"\t\t "+burst_time[i]+"\t\t
"+completion_time[i]+"\t\t"+waiting_time[i]+"\t\t "+tat[i]);
}

//Calculation of Average Turnaround Time
TAT_avg=(float)total/n;
System.out.println("\n\nAWT: "+wait_avg);
System.out.println("\nATAT: "+TAT_avg);

}
}
```

4. Priority (Non-Preemptive):

```
import java.util.*;

class Process {
	int at, bt, pri, pno;
	Process(int pno, int at, int bt, int pri)
	{
		this.pno = pno;
		this.pri = pri;
		this.at = at;
		this.bt = bt;
	}
}

class GChart {
	// process number, start time, complete time,
	// turn around time, waiting time
	int pno, stime, ctime, wtime, ttime;
}

class MyComparator implements Comparator {

	public int compare(Object o1, Object o2)
	{

		Process p1 = (Process)o1;
		Process p2 = (Process)o2;
		if (p1.at < p2.at)
			return (-1);

		else if (p1.at == p2.at && p1.pri > p2.pri)
			return (-1);

		else
			return (1);
	}
}
class FindGantChart {
```

```
	void findGc(LinkedList queue)
	{

		int time = 0;
		TreeSet prique = new TreeSet(new MyComparator());
		LinkedList result = new LinkedList();
		while (queue.size() > 0)
			prique.add((Process)queue.removeFirst());

		Iterator it = prique.iterator();
		time = ((Process)prique.first()).at;
		while (it.hasNext()) {
			Process obj = (Process)it.next();

			GChart gc1 = new GChart();
			gc1.pno = obj.pno;
			gc1.stime = time;
			time += obj.bt;
			gc1.ctime = time;
			gc1.ttime = gc1.ctime - obj.at;
			gc1.wtime = gc1.ttime - obj.bt;
			result.add(gc1);
		}
		new ResultOutput(result);
	}
}
```

Output:

Process_no	Start_time	Complete_time	Turn_Around_Time	Waiting_Time
1	1	4	3	0
2	5	10	8	3
3	4	5	2	1
4	10	17	13	6
5	17	21	16	12

Average Waiting Time is : 4.4

Average Turn Around time is : 8.4