**CODE:-**

import java.util.\*;

class Process {

int processID;

int arrival, burst, waiting, turnAround, remainingTime;

int finish,completionTime;

}

public class RRScheduling {

public static void main(String[] args) {

int n,sumBurst=0,quantum,time;

double avgWAT=0,avgTAT=0;

Scanner sc=new Scanner(System.in);

Queue<Integer> q = new LinkedList<>();

System.out.println("\*\*\* RR Scheduling (Preemptive) \*\*\*");

System.out.print("Enter Number of Process: ");

n=sc.nextInt();

Process[] p = new Process[n];

for (int i = 0; i < n; i++) {

p[i] = new Process();

p[i].processID = i + 1;

System.out.print("Enter the arrival time for P" + (i + 1) + ": ");

p[i].arrival = sc.nextInt();

System.out.print("Enter the burst time for P" + (i + 1) + ": ");

p[i].burst = sc.nextInt();

p[i].remainingTime = p[i].burst;

p[i].finish = 0;

sumBurst += p[i].burst;

System.out.println();

}

System.out.print("\nEnter time quantum: ");

quantum = sc.nextInt();

Process pTemp;

for (int i = 0; i < n - 1; i++) {

for (int j = i + 1; j < n; j++) {

if (p[i].arrival > p[j].arrival) {

pTemp = p[i];

p[i] = p[j];

p[j] = pTemp;

}

}

}

q.add(0);

for (time = p[0].arrival; time < sumBurst;) {

Integer I = q.remove();

int i = I.intValue();

if (p[i].remainingTime <= quantum) {

time += p[i].remainingTime;

p[i].remainingTime = 0;

p[i].finish = 1;

p[i].completionTime=time;

p[i].waiting = time - p[i].arrival - p[i].burst;

p[i].turnAround = time - p[i].arrival;

for (int j = 0; j < n; j++) {

Integer J = Integer.valueOf(j);

if ((p[j].arrival <= time) && (p[j].finish != 1) && (!q.contains(J)))

q.add(j);

}

} else {

time += quantum;

p[i].remainingTime -= quantum;

for (int j = 0; j < n; j++) {

Integer J = Integer.valueOf(j);

if (p[j].arrival <= time && p[j].finish != 1 && i != j && (!q.contains(J)))

q.add(j);

}

q.add(i);

}

}

System.out.println("\n\*\*\* RR Scheduling (Preemptive) \*\*\*");

System.out.println("Processor\tArrival time\tBrust time\tCompletion Time\t\tTurn around time\tWaiting time");

System.out.println("----------------------------------------------------------------------------------------------------------");

for (int i = 0; i < n; i++) {

System.out.println(("P"+(i+1))+"\t\t"+p[i].arrival+"ms\t\t"+p[i].burst+"ms\t\t"+p[i].completionTime+"ms\t\t\t"+p[i].turnAround+"ms\t\t\t"+p[i].waiting+"ms");

avgWAT += p[i].waiting;

avgTAT += p[i].turnAround;

}

System.out.println("\nAverage turn around time of processor: "+(avgTAT/n)+"ms\nAverage waiting time of processor: "+(avgWAT/n)+"ms");

}

}

**OUTPUT: -**

\*\*\* RR Scheduling (Preemptive) \*\*\*

Enter Number of Process: 6

Enter the arrival time for P1: 0

Enter the burst time for P1: 7

Enter the arrival time for P2: 1

Enter the burst time for P2: 4

Enter the arrival time for P3: 2

Enter the burst time for P3: 15

Enter the arrival time for P4: 3

Enter the burst time for P4: 11

Enter the arrival time for P5: 4

Enter the burst time for P5: 20

Enter the arrival time for P6: 4

Enter the burst time for P6: 9

Enter time quantum: 5

\*\*\* RR Scheduling (Preemptive) \*\*\*

Processor Arrival time Brust time Completion Time Turn around time Waiting time

----------------------------------------------------------------------------------------------------------

P1 0ms 7ms 31ms 31ms 24ms

P2 1ms 4ms 9ms 8ms 4ms

P3 2ms 15ms 55ms 53ms 38ms

P4 3ms 11ms 56ms 53ms 42ms

P5 4ms 20ms 66ms 62ms 42ms

P6 4ms 9ms 50ms 46ms 37ms

Average turn around time of processor: 42.166666666666664ms

Average waiting time of processor: 31.166666666666668ms