**Group : B**

**Assignment No. : 4.2**

**Title : Write a program to implement Round-Robin.**

**Roll No. : 2365**

**-------------------------------------------------------------------------------------------**

import java.util.Scanner;

class CQueue

{

int front,rear,array[],size;

CQueue(int n)

{

front=0;

rear=0;

array=new int[n];

size=n;

}

void enqueue(int pnm)

{

if((rear == size && front == 0) || (rear==front) && front!=0)

System.out.println("Queue is full!!");

else

{

if (rear == size&& front != 0)

rear=0;

array[rear]=pnm;

rear++;

}

}

int dequeue()

{

if(front==0 && rear==0)

return 0;

else

{

if(front==size)

front=0;

int d=array[front];

array[front]=0;

front++;

return d;

}

}

}

class ProcessP

{

double at,bt,ft,tt,wt;

ProcessP()

{

at=bt=tt=wt=ft=0;

}

public void accept(double b)

{

bt=b;

}

public void display(int i)

{

System.out.println("P"+i+"\t"+at+"\t"+bt+"\t"+ft+"\t"+tt+"\t"+wt);

}

public void calculate(int n,ProcessP P[])

{

tt=ft-at;

wt=tt-bt;

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

{

P[i].tt=P[i].ft-P[i].at;

P[i].wt=P[i].tt-P[i].bt;

}

}

}

public class RoundRobin

{

public static void main(String[] args)

{

int rr\_t[],q;

double rt[],sumT=0,sumW=0,att=0,awt=0;

Scanner sc =new Scanner(System.in);

rr\_t=new int[100];

double b;

System.out.println("Enter number of processes:");

int n=sc.nextInt();

System.out.println("Enter time quantum:");

q=sc.nextInt();

rt=new double[n];

CQueue CQ=new CQueue(n);

ProcessP P[]=new ProcessP[n];

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

{

//System.out.println("Enter the arrival time of the process:");

//A=sc.nextDouble();

System.out.println("Enter the burst time of the process:");

b=sc.nextDouble();

P[i]=new ProcessP();

P[i].accept(b);

rt[i]=b;

CQ.enqueue(i);

}

int nm,j,cnt=0,cur=0;

do

{

j=0;

nm=CQ.dequeue();

if(rt[nm]>q)

{

while(j<q)

{

rr\_t[cur]=nm;

cur++;

j++;

}

rt[nm]=rt[nm]-q;

CQ.enqueue(nm);

}

else

{

j=0;

while(j<rt[nm])

{

rr\_t[cur]=nm;

cur++;

j++;

}

rt[nm]=0;

P[nm].ft=cur;

cnt++;

}

}while(cnt!=n);

P[0].calculate(n, P);

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

{

sumT+=P[i].tt;

sumW+=P[i].wt;

}

att=sumT/n;

awt=sumW/n;

System.out.println("Process\tAT\tBT\tFT\tTT\tWT");

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

{

P[i].display(i+1);

}

System.out.println("Average Turnaround time is:"+att);

System.out.println("Average Waiting time is:"+awt+"\n");

System.out.println("Gantt Chart :");

for(int i=0;i<cur;i=i+2)

{

System.out.print("P"+(rr\_t[i]+1)+" |");

}

sc.close();

}

}

/\*

Enter number of processes:

4

Enter time quantum:

2

Enter the burst time of the process:

6

Enter the burst time of the process:

5

Enter the burst time of the process:

8

Enter the burst time of the process:

4

Queue is full!!

Process AT BT FT TT WT

P1 0.0 6.0 21.0 21.0 15.0

P2 0.0 5.0 19.0 19.0 14.0

P3 0.0 8.0 0.0 0.0 -8.0

P4 0.0 4.0 16.0 16.0 12.0

Average Turnaround time is:14.0

Average Waiting time is:8.25

Gantt Chart :

P1 |P2 |P3 |P4 |P1 |P2 |P3 |P4 |P1 |P2 |P3 |

\*/