**SIMULATION BASED ASSIGNMENT**

**Name :** CVS Vishwanath

**Regd. No :** 11500991

**Email :** chavali.vishwanath999@gmail.com

**Github Link :** https://github.com/cvsvishwanath/sunrisers/

**PROGRAM 6 :**

#include<stdio.h>

#include<conio.h>

int main()

{

int ch,count,num,time,remain,flag=0,tq=4,wt=0,tat=0,at[10],bt[10],rt[10],bt1[20],p[20],wt1[20],tat1[20],pr[20],c,d,e,t1=0,t2=0,pos,temp,avg\_wt,avg\_tat,f,g,h,bt2[20],wt2[20],tat2[20],avwt=0,avtat=0;

printf("Press Y/y to continue : ");

while(getch()=='Y' || getch()=='y')

{

printf("\nEnter your choice : 1.Round Robin\t2.Priority scheduling\t3.FCFS : ");

scanf("%d",&ch);

printf("Enter total Processes : ");

scanf("%d",&num);

switch(ch) {

case 1: remain=num;

for(count=0;count<num;count++)

{

printf("Enter Arrival Time and Burst Time for Process %d : ",count+1);

scanf("%d",&at[count]);

scanf("%d",&bt[count]);

rt[count]=bt[count];

}

printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");

for(time=0,count=0;remain!=0;)

{

if(rt[count]<=tq && rt[count]>0)

{

time+=rt[count];

rt[count]=0;

flag=1;

}

else if(rt[count]>0)

{

rt[count]-=tq;

time+=tq;

}

if(rt[count]==0 && flag==1)

{

remain--;

printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);

wt+=time-at[count]-bt[count];

tat+=time-at[count];

flag=0;

}

if(count==num-1)

count=0;

else if(at[count+1]<=time)

count++;

else

count=0;

}

printf("\nAverage Waiting Time= %f\n",wt\*1.0/num);

printf("Avg Turnaround Time = %f",tat\*1.0/num);

break;

case 2: for(c=0;c<num;c++)

{

printf("\nEnter Burst Time and Priority for Process %d : ",c+1);

scanf("%d %d",&bt1[c],&pr[c]);

p[c]=c+1; //contains process number

}

for(c=0;c<num;c++)

{

pos=c;

for(d=c+1;d<num;d++)

{

if(pr[d]<pr[pos])

pos=d;

}

temp=pr[c];

pr[c]=pr[pos];

pr[pos]=temp;

temp=bt1[c];

bt1[c]=bt1[pos];

bt1[pos]=temp;

temp=p[c];

p[c]=p[pos];

p[pos]=temp;

}

wt1[0]=0;

for(c=1;c<num;c++)

{

wt1[c]=0;

for(d=0;d<c;d++)

wt1[c]+=bt1[d];

t1+=wt1[c];

}

printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");

for(c=0;c<num;c++)

{

tat1[c]=bt1[c]+wt1[c]; //calculate turnaround time

t2+=tat1[c];

printf("\nP[%d]\t\t %d\t\t %d\t\t\t%d",p[c],bt1[c],wt1[c],tat1[c]);

}

printf("\n\nAverage Waiting Time = %f\n",t1\*1.0/num);

printf("Average Turnaround Time = %f",t2\*1.0/num);

break;

case 3: for(f=0;f<num;f++)

{

printf("Enter Burst Time for Process %d : ",f+1);

scanf("%d",&bt2[f]);

}

wt2[0]=0;

for(f=1;f<num;f++)

{

wt2[f]=0;

for(g=0;g<f;g++)

wt2[f]+=bt2[g];

}

printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");

for(f=0;f<num;f++)

{

tat2[f]=bt2[f]+wt2[f];

avwt+=wt2[f];

avtat+=tat2[f];

printf("\nP[%d]\t\t%d\t\t%d\t\t%d",f+1,bt2[f],wt2[f],tat2[f]);

}

printf("\n\nAverage Waiting Time : %f\n",avwt\*1.0/num);

printf("Average Turnaround Time : %f",avtat\*1.0/num);

break;

default: printf("Wrong choice !!!");

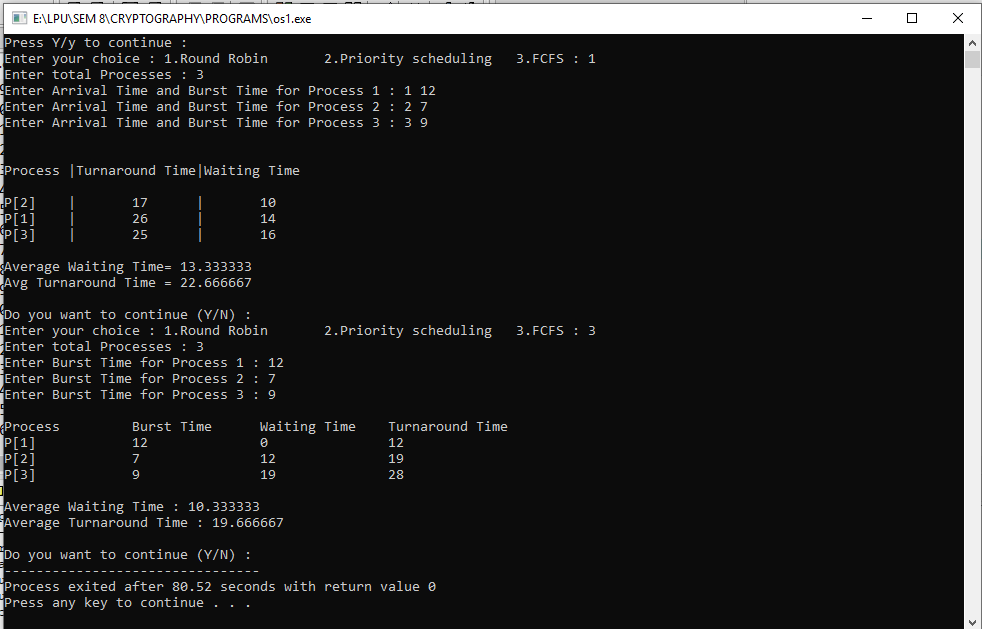
}

printf("\n\nDo you want to continue (Y/N) : ");

}

return 0;

}

**OUTPUT :**

1. Explain the problem in terms of operating system concept? (Max 200 word)

The given problem statement is regarding a multi level queue in which there are 3 queues to be implemented : Round Robin, Priority & FCFS scheduling. CPU keeps on shifting between processes with each process having 10 units of time. Priority is associated with every queue. Quantum time for Round Robin is 4 units.

2. Write the algorithm for proposed solution of the assigned problem.

* Prompt user to select for any Scheduling algorithm using Menu options
* Prompt user to enter the number of processes
* Prompt user to enter arrival time and burst time for each process
* Under switch case, execute Waiting time and Turnaround time for each scheduling algorithm :

(i) Create an array rem[] to keep track of remaining burst time of processes. This array is initially a copy of bt[] (burst times array)

(ii) Create another array wt[] to store waiting times of processes. Initialize this array as 0.

(iii) Initialize time : t = 0

(iv) Keep traversing the all processes while all processes are not done. Do following for i'th process if it is not done yet.

a- If rem\_bt[i] > quantum

(i) t = t + quantum

(ii) bt\_rem[i] -= quantum;

c- Else // Last cycle for this process

(i) t = t + bt\_rem[i];

(ii) wt[i] = t - bt[i]

(ii) bt\_rem[i] = 0;

* Prompt user to continue or exit
* Return and Exit

3. Calculate complexity of implemented algorithm. (Student must specify complexity of each line of code along with overall complexity)

Complexity for Switch case 1 : O(1)

Complexity for Switch case 2 : O(n)

Complexity for Switch case 3 : O(1)

Overall Complexity : O(1) + O(n) + O(1) = O(n)

4. Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint.

Constraint 1 : There should be 3 queues generated.

Constraint 2 : CPU must remain for 10 units of time for each queue.

Constraint 3 : CPU must shift to other queues after 10 units of time on the basis of priority.

Constraint 4 : Time quantum for Round Robin scheduling is 4 units.

5. Explain the boundary conditions of the implemented code.

In Round Robin scheduling, at the beginning before any job is released and at the end after all jobs are completed, Potential value Φ = 0. FCFS can also block the system in a busy dynamic system in another way, known as the ***convoy effect***. When one CPU intensive process blocks the CPU, a number of I/O intensive processes can get backed up behind it, leaving the I/O devices idle. When the CPU hog finally relinquishes the CPU, then the I/O processes pass through the CPU quickly, leaving the CPU idle while everyone queues up for I/O, and then the cycle repeats itself when the CPU intensive process gets back to the ready queue. Priority queue may suffer from indefinite blocking or starvation.

6.Explain all the test cases applied on the solution of assigned problem**.**