NAME : T Vijay

REG NO : 11701013

EMAIL ID : [vijay.tvr198@gmail.com](mailto:vijay.tvr198@gmail.com)

***Description:***

* This is a scheduling program to implement a Queue with two levels:
* Level 1 : Fixed priority preemptive Scheduling
* Level 2 : Round Robin Scheduling
* For a Fixed priority pre-emptive scheduling if one process P1 is scheduled and running and another process P2 with higher priority comes. The New process with high priority process P2 preempts currently running process P1 and process P1 will go to second level queue. Time for which process will strictly execute must be considered in themultiplesof 2.
* All theprocesses in second level queue will complete their execution according to round robin scheduling.
* In this program Queue 2 will be processed after Queue 1 becomes empty and Priority of Queue 2 has lower priority than in Queue 1.

**Code Snippet :**

#include<stdio.h>

struct process

{

int pro\_name;

int arrT, waiturnT, turnT, burT,priority, burTcopy;

}queue1[10],queue2[10];

void main()

{

struct process temp;

int i,time=0,t1,t2,bu\_t=0,largest,totalProcess,count=0,k,pf2=0,totalProcess2,n,pos,j,flag=0,y;

float wait\_time=0,turnaround\_time= 0,average\_waiting\_time,average\_turnaround\_time;

printf("\n Enter Total Number of Processes:\t");

scanf("%d", &totalProcess);

n=totalProcess;

for(i=0;i<totalProcess;i++)

{

printf("\nEnter Process name:-");

//fflush(stdin);

scanf("%d",&queue1[i].pro\_name);

printf("\nEnter Details For processor %d:\n",queue1[i].pro\_name);

printf("Enter Arrival Time:-");

scanf("%d",&queue1[i].arrT);

printf("Enter Burst Time:-");

scanf("%d",&queue1[i].burT);

queue1[i].burTcopy=queue1[i].burT;

printf("Enter Priority:\t");

scanf("%d",&queue1[i].priority);

}

printf("\nEnter Time Quantum for Fixed priority queue:-");

scanf("%d",&t1);

printf("\nEnter Time Quantum for Round Robin queue:-");

scanf("%d",&t2);

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

for(i=0;i<totalProcess;i++)

{

pos=i;

for(j=i+1;j<totalProcess;j++)

{

if(queue1[j].arrT<queue1[pos].arrT)

pos=j;

}

temp=queue1[i];

queue1[i]=queue1[pos];

queue1[pos]=temp;

}

time=queue1[0].arrT;

for(i=0;totalProcess!=0;i++)

{

while(count!=t1)

{

count++;

if(queue1[i].arrT<=time)

{

for(j=i+1;j<totalProcess;j++)

{

if(queue1[j].arrT==time && queue1[j].priority<queue1[i].priority)//pr<

{

queue2[pf2]=queue1[i];

pf2++;

for(k=i; k<totalProcess-1;k++)

queue1[k]=queue1[k+1];

totalProcess--;

count=0;

i=j-1;

j--;

}

}

}

time++;

queue1[i].burT--;

if(queue1[i].burT==0)

{

queue1[i].turnT=time-queue1[i].arrT;

queue1[i].waiturnT=queue1[i].turnT-queue1[i].burTcopy;

printf("%d\t|\t%d\t|\t%d\n",queue1[i].pro\_name,queue1[i].turnT,queue1[i].waiturnT);

wait\_time+=time-queue1[i].waiturnT;

turnaround\_time+=time-queue1[i].turnT;

for(k=i;k<totalProcess-1;k++)

queue1[k]=queue1[k+1];i--;

totalProcess--;

count=t1;break;

}

}

count=0;

if(queue1[i].burT!=0)

{

queue2[pf2]=queue1[i];

pf2++;

for(k=i;k<totalProcess-1;k++)

queue1[k]=queue1[k+1];

totalProcess--;

}

if(i==totalProcess-1)

i=-1;

}

totalProcess2=pf2;

for(count=0;totalProcess2!=0;)

{

if(queue2[count].burT<=t2&&queue2[count].burT>0)

{

time+=queue2[count].burT;

queue2[count].burT=0;

flag=1;

}

else if(queue2[count].burT>0)

{

queue2[count].burT-=t2;

time+=t2;

}

if(queue2[count].burT==0&&flag==1)

{

totalProcess2--;

queue2[count].turnT=time-queue2[count].arrT;

queue2[count].waiturnT=queue2[count].turnT-queue2[count].burTcopy;

printf("%d\t|\t%d\t|\t%d\n",queue2[count].pro\_name,queue2[count].turnT,queue2[count].waiturnT);

turnaround\_time+=time-queue2[count].arrT;

wait\_time+=time-queue2[count].arrT-queue2[count].burTcopy;

for(k=count; k<totalProcess2;k++)

queue2[k]=queue2[k+1];count--;

flag=0;

}

if(count==totalProcess2-1)

count=0;

else

count++;

}

printf("\n Average Waiting Time= %f\n", wait\_time/n);

printf("Avg Turnaround Time = %f" ,turnaround\_time/n);

}

#include<stdio.h>

struct process

{

int pro\_name;

int arrT , waiturnT , burT turnT, priority, burTcopy;

}

queue1[10],queue2[10];

int main()

{

struct process temp;

int i,time=0,t1,t2,bu\_t=0,largest,totalProcess, count =0,k,pf2=0,totalProcess2,n,pos,j, flag =0,y;

float wait\_time=0,turnaround\_time=0,average\_waiting\_time,average\_turnaround\_time;

printf("\n Enter Total Number of Processes:\t");

scanf("%d", &totalProcess);

n=totalProcess;

for (i=0;i<totalProcess;i ++ )

{

printf("\nEnter Process name:-");

//fflush (stdin);

scanf("%d",&queue1[i].pro\_name);

printf("\nEnter Details For processor %d:\n",queue1[i].pro\_name);

printf(" Enter Arrival Time :-");

scanf("%d",&queue1[i]. arrT );

printf(" Enter Burst Time :-");

scanf("%d",&queue1[i].burTcopy);

queue1[i].burTcopy=queue1[i].burTcopy;

printf(" Enter Priority:\t");

scanf("%d",&queue1[i].priority);

}

printf("\nEnter Time Quantum for Fixed priority queue:-");

scanf("%d",&t1);

printf("\nEnter Time Quantum for Round Robin queue:-");

scanf("%d",&t2);

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

for (i=0;i<totalProcess;i ++ )

{

pos=i;

for (j=i+1;j<totalProcess;j ++ )

{

if(queue1[j]. arrT <queue1[pos]. arrT )

pos=j;

}

temp=queue1[i];

queue1[i]=queue1[pos];

queue1[pos]=temp;

}

time=queue1[0]. arrT ;

for (i=0;totalProcess!=0;i ++ )

{

while ( count !=t1)

{

count ++ ;

if(queue1[i]. arrT <= time)

{

for (j=i+1;j<totalProcess;j ++ )

{

if(queue1[j]. arrT == time &&

queue1[j].priority<queue1[i].priority)//pr<

{

queue2[pf2]=queue1[i];

pf2 ++ ;

for (k=i; k<totalProcess-1;k ++ )

{

queue1[k]=queue1[k+1];

totalProcess--;

count =0;

i=j-1;

j--;

}

}

}

time ++ ;

queue1[i].burTcopy--;

if(queue1[i].burTcopy == 0)

{

queue1[i].turnT=time-queue1[i]. arrT ;

queue1[i]. waiturnT =queue1[i].turnT-queue1[i].burTcopy;

printf("%d\t t%d\t t%d\n",queue1[i].pro\_name,queue1[i].turnT,queue1[i]. waiturnT );

wait\_time += time-queue1[i]. waiturnT ;

turnaround\_time += time-queue1[i].turnT;

for (k=i;k<totalProcess-1;k ++ )

{

queue1[k]=queue1[k+1];i--;

totalProcess--;

count =t1;break;

}

}

}

count =0;

if(queue1[i].burTcopy!=0)

{

queue2[pf2]=queue1[i];

pf2 ++ ;

for (k=i;k<totalProcess-1;k ++ )

{

queue1[k]=queue1[k+1];

totalProcess--;

}

if(i == totalProcess-1)

i=-1;

}

totalProcess2=pf2;

for ( count =0;totalProcess2!=0;)

{

if(queue2[ count ].burTcopy <= t2 && queue2[ count ].burTcopy>0)

{

time += queue2[ count ].burTcopy;

queue2[ count ].burTcopy=0;

flag =1;

}

else if(queue2[ count ].burTcopy>0)

{

queue2[ count ].burTcopy-=t2;

time += t2;

}

if(queue2[ count ].burTcopy == 0 && flag == 1)

{

totalProcess2--;

queue2[ count ].turnT=time-queue2[ count ]. arrT ;

queue2[ count ]. waiturnT =queue2[ count ].turnT-

queue2[ count ].burTcopy;

printf("%d\t t%d\t t%d\n",queue2[ count ].pro\_name,queue2[count].turnT,queue2[count].waiturnT);

turnaround\_time += time-queue2[ count ]. arrT ;

wait\_time += time-queue2[ count ]. arrT -queue2[ count ].burTcopy;

for (k= count ; k<totalProcess2;k ++ )

{

queue2[k]=queue2[k+1]; count --;

flag =0;

}

if( count == totalProcess2-1)

count =0;

else

count ++ ;

}

}

}

}

printf("\n Average Waiting Time = %f\n", wait\_time/n);

printf(" Avg Turnaround Time = %f" ,turnaround\_time/n);

}

TEST CASE

