/\* Question 4 code

(i) Priority preemptive Scheduling

(ii) Round Robin Scheduling \*/

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int n,i,j,k,q,pos,total\_turnaround\_time,total\_waiting\_time,temp,totaltime=0,total=0;

int p[50],priority[50],burst\_time[50],execution\_time[50],newburst\_time[50],newpriority[50],newexecution\_time[50],newp[50],total\_execution\_time[50],at[50],t[50],tt[50],waiting\_time[50],rrg[99],tat[50];

cout<<"Enter Total Number of Process:"<<endl;

cin>>n; //number of proccesses

cout<<"Enter Burst Time, Execution Time and Priority\n"<<endl;

for(i=0;i<n;i++) //details of the process

{

cout<<"Process ["<<i+1<<"]"<<endl;;

cout<<"Burst Time: ";

cin>>burst\_time[i];

cout<<"Execution time: ";

cin>>execution\_time[i];

total\_execution\_time[i]=execution\_time[i];

at[i]=t[i]=tt[i]=waiting\_time[i]=0;

cout<<"Priority: ";

cin>>priority[i];

p[i]=i+1; //position of the process

}

for(i=0;i<n;i++) //sorting in ascending using insertion sort

{

pos=i;

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

{

if(priority[j]<priority[pos])

pos=j;

}

temp=priority[i];

priority[i]=priority[pos];

priority[pos]=temp;

temp=burst\_time[i];

burst\_time[i]=burst\_time[pos];

burst\_time[pos]=temp;

temp=execution\_time[i];

execution\_time[i]=execution\_time[pos];

execution\_time[pos]=temp;

temp=p[i];

p[i]=p[pos];

p[pos]=temp;

}

cout<<endl<<"Adding a new process to the queue"<<endl; //new process with higher priority added into the queue

cout<<"Enter the new Burst time, Execution Time and Priority"<<endl;

cout<<"Process ["<<n+1<<"]"<<endl;

cout<<"Burst time: ";

cin>>burst\_time[n];

cout<<"Execution Time: ";

cin>>execution\_time[n];

total\_execution\_time[i]=execution\_time[i];

at[i]=t[i]=tt[i]=waiting\_time[i]=0;

cout<<"Priority: ";

cin>>priority[n];

p[n]=n+1;

if(priority[0]>priority[n]) //comparing if the first process has higher priority then the new process

{

temp=priority[n];

priority[n]=priority[0];

priority[0]=temp;

temp=burst\_time[n];

burst\_time[n]=burst\_time[0];

burst\_time[0]=temp;

temp=execution\_time[n];

execution\_time[n]=execution\_time[0];

execution\_time[0]=temp;

temp=p[n];

p[n]=p[0];

p[0]=temp;

}

j=1;

for(i=0;i<n;i++) //transfering the process into second queue to be solved with Round robin scheduling

{

newburst\_time[i]=burst\_time[i+j];

newpriority[i]=priority[i+j];

newexecution\_time[i]=execution\_time[i+j];

newp[i]=p[i+j];

}

for(i=0;i<n;i++) //arranding in ascending order

{

pos=i;

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

{

if(newpriority[j]<newpriority[pos])

pos=j;

}

temp=newpriority[i];

newpriority[i]=newpriority[pos];

newpriority[pos]=temp;

temp=newburst\_time[i];

newburst\_time[i]=newburst\_time[pos];

newburst\_time[pos]=temp;

temp=newexecution\_time[i];

newexecution\_time[i]=newexecution\_time[pos];

newexecution\_time[pos]=temp;

temp=newp[i];

newp[i]=newp[pos];

newp[pos]=temp;

}

cout<<endl<<"Added the lower Priorities to 2nd queue"<<endl; //First queue execution

cout<<"Execution of the first queue"<<endl;

cout<<"\nProcess\t Burst Time \tWaiting Time\tTurnaround Time";

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

{

tat[i]=burst\_time[i]+waiting\_time[i];

cout<<"\nP["<<p[i]<<"]\t\t "<<burst\_time[i]<<"\t\t "<<waiting\_time[i]<<"\t\t\t"<<tat[i];

}

cout<<endl<<"Enter Time Quantum:"<<endl; //Quantum time

cin>>q;

waiting\_time[0]=0;

for(i=1;i<=n;i++) //Calculating wating time and total waiting time

{

waiting\_time[i]=0;

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

waiting\_time[i]+=newburst\_time[j];

total+=waiting\_time[i];

}

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

{

totaltime+=newexecution\_time[i];

}

i=0;

k=0;

for(j=0;j<totaltime;j++) //round robin scheduling

{

if((k==0)&&(newexecution\_time[i]!=0))

{

waiting\_time[i]=j;

if((t[i]!=0))

{

waiting\_time[i]-=q\*t[i];

}

}

if((newexecution\_time[i]!=0)&&(k!=q))

{

rrg[j]=newp[i];

newexecution\_time[i]-=1;

k++;

}

else

{

if((k==q)&&(newexecution\_time[i]!=0))

{

t[i]+=1;

}

i=i+1;

if(i==n)

{

i=0;

}

k=0;

j=j-1;

}

}

total\_waiting\_time=0;

total\_turnaround\_time=0;

cout<<"\n Result Of Round Robin";

cout<<"\n Process No\tExecution Time\tWaiting Time\tTurnaround Time";

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

{

tt[i]=waiting\_time[i]+total\_execution\_time[i];

total\_turnaround\_time+=tt[i];

total\_waiting\_time+=waiting\_time[i];

cout<<"\n "<<i+1<<"\t\t"<<total\_execution\_time[i]<<"\t\t"<<waiting\_time[i]<<"\t\t"<<tt[i];

}

cout<<"\n Average Waiting Time:"<<(float)total\_waiting\_time/n;

cout<<"\n Average Turn Around Time:"<<(float)total\_turnaround\_time/n;

return 0;

}