

ASSIGNMENT #2

*WAQAS ASHIQ.*

*BCS-F11-201.*

*SECTION C.*

*SUBMITTED TO: Dr.Adnan Ahmed*

***Code:***

#include<iostream>

#include<cstdlib>

#include<queue>

#include<conio.h>

#include<cstdio>

#include<stdio.h>

using namespace std;

typedef struct process

{

int id,at,bt,st,ft,pr;

float wt,tat,awt;

}process;

process p[10],p1[10],temp;

queue<int> q1;

int accept();

void sjsort(int n);

void ganttsj(int n);

void ganttps(int n);

float turnwait(int n);

void display(int n);

void ganttrr(int n);

void ganttfcfs(int n);

int main()

{

int i,n,ts,ch,j,x;

float awtt[4];

p[0].tat=0;

p[0].wt=0;

printf("Finding best Scheduling Algorithm");

n=accept();

ganttfcfs(n);

awtt[0] = turnwait(n);

display(n);

sjsort(n);

ganttsj(n);

awtt[1] = turnwait(n);

display(n);

ganttps(n);

awtt[2]=turnwait(n);

display(n);

ganttrr(n);

awtt[3] = turnwait(n);

display(n);

int minimum = awtt[0];

int location = 0;

for (int i = 1 ; i < 4 ; i++ )

{

if ( awtt[i] < minimum )

{

minimum = awtt[i];

location = i+1;

}

}

switch(location){

case 0:

printf("\nThe most efficient is FIRST COME FIRST SCHEDULING: ");

printf("%f",awtt[0]);

break;

case 1:

printf("\nThe most efficient is SHORT JOB FIRST SCHEDULING: ");

printf(" %f",awtt[1]);

break;

case 2:

printf("\nThe most efficient is PRIORITY SCHEDULING: ");

printf("\n%f",awtt[2]);

break;

case 3:

printf("\nThe most efficient is ROUND ROBIN SCHEDULING: ");

printf("\n%f",awtt[3]);

break;

}

getch();

return 0;

}

int accept() //functions for accepting inputs from the user

{

int i,n;

printf("\nEnter the no. of process: ");

scanf("%d",&n);

if(n==0)

{

printf("\nInvalid no. of process");

exit(1);

}

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

{

printf("\nEnter the arrival time for process P%d: ",i);

scanf("%d",&p[i].at);

p[i].id=i;

}

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

{

printf("\nEnter the burst time for process P%d: ",i);

scanf("%d",&p[i].bt);

}

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

{

printf("\nEnter the priority for P%d: ",i);

scanf("%d",&p[i].pr);

}

printf("\nInputs given by the user are:\n");

printf("==============================================================");

printf("\nProcess\tAT\tBT");

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

printf("\nP%d\t%d\t%d",p[i].id,p[i].at,p[i].bt);

printf("\n============================================================");

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

p1[i]=p[i];

return n;

}

/\*FUNCTION FOR DISPLAYING GANTT CHART FOR FIRST COME FIRST SERVE SCHEDULING\*/

void ganttfcfs(int n)

{

printf("\nHere it is gant chart for fcfs");

int i;

p[1].st=p[1].at;

for(i=2;i<=n;i++)

p[i].st=p[i-1].st+p[i-1].bt;

p[1].ft=p[1].bt;

for(i=2;i<=n;i++)

p[i].ft=p[i-1].ft+p[i].bt;

printf("\nGant Chart is as follows:\n\n");

printf("%d->P%d->%d",p[1].st,p[1].id,p[1].ft);

for(i=2;i<=n;i++)

printf("->P%d->%d",p[i].id,p[i].ft);

printf("\n");

}

/\* FUNCTION FOR SORTING ON THE BASIS OF ARRIVAL TIME OF GIVEN INPUTS FOR SJSF \*/

void sjsort(int n)

{

int i,j;

for(i=2;i<=n;i++)

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

if(p[j].at>p[i].at)

{

temp=p[i];

p[i]=p[j];

p[j]=temp;

}

else if(p[j].at==p[i].at) //if both the arrival time are equal then sort on the basis of burst time

if(p[j].bt>p[i].bt)

{

temp=p[i];

p[i]=p[j];

p[j]=temp;

}

}

/\* FUNCTION FOR DISPLAYING GANTT CHART FOR SHORTEST JOB FIRST SCHEDULING \*/

void ganttsj(int n)

{

printf("Here is ghant chart for SJF");

int i,j,min,nextval,limit,ts,m=1;

limit=0;

nextval=p[1].at;

printf("\nEnter the time slice: ");

scanf("%d",&ts);

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

limit=limit+p[i].bt;

limit=limit+p[1].at;

if(p1[1].bt<ts)

nextval=nextval+p1[1].bt;

else

nextval=nextval+ts;

printf("Gantt chart is as follows\n\n");

printf("%d->P%d->%d",p1[1].at,p1[1].id,nextval);

if(p1[m].bt<ts)

p1[m].bt=0;

else

p1[m].bt=p1[m].bt-ts;

do

{

if(p1[m].bt<=0)

{

min=9999;

for(i=1;(p1[i].at)<=nextval,i<=n;i++)

if(p1[i].bt<min && p1[i].bt>0)

{

min=p1[i].bt;

m=i;

}

}

for(i=1;(p1[i].at)<=nextval,i<=n;i++)

if((p1[i].bt<p1[m].bt) && (p1[m].bt>0) && (p1[i].bt>0))

m=i;

if(p1[m].bt<ts)

nextval=nextval+p1[m].bt;

else

nextval=nextval+ts;

printf("->P%d->%d",p1[m].id,nextval);

if(p1[m].bt<ts)

p1[m].bt=0;

else

p1[m].bt=p1[m].bt-ts;

if(p1[m].bt==0)

p[m].ft=nextval;

}while(nextval<limit);

}

/\* FUNCTION FOR DISPLAYING GANTT CHART FOR DISPLAYING FOR PRIORITY SCHEDULING \*/

void ganttps(int n)

{

int i,j,nextval,limit;

printf("\nFor pirority queue");

printf("\n==============================================================\n");

printf("\nProcess\tAT\tBT\tPRIORITY");

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

printf("\nP%d\t%d\t%d\t%d",p[i].id,p[i].at,p[i].bt,p[i].pr);

printf("\n==============================================================\n");

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

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

if((p[i].pr > p[j].pr))

{

temp=p[j];

p[j]=p[i];

p[i]=temp;

}

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

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

{

if((p[i].pr == p[j].pr))

{

temp=p[j];

p[j]=p[i];

p[i]=temp;

}

}

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

p1[i]=p[i];

printf("Gantt chart is as follows\n\n");

printf("\n%d",p[1].at);

nextval=p1[1].at;

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

{

nextval=nextval+p1[i].bt;

printf("->P%d->%d",p1[i].id,nextval);

p[i].ft=nextval;

}

}

/\* FUNCTION FOR DISPLAYING GANTT CHART FOR ROUND ROBIN SCHEDULING \*/

void ganttrr(int n)

{

int i,ts,m,nextval,nextarr;

nextval=p[1].at;

printf("\nEnter time slice for RR: ");

scanf("%d",&ts);

nextval=p1[1].at;

printf("Gantt chart is as follows\n\n");

cout<<p1[1].at;

for(i=1;i<=n,p1[i].at<=nextval;i++)

q1.push(p1[i].id);

nextarr=p1[i].at;

while(!q1.empty())

{

m=q1.front();

q1.pop();

if(p1[m].bt>=ts)

nextval=nextval+ts;

else

nextval=nextval+p1[m].bt;

cout<<"->P"<<p1[m].id<<"->"<<nextval;

p1[m].bt=p1[m].bt-ts;

if(nextval<nextarr)

{

if(p1[m].bt>0)

q1.push(m);

if(p1[m].bt<=0)

p[m].ft=nextval;

}

else

{

while(i<=n&&p1[i].at<=nextval)

{

q1.push(p1[i].id);

i++;

}

if(i<=n)

nextarr=p1[i].at;

if(p1[m].bt>0)

q1.push(m);

if(p1[m].bt<=0)

p[m].ft=nextval;

}

}

printf("\n\n################################################################################");

}

/\* FUNCTION FOR CALCULATING TURN AROUND TIME OR WAIT TIME \*/

float turnwait(int n)

{

int i;

float ret;

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

{

p[i].tat=p[i].ft-p[i].at;

p[i].wt=p[i].tat-p[i].bt;

p[0].tat=p[0].tat+p[i].tat;

p[0].wt=p[0].wt+p[i].wt;

}

p[0].tat=p[0].tat/n;

p[0].wt=p[0].wt/n;

ret=p[0].wt;

return ret;

}

/\* FUNCTION FOR DISPLAYING THE TABLE \*/

void display(int n)

{

int i;

cout<<"\n\n-------------------TABLE----------------------------------\n";

printf("\nProcess\tAT\tBT\tFT\tTAT\t\tWT");

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

printf("\nP%d\t%d\t%d\t%d\t%f\t%f",p[i].id,p[i].at,p[i].bt,p[i].ft,p[i].tat,p[i].wt);

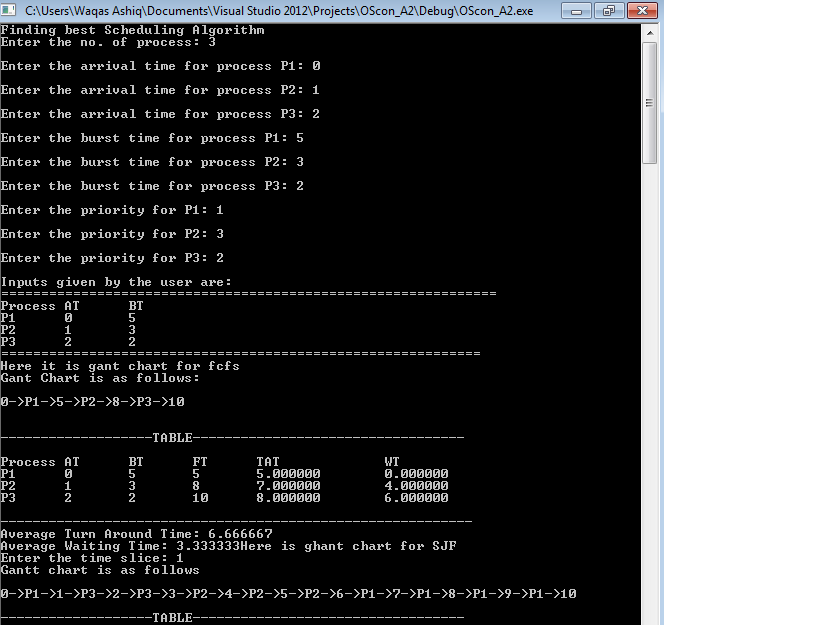
cout<<"\n\n-----------------------------------------------------------";

printf("\nAverage Turn Around Time: %f",p[0].tat);

printf("\nAverage Waiting Time: %f",p[0].wt);

}

***Output:***

******

******