**OS : Practical Assignment no. 4**

***Title* :**

Write a menu driven program for implementing CPU Scheduling Algorithms - FCFS, SJF, Priority and Round Robin.

***Program* :**

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

void FCFS(int num, int burst\_time[])

{

int waiting\_time[num];

waiting\_time[0]=0;

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

{

waiting\_time[i]=waiting\_time[i-1]+burst\_time[i-1];

}

int turn\_around\_time[num];

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

{

turn\_around\_time[i]=waiting\_time[i]+burst\_time[i];

}

cout << "\nProcesses "<< " Burst time "

<< " Waiting time " << " Turn around time\n";

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

{

cout << " " << i+1 << "\t\t" << burst\_time[i] <<"\t "

<< waiting\_time[i] <<"\t\t " << turn\_around\_time[i] <<endl;

}

int avg\_waiting\_time=(accumulate(waiting\_time, waiting\_time + num, 0)/num);

int avg\_turn\_around\_time=(accumulate(turn\_around\_time, turn\_around\_time + num, 0)/num);

cout<<"\nAverage waiting time = "<<avg\_waiting\_time;

cout<<"\nAverage turn around time = "<<avg\_turn\_around\_time;

}

void SJF(int num, int burst\_time[], int index[])

{

int wt[20], tat[20], avwt=0, avtat=0, i, j;

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

for(j=i;j<num;j++){

if(burst\_time[j]<burst\_time[i]){

swap(burst\_time[i], burst\_time[j]);

swap(index[i], index[j]);

}

}

}

wt[0]=0;

cout<<endl<<"Process\t\tBurst Time\tWaiting Time\tTurnaround Time\n";

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

wt[i+1]= wt[i]+burst\_time[i];

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

avwt=avwt+wt[i];

avtat=avtat+tat[i];

cout<<"P"<<index[i]+1<<" \t\t"<<burst\_time[i]<<"\t\t"<<wt[i]<<"\t\t"<<tat[i]<<endl;

}

cout<<"\nAverage Waiting Time = "<<avwt/i<<endl;

cout<<"Average Turnaround Time = "<<avtat/num<<endl;

}

void SJF\_non(int num, int burst\_time[],int arrival\_time[], int index[])

{

int temp,tt=0,min,d,i,j;

float atat=0,awt=0,stat=0,swt=0;

int e[num],tat[num],wt[num];

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

{

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

{

if(burst\_time[i]>burst\_time[j])

{

swap(burst\_time[i], burst\_time[j]);

swap(index[i], index[j]);

swap(arrival\_time[i],arrival\_time[j]);

}

}

}

min=arrival\_time[0];

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

{

if(min>arrival\_time[i])

{

min=arrival\_time[i];

d=i;

}

}

tt=min;

e[d]=tt+burst\_time[d];

tt=e[d];

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

{

if(arrival\_time[i]!=min)

{

e[i]=burst\_time[i]+tt;

tt=e[i];

}

}

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

{

tat[i]=e[i]-arrival\_time[i];

stat=stat+tat[i];

wt[i]=tat[i]-burst\_time[i];

swt=swt+wt[i];

}

atat=stat/num;

awt=swt/num;

cout<<"\nProcess Arrival-time(s) Burst-time(s) Waiting-time(s) Turnaround-time(s)\n";

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

{

cout<<"P"<<index[i]+1<<" \t\t"<<arrival\_time[i]<<" \t\t"<<burst\_time[i]<<" \t\t"<<wt[i]<<" \t\t"<<tat[i]<<endl;

}

cout<<"\nAverage Waiting Time = "<<awt<<"\nAverage Turnaround Time = "<<atat;

}

void priority(int n, int p[], int bt[], int pr[])

{

int i,j,pos,temp,avg\_tat,avg\_wt,total=0;

int wt[n],tat[n];

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

pos=i;

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

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

pos=j;

}

temp=pr[i];

pr[i]=pr[pos];

pr[pos]=temp;

temp=bt[i];

bt[i]=bt[pos];

bt[pos]=temp;

temp=p[i];

p[i]=p[pos];

p[pos]=temp;

}

wt[0]=0;

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

wt[i]=0;

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

wt[i]+=bt[j];

total+=wt[i];

}

avg\_wt=total/n;

total=0;

cout<<"\nProcess Burst-time(s) Waiting-time(s) Turnaround-time(s)\n";

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

tat[i]=bt[i]+wt[i];

total+=tat[i];

cout<<"P "<<p[i]<<" \t\t"<<bt[i]<<"\t\t"<<wt[i]<<"\t\t"<<tat[i]<<endl;

}

avg\_tat=total/n;

cout<<"\nAverage Waiting Time = "<<avg\_wt;

cout<<"\nAverage Turnaround Time = "<<avg\_tat;

}

void findWaitingTime(int processes[], int n, int bt[], int wt[], int quantum)

{

int rem\_bt[n];

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

rem\_bt[i] = bt[i];

int t = 0;

while (1)

{

bool done = true;

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

{

if (rem\_bt[i] > 0)

{

done = false;

if (rem\_bt[i] > quantum)

{

t += quantum;

rem\_bt[i] -= quantum;

}

else

{

t = t + rem\_bt[i];

wt[i] = t - bt[i];

rem\_bt[i] = 0;

}

}

}

if (done == true)

break;

}

}

void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])

{

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

tat[i] = bt[i] + wt[i];

}

void round\_robin(int n, int processes[], int bt[], int quantum)

{

int wt[n], tat[n], total\_wt = 0, total\_tat = 0;

findWaitingTime(processes, n, bt, wt, quantum);

findTurnAroundTime(processes, n, bt, wt, tat);

cout << "\nProcesses "<< " Burst time "<< " Waiting time " << " Turn around time\n";

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

{

total\_wt = total\_wt + wt[i];

total\_tat = total\_tat + tat[i];

cout << " " << i+1 << "\t\t" << bt[i] <<"\t "<< wt[i] <<"\t\t " << tat[i] <<endl;

}

cout << "\nAverage waiting time = "<< (float)total\_wt / (float)n;

cout << "\nAverage turn around time = "<< (float)total\_tat / (float)n;

}

int main()

{

int ch=0,a=1;

while(a)

{

cout<<"\n\*-----\*-----\*-----\*-----\*-----\*-----\*-----\*-----\*"<<endl;

cout<<"\nCPU Scheduling Algorithms\n";

cout<<"-------------------------"<<endl;

cout<<"\n1.First Come First Serve\n2.Shortest Job First (Non-Preemptive)\n3.Shortest Job First (Preemptive)\n4.Priority Based\n5.Round Robin"<<endl;

cout<<"\nEnter your choice : ";

cin>>ch;

int num;

cout<<"\nEnter the number of processes : ";

cin>>num;

int burst\_time[num];

int index[num];

int arrival\_time[num];

cout<<"Enter the burst time for each process : "<<endl;

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

{

cout<<"P"<<i+1<<" : ";

cin>>burst\_time[i];

index[i]=i;

}

switch(ch)

{

case 1:"\nFirst Come First Serve";

FCFS(num, burst\_time);

break;

case 2:"\nShortest Job First (Non-Preemptive)";

SJF(num, burst\_time, index);

break;

case 3:

{

cout<<"\nShortest Job First (Preemptive)";

cout<<"\nEnter the arrival time for each process : "<<endl;

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

{

cout<<"P"<<i+1<<" : ";

cin>>arrival\_time[i];

}

SJF\_non(num, burst\_time, arrival\_time, index);

}

break;

case 4:

{

cout<<"\nPriority Based";

cout<<"\nEnter the Priority for each process : "<<endl;

int pr[num];

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

{

cout<<"Priority "<<i+1<<" : ";

cin>>pr[i];

}

priority(num, index, burst\_time, pr);

}

break;

case 5:

{

"\nRound Robin";

int quantum=0;

cout<<"\nEnter the time slice : ";

cin>>quantum;

round\_robin(num,index,burst\_time,quantum);

}

break;

default:"\nWrong choice.";

}

cout<<"\n\nDo you want to continue(1/0) : ";

cin>>a;

}

return 0;

}

***Output Screenshots***









