## Laboratory Work

#include<iostream>

#include <algorithm>

#include <iomanip>

#include <queue>

#include <string.h>

using namespace std;

void fcfs(){

int n,i,j;

double av[50],Arr\_Time[50], Serv\_T[50], Fin\_T[50], Turn\_T[50];

cout<<"Enter total number of processes:";

cin>>n;

cout<<"\nEnter Process Arrival Time\n";

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

{

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

cin>>Arr\_Time[i];

}

cout<<"\nEnter Process Service Time\n";

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

{

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

cin>>Serv\_T[i];

}

//calculating finish time

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

{

Fin\_T[i]=0;

for(j=0;j<i+1;j++)

Fin\_T[i]+=Serv\_T[j];

}

cout<<"\#P\t\tService Time\tArrival Time\tFinish Time\tTurnaround Time\tTr / Ts";

//calculating turnaround time

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

{

Turn\_T[i] = Fin\_T[i]-Arr\_Time[i];

av[i] = Turn\_T[i] / Serv\_T[i];

cout<<"\nP["<<i+1<<"]"<<"\t\t"<<Serv\_T[i]<<"\t\t"<<Arr\_Time[i]<<"\t\t"<<Fin\_T[i]<<"\t\t"<<Turn\_T[i]<<"\t\t"<<av[i];

}

}

struct process {

int pid;

float arrival\_time;

float service\_time;

float completion\_time;

float turnaround\_time;

float average;

int start\_time;

};

void roundRobin(){

int n;

int tq;

struct process p[100];

float service\_remaining[100];

float total\_turnaround\_time = 0;

int idx;

float average[100];

cout<<"Enter the number of processes: ";

cin>>n;

cout<<"Enter time quantum: ";

cin>>tq;

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

cout<<"Enter arrival time of process "<<i+1<<": ";

cin>>p[i].arrival\_time;

cout<<"Enter service time of process "<<i+1<<": ";

cin>>p[i].service\_time;

service\_remaining[i] = p[i].service\_time;

p[i].pid = i+1;

cout<<endl;

}

queue<int> q;

int current\_time = 0;

q.push(0);

int completed = 0;

int mark[100];

memset(mark,0,sizeof(mark));

mark[0] = 1;

while(completed != n) {

idx = q.front();

q.pop();

if(service\_remaining[idx]-tq > 0) {

service\_remaining[idx] -= tq;

current\_time += tq;

}

else {

current\_time += service\_remaining[idx];

service\_remaining[idx] = 0;

completed++;

p[idx].completion\_time = current\_time;

p[idx].turnaround\_time = p[idx].completion\_time - p[idx].arrival\_time;

p[idx].average = p[idx].turnaround\_time / p[idx].service\_time;

}

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

if(service\_remaining[i] > 0 && p[i].arrival\_time <= current\_time && mark[i] == 0) {

q.push(i);

mark[i] = 1;

}

}

if(service\_remaining[idx] > 0) {

q.push(idx);

}

if(q.empty()) {

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

if(service\_remaining[i] > 0) {

q.push(i);

mark[i] = 1;

break;

}

}

}

}

cout<<endl;

cout<<"#P\t"<<"AT\t"<<"ST\t"<<"FT\t"<<"TAT\t"<<"Tr/Ts\t"<<"\n"<<endl;

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

cout<<p[i].pid<<"\t"<<p[i].arrival\_time<<"\t"<<p[i].service\_time<<"\t"<<p[i].completion\_time<<"\t"<<p[i].turnaround\_time<<"\t"<<p[i].average<<"\t"<<"\n"<<endl;

}

}

void spn(){

int n;

struct process p[100];

float total\_turnaround\_time = 0;

int is\_completed[100];

float average[100];

memset(is\_completed,0,sizeof(is\_completed));

cout<<"Enter the number of processes: ";

cin>>n;

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

cout<<"Enter arrival time of process "<<i+1<<": ";

cin>>p[i].arrival\_time;

cout<<"Enter service time of process "<<i+1<<": ";

cin>>p[i].service\_time;

p[i].pid = i+1;

cout<<endl;

}

int current\_time = 0;

int completed = 0;

int prev = 0;

while(completed != n) {

int idx = -1;

int mn = 10000000;

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

if(p[i].arrival\_time <= current\_time && is\_completed[i] == 0) {

if(p[i].service\_time < mn) {

mn = p[i].service\_time;

idx = i;

}

if(p[i].service\_time == mn) {

if(p[i].arrival\_time < p[idx].arrival\_time) {

mn = p[i].service\_time;

idx = i;

}

}

}

}

if(idx != -1) {

p[idx].start\_time = current\_time;

p[idx].completion\_time = p[idx].start\_time + p[idx].service\_time;

p[idx].turnaround\_time = p[idx].completion\_time - p[idx].arrival\_time;

p[idx].average = p[idx].turnaround\_time / p[idx].service\_time;

is\_completed[idx] = 1;

completed++;

current\_time = p[idx].completion\_time;

prev = current\_time;

}

else {

current\_time++;

}

}

cout<<endl;

cout<<"#P\t"<<"AT\t"<<"ST\t"<<"FT\t"<<"TAT\t"<<"Tr/Ts\t"<<"\n"<<endl;

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

cout<<p[i].pid<<"\t"<<p[i].arrival\_time<<"\t"<<p[i].service\_time<<"\t"<<p[i].completion\_time<<"\t"<<p[i].turnaround\_time<<"\t"<<p[i].average<<"\t"<<"\n"<<endl;

}

}

int main()

{

while(true){

int choice;

cout<<"Choose method"<<endl;

cout<<"1.FCFS"<<endl;

cout<<"2.Round Robin"<<endl;

cout<<"3.SPN"<<endl;

cout<<"0.Exit"<<endl;

cin>>choice;

if(choice == 1){

fcfs();

}

if(choice == 2){

roundRobin();

}

if(choice == 3){

spn();

}

if(choice == 0){

break;

}

}

return 0;

}

## Practice Part

|  |  |  |
| --- | --- | --- |
| Process Time | Arrival Time | Service Time |
| A | 0 | 3 |
| B | 1 | 5 |
| C | 3 | 2 |
| D | 9 | 5 |
| E | 12 | 5 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | A | B | C | D | E |
| Arrival Time | 0 | 1 | 3 | 9 | 12 |
| Service Time  Ts | 3 | 5 | 2 | 5 | 5 |

FCFS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Finish Time | 3 | 8 | 10 | 15 | 20 |
| Turnaround Time Tr | 3 | 7 | 7 | 6 | 8 |
| Tr/Ts | 1.0 | 1.4 | 3.5 | 1.2 | 1.5 |

RR q=1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Finish Time | 6 | 11 | 8 | 18 | 20 |
| Turnaround Time Tr | 6 | 8 | 5 | 9 | 8 |
| Tr/Ts | 2 | 2 | 2.5 | 1.8 | 1.6 |

RR q=4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Finish Time | 3 | 10 | 9 | 19 | 20 |
| Turnaround Time Tr | 3 | 9 | 6 | 10 | 8 |
| Tr/Ts | 1 | 1.8 | 3 | 2 | 1.6 |

SPN

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Finish Time | 3 | 10 | 5 | 15 | 20 |
| Turnaround Time Tr | 3 | 9 | 2 | 6 | 8 |
| Tr/Ts | 1 | 1.8 | 1 | 1.2 | 1.6 |

