Name : Sarthak Pagar

Roll No. : 34

Classs : TE (IT)

Practical : 3

Statement : [Implement the C program for CPU Scheduling Algorithms: Shortest Job First (Preemptive) and Round Robin with different arrival time.](https://drive.google.com/file/d/1dJH_SnEUQ6jltqC6WddLZcd-kg_lKpeG/view?usp=sharing)

(I) Round Robin :-

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAX 20

typedef struct process{

int BT, AT, TAT, WT, PNO, PID;

char name[10];

} process;

typedef struct RQ{

process pr[MAX];

int f, r;

} RQ;

void get\_PCB(process[], int \*);

void sort\_AT(process[], int);

void sort\_BT(RQ[]);

void disp\_table(process[], int);

void RR(process p[], int n, int);

float cal\_avgwt(process[], int);

float cal\_avgtat(process[], int);

void menu(){

printf("\n\t\t\*\*\*\*MENU\*\*\*\*\*");

printf("\n\t\t1. RR");

printf("\n\t\t2. EXIT");

printf("\n\t\tEnter Choice: \t");

}

void main(){

int ch, TQ, n;

process P[MAX];

float avg\_WT, avg\_TAT;

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

scanf("%d", &TQ);

get\_PCB(P, &n);

do{

menu();

scanf("%d", &ch);

switch (ch){

case 1:

{

RR(P, n, TQ);

disp\_table(P, n);

avg\_WT = cal\_avgwt(P, n);

avg\_TAT = cal\_avgtat(P, n);

printf("\nAVERAGE WT : %f", avg\_WT);

printf("\nAVERAGE TAT : %f", avg\_TAT);

break;

}

case 2:

break;

}

} while (ch != 2);

}

float cal\_avgwt(process p[], int n){

float avg = 0;

int i;

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

avg += p[i].WT;

}

avg = avg / n;

return avg;

}

float cal\_avgtat(process p[], int n){

float avg = 0;

int i;

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

avg += p[i].TAT;

}

avg = avg / n;

return avg;

}

int get\_first\_process(process p[], int n){

int min, j, in;

min = p[0].AT;

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

if (p[j].AT < min){

in = j;

}

}

return in;

}

void check\_arrival(RQ \*r, process p[], int time, int n){

int i, j, flag = 0;

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

for (j = 0; j <= r->r; j++){

if (strcmp(p[i].name, r->pr[j].name) == 0)

flag = 1;

else

break;

}

if (p[i].AT == time && flag == 0){

r->r = r->r + 1;

r->pr[r->r] = p[i];

}

flag = 0;

}

}

void RR(process p[], int n, int tq){

int count = 0, i, start, time = 0;

RQ r;

r.f = r.r = -1;

start = get\_first\_process(p, n);

r.pr[0] = p[start];

r.f = r.r = 0;

check\_arrival(&r, p, time, n);

while (r.f != -1){

for (count = 0; count < tq; count++){

r.pr[r.f].BT--;

time++;

if (r.pr[r.f].BT == 0)

break;

check\_arrival(&r, p, time, n);

}

if (r.pr[r.f].BT != 0){

r.pr[r.r + 1] = r.pr[r.f];

r.r++;

}

else{

p[r.pr[r.f].PID].TAT = time - r.pr[r.f].AT;

p[r.pr[r.f].PID].WT = p[r.pr[r.f].PID].TAT - p[r.pr[r.f].PID].BT;

}

if (r.f == r.r)

r.f = r.r = -1;

else

r.f++;

}

}

void sort\_BT(RQ \*r){

int i, j;

process temp;

for (i = r->f; i <= r->r; i++){

for (j = i + 1; j <= r->r; j++){

if (r->pr[j].BT < r->pr[i].BT){

temp = r->pr[j];

r->pr[j] = r->pr[i];

r->pr[i] = temp;

}

}

}

}

int get\_total\_time(process p[], int n){

int i, sum = 0;

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

sum += p[i].BT;

}

return sum;

}

void sort\_AT(process p[], int n){

int i, j;

process temp;

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

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

if (p[j].AT < p[i].AT){

temp = p[j];

p[j] = p[i];

p[i] = temp;

}

}

}

}

void disp\_table(process p[], int n){

int i;

printf("\n\n P\_NAME \t AT \t BT \t WT \t TAT \t");

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

printf("\n %-10s \t %d \t %d \t %d \t %d \t", p[i].name, p[i].AT, p[i].BT, p[i].WT, p[i].TAT);

}

}

void get\_PCB(process p[], int \*n){

int i;

printf("\nEnter total no of processes : ");

scanf("%d", n);

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

printf("\n Enter Following details for Process\n%d", i + 1);

printf("\nName :\t");

scanf("%s", p[i].name);

printf("\nArrival Time :\t");

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

printf("\nBurst Time :\t");

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

p[i].PID = i;

}

}

Output :-

[Saru1594@localhost Downloads]$ gcc 3

3RR.c 3SJF(P).c

[Saru1594@localhost Downloads]$ gcc 3RR.c

[Saru1594@localhost Downloads]$ ./a.out

Enter Time Quantum for Round Robin : 2

Enter total no of processes : 6

Enter Following details for Process

1

Name : P1

Arrival Time : 0

Burst Time : 4

Enter Following details for Process

2

Name : P2

Arrival Time : 1

Burst Time : 5

Enter Following details for Process

3

Name : P3

Arrival Time : 2

Burst Time : 2

Enter Following details for Process

4

Name : P4

Arrival Time : 3

Burst Time : 1

Enter Following details for Process

5

Name : P5

Arrival Time : 4

Burst Time : 6

Enter Following details for Process

6

Name : P6

Arrival Time : 5

Burst Time : 3

\*\*\*\*MENU\*\*\*\*\*

1. RR

2. EXIT

Enter Choice: 1

P\_NAME AT BT WT TAT

P1 0 4 4 8

P2 1 5 12 17

P3 2 2 2 4

P4 3 1 5 6

P5 4 6 11 17

P6 5 3 11 14

AVERAGE WT : 7.500000

AVERAGE TAT : 11.000000

\*\*\*\*MENU\*\*\*\*\*

1. RR

2. EXIT

Enter Choice: 2

(ii) SJF\_P

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAX 20

typedef struct process {

int BT, AT, TAT, WT, PID;

char name[10];

} process;

void get\_PCB(process[], int \*);

void disp\_table(process[], int);

void SJF\_P(process[], int);

float cal\_avgwt(process[], int);

float cal\_avgtat(process[], int);

void menu() {

printf("\n\t\t\*\*\*MENU\*\*");

printf("\n\t\t1. SJF P");

printf("\n\t\t2. EXIT");

printf("\n\t\tEnter Choice: \t");

}

int main() {

int ch, TQ, n;

process P[MAX];

float avg\_WT, avg\_TAT;

get\_PCB(P, &n);

do {

menu();

scanf("%d", &ch);

switch (ch) {

case 1:

SJF\_P(P, n);

disp\_table(P, n);

avg\_WT = cal\_avgwt(P, n);

avg\_TAT = cal\_avgtat(P, n);

printf("\nAVERAGE WT : %.2f", avg\_WT); // Ensure floating-point output

printf("\nAVERAGE TAT : %.2f", avg\_TAT);

break;

case 2:

break;

default:

printf("Invalid choice! Please try again.");

}

} while (ch != 2);

return 0;

}

float cal\_avgwt(process p[], int n) {

float avg = 0;

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

avg += p[i].WT;

}

return avg / n; // Correct floating-point average calculation

}

float cal\_avgtat(process p[], int n) {

float avg = 0;

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

avg += p[i].TAT;

}

return avg / n; // Correct floating-point average calculation

}

void disp\_table(process p[], int n) {

printf("\n\n P\_NAME \t AT \t BT \t WT \t TAT \t");

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

printf("\n %-10s \t %d \t %d \t %d \t %d \t", p[i].name, p[i].AT, p[i].BT, p[i].WT, p[i].TAT);

}

}

void get\_PCB(process p[], int \*n) {

printf("\nEnter total number of processes: ");

scanf("%d", n);

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

printf("\nEnter Following details for Process %d", i + 1);

printf("\nName: ");

scanf("%s", p[i].name);

printf("Arrival Time: ");

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

printf("Burst Time: ");

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

p[i].TAT = 0; // Initialize TAT

p[i].WT = 0; // Initialize WT

p[i].PID = i; // Set process ID

}

}

void SJF\_P(process p[], int n) {

int time = 0, completed = 0;

int min\_index;

int is\_completed[MAX] = {0}; // Track completed processes

while (completed < n) {

min\_index = -1;

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

if (p[i].AT <= time && is\_completed[i] == 0) {

if (min\_index == -1 || p[i].BT < p[min\_index].BT) {

min\_index = i;

}

}

}

if (min\_index != -1) {

time += p[min\_index].BT; // Increment time by burst time of selected process

p[min\_index].TAT = time - p[min\_index].AT; // Calculate TAT

p[min\_index].WT = p[min\_index].TAT - p[min\_index].BT; // Calculate WT

is\_completed[min\_index] = 1; // Mark process as completed

completed++; // Increment completed process count

} else {

time++; // If no process is ready, just increment time

}

}

}

Output :-

[Saru1594@localhost Downloads]$ gcc 3SJF(P).c

bash: syntax error near unexpected token `('

[Saru1594@localhost Downloads]$ gcc 3SJF\(P\).c

[Saru1594@localhost Downloads]$ ./a.out

Enter total number of processes: 5

Enter Following details for Process 1

Name: P1

Arrival Time: 1

Burst Time: 7

Enter Following details for Process 2

Name: P2

Arrival Time: 2

Burst Time: 5

Enter Following details for Process 3

Name: P3

Arrival Time: 3

Burst Time: 1

Enter Following details for Process 4

Name: P4

Arrival Time: 4

Burst Time: 2

Enter Following details for Process 5

Name: P5

Arrival Time: 5

Burst Time: 8

\*\*\*MENU\*\*

1. SJF P

2. EXIT

Enter Choice: 1

P\_NAME AT BT WT TAT

P1 1 7 0 7

P2 2 5 9 14

P3 3 1 5 6

P4 4 2 5 7

P5 5 8 11 19

AVERAGE WT : 6.00

AVERAGE TAT : 10.60

\*\*\*MENU\*\*

1. SJF P

2. EXIT

Enter Choice: 2