**Non PreEmptive First Come First Serve Scheduling :**

#include <stdio.h>

struct Process

{

int pid;

int burst\_time;

int arrival\_time;

int waiting\_time;

int turnaround\_time;

};

int main()

{

int n;

printf("Enter the number of processes: ");

scanf("%d", &n);

struct Process processes[n];

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

{

printf("Enter the burst time for process %d: ", i + 1);

scanf("%d", &processes[i].burst\_time);

printf("Enter the arrival time for process %d: ", i + 1);

scanf("%d", &processes[i].arrival\_time);

processes[i].pid = i + 1;

}

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - i - 1; j++)

{

if (processes[j].arrival\_time > processes[j + 1].arrival\_time)

{

struct Process temp = processes[j];

processes[j] = processes[j + 1];

processes[j + 1] = temp;

}

}

}

int total\_waiting\_time = 0;

int total\_turnaround\_time = 0;

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

{

if (i == 0)

{

processes[i].waiting\_time = 0;

}

else

{

int completion\_time = processes[i - 1].turnaround\_time + processes[i - 1].arrival\_time;

processes[i].waiting\_time = (completion\_time > processes[i].arrival\_time) ? (completion\_time - processes[i].arrival\_time) : 0;

}

processes[i].turnaround\_time = processes[i].waiting\_time + processes[i].burst\_time;

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

total\_turnaround\_time += processes[i].turnaround\_time;

}

double average\_waiting\_time = (double)total\_waiting\_time / n;

double average\_turnaround\_time = (double)total\_turnaround\_time / n;

printf("Process\tBurst Time\tArrival Time\tWaiting Time\tTurnaround Time\n");

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

{

printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", processes[i].pid, processes[i].burst\_time,

processes[i].arrival\_time, processes[i].waiting\_time, processes[i].turnaround\_time);

}

printf("\nAverage Waiting Time: %.2f\n", average\_waiting\_time);

printf("Average Turnaround Time: %.2f\n", average\_turnaround\_time);

return 0;

}

**OUTPUT:**

**1:**

Enter the number of processes: 4

Enter the burst time for process 1: 4

Enter the arrival time for process 1: 2

Enter the burst time for process 2: 3

Enter the arrival time for process 2: 3

Enter the burst time for process 3: 2

Enter the arrival time for process 3: 1

Enter the burst time for process 4: 5

Enter the arrival time for process 4: 4

Process Burst Time Arrival Time Waiting Time Turnaround Time

3 2 1 0 2

1 4 2 1 5

2 3 3 4 7

4 5 4 6 11

Average Waiting Time: 2.75

Average Turnaround Time: 6.25

**2:**

Enter the number of processes: 4

Enter the burst time for process 1: 4

Enter the arrival time for process 1: 0

Enter the burst time for process 2: 3

Enter the arrival time for process 2: 0

Enter the burst time for process 3: 2

Enter the arrival time for process 3: 0

Enter the burst time for process 4: 5

Enter the arrival time for process 4: 0

Process Burst Time Arrival Time Waiting Time Turnaround Time

1 4 0 0 4

2 3 0 4 7

3 2 0 7 9

4 5 0 9 14

Average Waiting Time: 5.00

Average Turnaround Time: 8.50

**Non-PreEmptive Priority Scheduling:**

#include <stdio.h>

struct Process

{

int pid;

int burst\_time;

int priority;

int arrival\_time;

int waiting\_time;

int turnaround\_time;

};

void calculate\_times(struct Process\* processes, int n)

{

int total\_waiting\_time = 0;

int total\_turnaround\_time = 0;

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

{

if (i == 0)

{

processes[i].waiting\_time = 0;

}

else

{

int completion\_time = processes[i - 1].turnaround\_time + processes[i - 1].arrival\_time;

processes[i].waiting\_time = (completion\_time > processes[i].arrival\_time) ? (completion\_time - processes[i].arrival\_time) : 0;

}

processes[i].turnaround\_time = processes[i].waiting\_time + processes[i].burst\_time;

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

total\_turnaround\_time += processes[i].turnaround\_time;

}

double average\_waiting\_time = (double)total\_waiting\_time / n;

double average\_turnaround\_time = (double)total\_turnaround\_time / n;

printf("Process\tBurst Time\tArrival Time\tPriority\tWaiting Time\tTurnaround Time\n");

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

{

printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", processes[i].pid, processes[i].burst\_time,

processes[i].arrival\_time, processes[i].priority, processes[i].waiting\_time, processes[i].turnaround\_time);

}

printf("\nAverage Waiting Time: %.2f\n", average\_waiting\_time);

printf("Average Turnaround Time: %.2f\n", average\_turnaround\_time);

}

int main()

{

int n;

printf("Enter the number of processes: ");

scanf("%d", &n);

struct Process processes[n];

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

{

printf("Enter the arrival time, burst time and priority for process %d: ", i + 1);

scanf("%d", &processes[i].arrival\_time);

scanf("%d", &processes[i].burst\_time);

scanf("%d", &processes[i].priority);

processes[i].pid = i + 1;

}

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - i - 1; j++)

{

if (processes[j].arrival\_time > processes[j + 1].arrival\_time ||

(processes[j].arrival\_time == processes[j + 1].arrival\_time &&

processes[j].priority > processes[j + 1].priority))

{

struct Process temp = processes[j];

processes[j] = processes[j + 1];

processes[j + 1] = temp;

}

}

}

calculate\_times(processes, n);

return 0;

}

**OUTPUT:**

**1:**

Enter the number of processes: 4

Enter the arrival time, burst time and priority for process 1: 2 4 2

Enter the arrival time, burst time and priority for process 2: 0 3 1

Enter the arrival time, burst time and priority for process 3: 3 2 3

Enter the arrival time, burst time and priority for process 4: 4 5 4

Process Burst Time Arrival Time Priority Waiting Time Turnaround Time

2 3 0 1 0 3

1 4 2 2 1 5

3 2 3 3 4 6

4 5 4 4 5 10

Average Waiting Time: 2.50

Average Turnaround Time: 6.00

**2:**

Enter the number of processes: 4

Enter the arrival time, burst time and priority for process 1: 0 4 2

Enter the arrival time, burst time and priority for process 2: 0 3 1

Enter the arrival time, burst time and priority for process 3: 0 2 3

Enter the arrival time, burst time and priority for process 4: 0 5 4

Process Burst Time Arrival Time Priority Waiting Time Turnaround Time

2 3 0 1 0 3

1 4 0 2 3 7

3 2 0 3 7 9

4 5 0 4 9 14

Average Waiting Time: 4.75

Average Turnaround Time: 8.25

**Non-PreEmptive Shortest Job First Scheduling:**

#include <stdio.h>

struct Process

{

int pid;

int burst\_time;

int arrival\_time;

int waiting\_time;

int turnaround\_time;

};

void calculate\_times(struct Process\* processes, int n)

{

int total\_waiting\_time = 0;

int total\_turnaround\_time = 0;

int current\_time = 0;

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

{

int min\_burst\_index = -1;

int min\_burst\_time = 999999;

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

{

if (processes[j].arrival\_time <= current\_time && processes[j].burst\_time < min\_burst\_time && processes[j].burst\_time > 0)

{

min\_burst\_time = processes[j].burst\_time;

min\_burst\_index = j;

}

}

if (min\_burst\_index == -1)

{

current\_time++;

i--;

continue;

}

int process\_index = min\_burst\_index;

processes[process\_index].waiting\_time = current\_time - processes[process\_index].arrival\_time;

if (processes[process\_index].waiting\_time < 0)

processes[process\_index].waiting\_time = 0;

processes[process\_index].turnaround\_time = processes[process\_index].waiting\_time + processes[process\_index].burst\_time;

current\_time += processes[process\_index].burst\_time;

processes[process\_index].burst\_time = 0;

total\_waiting\_time += processes[process\_index].waiting\_time;

total\_turnaround\_time += processes[process\_index].turnaround\_time;

}

double average\_waiting\_time = (double)total\_waiting\_time / n;

double average\_turnaround\_time = (double)total\_turnaround\_time / n;

printf("Process\tWaiting Time\tTurnaround Time\n");

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

{

printf("%d\t\t%d\t\t%d\n", processes[i].pid, processes[i].waiting\_time, processes[i].turnaround\_time);

}

printf("\nAverage Waiting Time: %.2f\n", average\_waiting\_time);

printf("Average Turnaround Time: %.2f\n", average\_turnaround\_time);

}

int main()

{

int n;

printf("Enter the number of processes: ");

scanf("%d", &n);

struct Process processes[n];

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

{

printf("Enter the arrival time and burst time for process %d: ", i + 1);

scanf("%d %d", &processes[i].arrival\_time, &processes[i].burst\_time);

processes[i].pid = i + 1;

}

calculate\_times(processes, n);

return 0;

}

**OUTPUT:**

Enter the number of processes: 4

Enter the arrival time and burst time for process 1: 1 4

Enter the arrival time and burst time for process 2: 0 3

Enter the arrival time and burst time for process 3: 2 2

Enter the arrival time and burst time for process 4: 3 5

Process Waiting Time Turnaround Time

1 4 8

2 0 3

3 1 3

4 6 11

Average Waiting Time: 2.75

Average Turnaround Time: 6.25

Enter the number of processes: 4

Enter the arrival time and burst time for process 1: 0 4

Enter the arrival time and burst time for process 2: 0 3

Enter the arrival time and burst time for process 3: 0 2

Enter the arrival time and burst time for process 4: 0 5

Process Waiting Time Turnaround Time

1 5 9

2 2 5

3 0 2

4 9 14

Average Waiting Time: 4.00

Average Turnaround Time: 7.50

**PreEmptive Round Robin Scheduling:**

#include<stdio.h>

int main()

{

int i, limit, total = 0, x, counter = 0, time\_quantum;

int wait\_time = 0, turnaround\_time = 0, arrival\_time[10], burst\_time[10], temp[10];

float average\_wait\_time, average\_turnaround\_time;

printf("Enter Total Number of Processes: ");

scanf("%d", &limit);

x = limit;

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

{

printf("Enter Details of Process %d:\n", i + 1);

printf("Arrival Time: ");

scanf("%d", &arrival\_time[i]);

printf("Burst Time: ");

scanf("%d", &burst\_time[i]);

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

}

printf("Enter Time Quantum: ");

scanf("%d", &time\_quantum);

printf("Process ID\tBurst Time\tTurnaround Time\tWaiting Time\n");

for(total = 0, i = 0; x != 0;)

{

if(temp[i] <= time\_quantum && temp[i] > 0)

{

total = total + temp[i];

temp[i] = 0;

counter = 1;

}

else if(temp[i] > 0)

{

temp[i] = temp[i] - time\_quantum;

total = total + time\_quantum;

}

if(temp[i] == 0 && counter == 1)

{

x--;

printf("Process%d\t%d\t%d\t%d\n", i + 1, burst\_time[i], total - arrival\_time[i], total - arrival\_time[i] - burst\_time[i]);

wait\_time = wait\_time + total - arrival\_time[i] - burst\_time[i];

turnaround\_time = turnaround\_time + total - arrival\_time[i];

counter = 0;

}

if(i == limit - 1)

{

i = 0;

}

else if(arrival\_time[i + 1] <= total)

{

i++;

}

else

{

i = 0;

}

}

average\_wait\_time = wait\_time \* 1.0 / limit;

average\_turnaround\_time = turnaround\_time \* 1.0 / limit;

printf("\nAverage Waiting Time: %f", average\_wait\_time);

printf("\nAverage Turnaround Time: %f", average\_turnaround\_time);

return 0;

}

**OUTPUT:**

**1:**

Enter Total Number of Processes: 4

Enter Details of Process 1:

Arrival Time: 0

Burst Time: 4

Enter Details of Process 2:

Arrival Time: 3

Burst Time: 4

Enter Details of Process 3:

Arrival Time: 5

Burst Time: 6

Enter Details of Process 4:

Arrival Time: 6

Burst Time: 5

Enter Time Quantum: 2

Process ID Burst Time Turnaround Time Waiting Time

Process1 4 4 0

Process2 4 9 5

Process3 6 13 7

Process4 5 13 8

Average Waiting Time: 5.000000

Average Turnaround Time: 9.750000

**2:**

Enter Total Number of Processes: 4

Enter Details of Process 1:

Arrival Time: 0

Burst Time: 5

Enter Details of Process 2:

Arrival Time: 0

Burst Time: 7

Enter Details of Process 3:

Arrival Time: 0

Burst Time: 9

Enter Details of Process 4:

Arrival Time: 0

Burst Time: 4

Enter Time Quantum: 2

Process ID Burst Time Turnaround Time Waiting Time

Process4 4 16 12

Process1 5 17 12

Process2 7 22 15

Process3 9 25 16

Average Waiting Time: 13.750000

Average Turnaround Time: 20.000000

**PreEmptive Priority SCheduling:**

#include<stdio.h>

struct process

{

int WT,AT,BT,TAT,PT;

};

struct process a[10];

int main()

{

int n,temp[10],t,count=0,short\_p;

float total\_WT=0,total\_TAT=0,Avg\_WT,Avg\_TAT;

printf("Enter the number of the process: ");

scanf("%d",&n);

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

{

printf("Enter the arrival time , burst time and priority of the process %d: ", i+1);

scanf("%d%d%d",&a[i].AT,&a[i].BT,&a[i].PT);

temp[i]=a[i].BT;

}

a[9].PT=10000;

for(t=0;count!=n;t++)

{

short\_p=9;

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

{

if(a[short\_p].PT>a[i].PT && a[i].AT<=t && a[i].BT>0)

{

short\_p=i;

}

}

a[short\_p].BT=a[short\_p].BT-1;

if(a[short\_p].BT==0)

{

count++;

a[short\_p].WT=t+1-a[short\_p].AT-temp[short\_p];

a[short\_p].TAT=t+1-a[short\_p].AT;

total\_WT=total\_WT+a[short\_p].WT;

total\_TAT=total\_TAT+a[short\_p].TAT;

}

}

Avg\_WT=total\_WT/n;

Avg\_TAT=total\_TAT/n;

printf("ID WT TAT\n");

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

{

printf("%d %d\t%d\n",i+1,a[i].WT,a[i].TAT);

}

printf("Average waiting time of the process is %f\n",Avg\_WT);

printf("Average turn around time of the process is %f\n",Avg\_TAT);

return 0;

}

**OUTPUT:**

**1:**

Enter the number of the process: 4

Enter the arrival time , burst time and priority of the process 1: 0 4 2

Enter the arrival time , burst time and priority of the process 2: 2 3 1

Enter the arrival time , burst time and priority of the process 3: 0 4 3

Enter the arrival time , burst time and priority of the process 4: 6 2 4

ID WT TAT

1 3 7

2 0 3

3 7 11

4 5 7

Average waiting time of the process is 3.750000

Average turn around time of the process is 7.000000

**PreEmptive Shortest Job First:**

#include <stdio.h>

int main()

{

int arrival\_time[10], burst\_time[10], temp[10];

int i, smallest, count = 0, time, limit;

double wait\_time = 0, turnaround\_time = 0, end;

float average\_waiting\_time, average\_turnaround\_time;

printf("Enter the Total Number of Processes: ");

scanf("%d", &limit);

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

{

printf("Enter Details of Processes %d: \n", i+1);

printf("Enter Arrival Time: ");

scanf("%d", &arrival\_time[i]);

printf("Enter Burst Time: ");

scanf("%d", &burst\_time[i]);

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

}

burst\_time[9] = 9999;

for(time = 0; count != limit; time++)

{

smallest = 9;

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

{

if(arrival\_time[i] <= time && burst\_time[i] < burst\_time[smallest] && burst\_time[i] > 0)

{

smallest = i;

}

}

burst\_time[smallest]--;

if(burst\_time[smallest] == 0)

{

count++;

end = time + 1;

wait\_time = wait\_time + end - arrival\_time[smallest] - temp[smallest];

turnaround\_time = turnaround\_time + end - arrival\_time[smallest];

}

}

average\_waiting\_time = wait\_time / limit;

average\_turnaround\_time = turnaround\_time / limit;

printf("Average Waiting Time: %lf\n", average\_waiting\_time);

printf("Average Turnaround Time: %lf", average\_turnaround\_time);

return 0;

}

**OUTPUT:**

**1:**

Enter the Total Number of Processes: 4

Enter Details of Processes 1:

Enter Arrival Time: 0

Enter Burst Time: 5

Enter Details of Processes 2:

Enter Arrival Time: 2

Enter Burst Time: 7

Enter Details of Processes 3:

Enter Arrival Time: 5

Enter Burst Time: 2

Enter Details of Processes 4:

Enter Arrival Time: 6

Enter Burst Time: 3

Average Waiting Time: 2.250000

Average Turnaround Time: 6.500000

**2:**

Enter the Total Number of Processes: 4

Enter Details of Processes 1:

Enter Arrival Time: 0

Enter Burst Time: 5

Enter Details of Processes 2:

Enter Arrival Time: 2

Enter Burst Time: 1

Enter Details of Processes 3:

Enter Arrival Time: 4

Enter Burst Time: 1

Enter Details of Processes 4:

Enter Arrival Time: 6

Enter Burst Time: 3

Average Waiting Time: 0.750000

Average Turnaround Time: 3.250000