**Student Name: Aditi Utreja**

**Student ID:**

**Email Address: aditir851@gmail.com**

**GitHub Link:**

**Code: 9**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main()

{

FILE \*fp = fopen("cpu\_burst.txt", "r");

int bt[20],p[20],wt[20],tat[20],i=0,j,n=5,total=0,pos,temp;

float avg\_wt,avg\_tat;

printf("\nReading CPU\_BURST.txt File\n");

while((getc(fp))!=EOF)

{

fscanf(fp, "%d", &bt[i]);

if(bt[i]>0){

p[i]=i+1;

i++;

}

}

n=i;

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

{

pos=i;

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

{

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

pos=j;

}

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=(float)total/n;

total=0;

printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");

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

{

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

total+=tat[i];

printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

}

avg\_tat=(float)total/n;

printf("\n\nAverage Waiting Time=%f",avg\_wt);

printf("\nAverage Turnaround Time=%f\n",avg\_tat);

fclose(fp);

return 0;

}

**DESCRIPTION:-**

Shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next. It has the advantage of having a minimum average waiting time among all scheduling algorithms. It is a Greedy Algorithm. It may cause starvation if shorter processes keep on coming but this problem can be solved by the process of ageing.

SJF can be used in specialized environment where accurate estimates of running time are available.

1. Completion Time: Time at which process completes its execution.
2. Turn Around Time: Time Difference between completion time and arrival time. Turn Around Time = Completion Time – Arrival Time
3. Waiting Time(WT): Time Difference between turnaround time and burst time.  
   Waiting Time = Turn Around Time – Burst Time
4. Arrival Time: Time at which process arrives. In this question we are given with the same arrival time for all the processes.

There are basically two types of SJF methods:

1) Non-Preemptive SJF:- In non-preemptive scheduling, once the CPU cycle is allocated to process, the process holds it till it reaches a waiting state or terminated.

2) Preemptive SJF:- In Preemptive SJF Scheduling, jobs are put into the ready queue as they come.

Shortest Job First can improve process throughput by making sure that shorter jobs are executed first, hence possibly have a short turnaround time. It is frequently used for long term scheduling. Probably optimal with regard to average turnaround time.

**Algorithm:**

1. Traverse until all process gets completely executed.
2. Find process with minimum remaining time at every single time lap.
3. Reduce its time by 1.
4. Check if its remaining time becomes 0.
5. Increment the counter of process completion.
6. Completion time of current process = current time + 1
7. Calculate waiting time for each completed process.

WT= turnaround time – Burst time

1. Increment time lap by 1.
2. Find average waiting time.

**Complexity:**

Mention all the complexities line to line in the code. Overall Complexity of SJF to calculate average waiting time and Turnaround Time is O(nLogn).

**Constraints:**

1. Job completion time must be known earlier, but it is hard to predict.
2. It is often used in a batch system for long term scheduling.
3. SJF can’t be implemented for CPU scheduling for short term. It is because there is no specific method to predict the length of upcoming CPU burst.
4. Requires a knowledge of how long a process will run.
5. It leads to the starvation that does not reduce average turnaround time.
6. It is hard to know the length of the upcoming CPU request.

**Additional Algorithm:**

We know that it is not Practically implementable because burst time of a process can’t be predicted in advance. But if you don’t know the length of the next CPU burst, then we can predict it. And we expect the next CPU burst will be similar in length to the previous one.

And I assume arrival time for all the processes are 0 (because we have given that all the processes arrives at the same time).

**Test Cases:**

We select five processes with their arrival time and burst time. These processes are scheduled by Round Robin,

Preemptive SJF and proposed algorithm. We calculate and compare average waiting time, contest switching and

turnaround time of each scheduling algorithm. We execute many processes sets and calculate and compare them.

The example is shown below,

Table 1:A set of five processes with arrival time and burst time

Processes Time of Arrival Burst Duration (ms)

P1

P2

P3

P4

P5

0

2

5

6

8

4.0

7.0

5.0

8.0

9.0

1. Round Robin with quantum = 5

Gantt chart:

P1

P2

P3 P4

P5 P2

P4

P5

No. of context switches = 7

Average waiting time= 11.4 ms

Average turnaround time= 18ms

2. Shortest Remaining Job First:

Gant chart:

P1 P2 P3 P2 P4 P5

No. of context switches = 5

Average waiting time = 6.6 ms

Average turnaround time = 13.2ms

3. Proposed Algorithm

Gantt chart:

P1 P2 P3 P4 P5

No. of context switches = 4

Average waiting time= 6.8 ms

Average turnaround time= 13.4ms

We select five processes with their arrival time and burst time. These processes are scheduled by Round Robin,

Preemptive SJF and proposed algorithm. We calculate and compare average waiting time, contest switching and

turnaround time of each scheduling algorithm. We execute many processes sets and calculate and compare them.

The example is shown below,

Table 1:A set of five processes with arrival time and burst time

Processes Time of Arrival Burst Duration (ms)

P1

P2

P3

P4

P5

0

2

5

6

8

4.0

7.0

5.0

8.0

9.0

1. Round Robin with quantum = 5

Gantt chart:

P1

P2

P3 P4

P5 P2

P4

P5

No. of context switches = 7

Average waiting time= 11.4 ms

Average turnaround time= 18ms

2. Shortest Remaining Job First:

Gant chart:

P1 P2 P3 P2 P4 P5

No. of context switches = 5

Average waiting time = 6.6 ms

Average turnaround time = 13.2ms

3. Proposed Algorithm

Gantt chart:

P1 P2 P3 P4 P5

No. of context switches = 4

Average waiting time= 6.8 ms

Average turnaround time= 13.4ms

We select five processes with their arrival time and burst time. These processes are scheduled by Round Robin,

Preemptive SJF and proposed algorithm. We calculate and compare average waiting time, contest switching and

turnaround time of each scheduling algorithm. We execute many processes sets and calculate and compare them.

The example is shown below,

Table 1:A set of five processes with arrival time and burst time

Processes Time of Arrival Burst Duration (ms)

P1

P2

P3

P4

P5

0

2

5

6

8

4.0

7.0

5.0

8.0

9.0

1. Round Robin with quantum = 5

Gantt chart:

P1

P2

P3 P4

P5 P2

P4

P5

No. of context switches = 7

Average waiting time= 11.4 ms

Average turnaround time= 18ms

2. Shortest Remaining Job First:

Gant chart:

P1 P2 P3 P2 P4 P5

No. of context switches = 5

Average waiting time = 6.6 ms

Average turnaround time = 13.2ms

3. Proposed Algorithm

Gantt chart:

P1 P2 P3 P4 P5

No. of context switches = 4

Average waiting time= 6.8 ms

Average turnaround time= 13.4ms

We select five processes with their arrival time and burst time. These processes are scheduled by Round Robin,

Preemptive SJF and proposed algorithm. We calculate and compare average waiting time, contest switching and

turnaround time of each scheduling algorithm. We execute many processes sets and calculate and compare them.

The example is shown below,

Table 1:A set of five processes with arrival time and burst time

Processes Time of Arrival Burst Duration (ms)

P1

P2

P3

P4

P5

0

2

5

6

8

4.0

7.0

5.0

8.0

9.0

1. Round Robin with quantum = 5

Gantt chart:

P1

P2

P3 P4

P5 P2

P4

P5

No. of context switches = 7

Average waiting time= 11.4 ms

Average turnaround time= 18ms

2. Shortest Remaining Job First:

Gant chart:

P1 P2 P3 P2 P4 P5

No. of context switches = 5

Average waiting time = 6.6 ms

Average turnaround time = 13.2ms

3. Proposed Algorithm

Gantt chart:

P1 P2 P3 P4 P5

No. of context switches = 4

Average waiting time= 6.8 ms

Average turnaround time= 13.4ms

We select five processes with their arrival time and burst time. These processes are scheduled by Round Robin,

Preemptive SJF and proposed algorithm. We calculate and compare average waiting time, contest switching and

turnaround time of each scheduling algorithm. We execute many processes sets and calculate and compare them.

The example is shown below,

Table 1:A set of five processes with arrival time and burst time

Processes Time of Arrival Burst Duration (ms)

P1

P2

P3

P4

P5

0

2

5

6

8

4.0

7.0

5.0

8.0

9.0

1. Round Robin with quantum = 5

Gantt chart:

P1

P2

P3 P4

P5 P2

P4

P5

No. of context switches = 7

Average waiting time= 11.4 ms

Average turnaround time= 18ms

2. Shortest Remaining Job First:

Gant chart:

P1 P2 P3 P2 P4 P5

No. of context switches = 5

Average waiting time = 6.6 ms

Average turnaround time = 13.2ms

3. Proposed Algorithm

Gantt chart:

P1 P2 P3 P4 P5

No. of context switches = 4

Average waiting time= 6.8 ms

Average turnaround time= 13.4ms

We select five processes with their arrival time and burst time. These processes are scheduled by Round Robin,

Preemptive SJF and proposed algorithm. We calculate and compare average waiting time, contest switching and

turnaround time of each scheduling algorithm. We execute many processes sets and calculate and compare them.

The example is shown below,

Table 1:A set of five processes with arrival time and burst time

Processes Time of Arrival Burst Duration (ms)

P1

P2

P3

P4

P5

0

2

5

6

8

4.0

7.0

5.0

8.0

9.0

1. Round Robin with quantum = 5

Gantt chart:

P1

P2

P3 P4

P5 P2

P4

P5

No. of context switches = 7

Average waiting time= 11.4 ms

Average turnaround time= 18ms

2. Shortest Remaining Job First:

Gant chart:

P1 P2 P3 P2 P4 P5

No. of context switches = 5

Average waiting time = 6.6 ms

Average turnaround time = 13.2ms

3. Proposed Algorithm

Gantt chart:

P1 P2 P3 P4 P5

No. of context switches = 4

Average waiting time= 6.8 ms

Average turnaround time= 13.4ms

We select five processes with their arrival time and burst time. These processes are scheduled by Round Robin,

Preemptive SJF and proposed algorithm. We calculate and compare average waiting time, contest switching and

turnaround time of each scheduling algorithm. We execute many processes sets and calculate and compare them.

The example is shown below,

Table 1:A set of five processes with arrival time and burst time

Processes Time of Arrival Burst Duration (ms)

P1

P2

P3

P4

P5

0

2

5

6

8

4.0

7.0

5.0

8.0

9.0

1. Round Robin with quantum = 5

Gantt chart:

P1

P2

P3 P4

P5 P2

P4

P5

No. of context switches = 7

Average waiting time= 11.4 ms

Average turnaround time= 18ms

2. Shortest Remaining Job First:

Gant chart:

P1 P2 P3 P2 P4 P5

No. of context switches = 5

Average waiting time = 6.6 ms

Average turnaround time = 13.2ms

3. Proposed Algorithm

Gantt chart:

P1 P2 P3 P4 P5

No. of context switches = 4

Average waiting time= 6.8 ms

Average turnaround time= 13.4ms

We select five processes with their arrival time and burst time. These processes are scheduled by Round Robin,

Preemptive SJF and proposed algorithm. We calculate and compare average waiting time, contest switching and

turnaround time of each scheduling algorithm. We execute many processes sets and calculate and compare them.

The example is shown below,

Table 1:A set of five processes with arrival time and burst time

Processes Time of Arrival Burst Duration (ms)

P1

P2

P3

P4

P5

0

2

5

6

8

4.0

7.0

5.0

8.0

9.0

1. Round Robin with quantum = 5

Gantt chart:

P1

P2

P3 P4

P5 P2

P4

P5

No. of context switches = 7

Average waiting time= 11.4 ms

Average turnaround time= 18ms

2. Shortest Remaining Job First:

Gant chart:

P1 P2 P3 P2 P4 P5

No. of context switches = 5

Average waiting time = 6.6 ms

Average turnaround time = 13.2ms

3. Proposed Algorithm

Gantt chart:

P1 P2 P3 P4 P5

No. of context switches = 4

Average waiting time= 6.8 ms

Average turnaround time= 13.4ms

We select five processes with their arrival time and burst time. These processes are scheduled by Round Robin,

Preemptive SJF and proposed algorithm. We calculate and compare average waiting time, contest switching and

turnaround time of each scheduling algorithm. We execute many processes sets and calculate and compare them.

The example is shown below,

Table 1:A set of five processes with arrival time and burst time

Processes Time of Arrival Burst Duration (ms)

P1

P2

P3

P4

P5

0

2

5

6

8

4.0

7.0

5.0

8.0

9.0

1. Round Robin with quantum = 5

Gantt chart:

P1

P2

P3 P4

P5 P2

P4

P5

No. of context switches = 7

Average waiting time= 11.4 ms

Average turnaround time= 18ms

2. Shortest Remaining Job First:

Gant chart:

P1 P2 P3 P2 P4 P5

No. of context switches = 5

Average waiting time = 6.6 ms

Average turnaround time = 13.2ms

3. Proposed Algorithm

Gantt chart:

P1 P2 P3 P4 P5

No. of context switches = 4

Average waiting time= 6.8 ms

Average turnaround time= 13.4ms

We select five processes with their arrival time and burst time. These processes are scheduled by Round Robin,

Preemptive SJF and proposed algorithm. We calculate and compare average waiting time, contest switching and

turnaround time of each scheduling algorithm. We execute many processes sets and calculate and compare them.

The example is shown below,

Table 1:A set of five processes with arrival time and burst time

Processes Time of Arrival Burst Duration (ms)

P1

P2

P3

P4

P5

0

2

5

6

8

4.0

7.0

5.0

8.0

9.0

1. Round Robin with quantum = 5

Gantt chart:

P1

P2

P3 P4

P5 P2

P4

P5

No. of context switches = 7

Average waiting time= 11.4 ms

Average turnaround time= 18ms

2. Shortest Remaining Job First:

Gant chart:

P1 P2 P3 P2 P4 P5

No. of context switches = 5

Average waiting time = 6.6 ms

Average turnaround time = 13.2ms

3. Proposed Algorithm

Gantt chart:

P1 P2 P3 P4 P5

No. of context switches = 4

Average waiting time= 6.8 ms

Average turnaround time= 13.4ms

We select five processes with their arrival time and burst time. These processes are scheduled by Round Robin,

Preemptive SJF and proposed algorithm. We calculate and compare average waiting time, contest switching and

turnaround time of each scheduling algorithm. We execute many processes sets and calculate and compare them.

The example is shown below,

Table 1:A set of five processes with arrival time and burst time

Processes Time of Arrival Burst Duration (ms)

P1

P2

P3

P4

P5

0

2

5

6

8

4.0

7.0

5.0

8.0

9.0

1. Round Robin with quantum = 5

Gantt chart:

P1

P2

P3 P4

P5 P2

P4

P5

No. of context switches = 7

Average waiting time= 11.4 ms

Average turnaround time= 18ms

2. Shortest Remaining Job First:

Gant chart:

P1 P2 P3 P2 P4 P5

No. of context switches = 5

Average waiting time = 6.6 ms

Average turnaround time = 13.2ms

3. Proposed Algorithm

Gantt chart:

P1 P2 P3 P4 P5

No. of context switches = 4

Average waiting time= 6.8 ms

Average turnaround time= 13.4ms

We select five processes with their arrival time and burst time. These processes are scheduled by Round Robin,

Preemptive SJF and proposed algorithm. We calculate and compare average waiting time, contest switching and

turnaround time of each scheduling algorithm. We execute many processes sets and calculate and compare them.

The example is shown below,

Table 1:A set of five processes with arrival time and burst time

Processes Time of Arrival Burst Duration (ms)

P1

P2

P3

P4

P5

0

2

5

6

8

4.0

7.0

5.0

8.0

9.0

1. Round Robin with quantum = 5

Gantt chart:

P1

P2

P3 P4

P5 P2

P4

P5

No. of context switches = 7

Average waiting time= 11.4 ms

Average turnaround time= 18ms

2. Shortest Remaining Job First:

Gant chart:

P1 P2 P3 P2 P4 P5

No. of context switches = 5

Average waiting time = 6.6 ms

Average turnaround time = 13.2ms

3. Proposed Algorithm

Gantt chart:

P1 P2 P3 P4 P5

No. of context switches = 4

Average waiting time= 6.8 ms

Average turnaround time= 13.4ms

We select five processes with their arrival time and burst time. These processes are scheduled by Round Robin, Preemptive SJF and proposed algorithm. We calculate and compare average waiting time, contest switching and turnaround time of each scheduling algorithm. We execute sets and calculate and compare them.

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Burst Time |
| P1 | 0 | 4 |
| P2 | 2 | 7 |
| P3 | 5 | 5 |
| P4 | 6 | 8 |
| P5 | 8 | 9 |
|  |  |  |

1. Round Robin with quantum=5

Gantt chart:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P5 | P2 | P4 | P5 |

No. of context switches = 7

Average waiting time= 11.4ms

Average turnaround time= 18ms

1. Shortest Remaining Job First:

Gantt chart:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P2 | P4 | P5 |

No. of context switches = 5

Average waiting time = 6.6ms

Average turnaround time = 13.2ms

1. Proposed Algorithm:

Gantt chart:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P5 |

No. of context switches = 4

Average waiting time= 6.8ms

Average turnaround time= 13.4ms

**Code: 18**

#include<stdio.h>

#include<stdlib.h>

int n;

struct process{

int p\_no;

int arrival\_t, burst\_t, ct, wait\_t ,taround\_time,p;

int flag;

}

p\_list[100];

void sorting(){

struct process p;

int i,j;

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

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

if(p\_list[i].arrival\_t > p\_list[j].arrival\_t){

p=p\_list[i];

p\_list[i]=p\_list[j];

p\_list[j]=p;

}

}

}

}

int main(){

int i, t=0, b\_t=0, peak;

int a[10];

float wait\_time=0, taround\_time=0, avg\_w\_t=0, avg\_taround\_time=0;

printf("Enter the no. of processes: ");

scanf("%d",&n);

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

p\_list[i].p\_no = i+1;

printf("\n Enter Details for P%d process:- \n",p\_list[i].p\_no);

printf("Enter Arrival Time: ");

scanf("%d",&p\_list[i].arrival\_t);

printf("Enter Burst Time: ");

scanf("%d",&p\_list[i].burst\_t);

p\_list[i].flag=0;

b\_t = b\_t + p\_list[i].burst\_t;

}

sorting();

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

a[i]=p\_list[i].burst\_t;

}

p\_list[9].burst\_t=9999;

for(t=p\_list[0].arrival\_t; t<=b\_t+1;){

peak=9;

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

if(p\_list[i].arrival\_t <= t && p\_list[i].burst\_t < p\_list[peak].burst\_t && p\_list[i].flag != 1){

peak=i;

}

if(p\_list[peak].burst\_t==0 && p\_list[i].flag!=1){

p\_list[i].flag=1;

p\_list[peak].ct=t;

p\_list[peak].burst\_t=9999;

printf("P%d completes in %d\n",p\_list[i].p\_no,p\_list[peak].ct);

}

}

t++;

(p\_list[peak].burst\_t)--;

}

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

p\_list[i].taround\_time=(p\_list[i].ct)-(p\_list[i].arrival\_t);

avg\_taround\_time=avg\_taround\_time+p\_list[i].taround\_time;

p\_list[i].wait\_t=((p\_list[i].taround\_time)-a[i]);

avg\_w\_t=avg\_w\_t+p\_list[i].wait\_t;

}

printf("PNO \t AT \t CT \t TAT \t WT \n");

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

printf("P%d \t %d \t %d \t %d \t %d \n",p\_list[i].p\_no, p\_list[i].arrival\_t, p\_list[i].ct, p\_list[i].taround\_time, p\_list[i].wait\_t);

}

printf("Average Turnaround Time: %f\t\n",avg\_taround\_time);

printf("Average Waiting Time: %f \t\n",avg\_w\_t);

}

**Description:**

In this problem, we are given that he process which has lowest CPU burst length has given the highest priority and vice-versa. In this problem, we have to take care for the higher priority process that they do not starve due to lower priority processes.

Firstly, the higher priority processes have the right to execute than the lower priority can move further to execute. If the lower priority process is executing and in between higher priority task arrives than that process has to stop executing and the higher priority task will start the execution and the lower priority has to wait. But this problem of starvation can be solved by the process name ageing. It is a technique in which system gradually increases the priority of those processes which are waiting in the system for a long time.

In this problem, the higher priority tasks will be executed first than others. If the higher priority tasks takes a lot of time than lower priority process has to wait for a long duration and it takes lower priority tasks to starvation.

**Algorithm:**

1. Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
2. Each process is assigned a priority. Process with highest priority is to be executed first and so on.
3. Processes with same priority are executed on first come first served basis.
4. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

**Complexity:**

If the problem is unsorted then we have to find the process which is having the highest priority. So this is the worst case and in this scenario the complexity is O(n).

And if the Processes are sorted as in this problem it is given that the lowest burst time process will have the higher priority. In this kind of implementation it reduces the time complexity and that is O(logn).

**Constraints:**

1. The major constraint or problem of priority scheduling is the process of indefinite blocking or starvation. This problem appears when the process is ready to be executed but that process has to wait because another process with higher priority is executing.
2. In case if we are having the processes with the same priority then we have to use the 2nd scheduling algorithm.
3. If the system gets crashes eventually, then all the processes having low priority which are not finished yet also gets lost.

**Boundary Conditions:**

In code I have taken the boundary values for burst time that is 9999 and in problem we are given that there are n number of CPU processing but I have given only 100 number of CPU processing here.

**Test cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Priority |
| P1 | 0 | 11 | 2 |
| P2 | 5 | 28 | 0 |
| P3 | 12 | 2 | 3 |
| P4 | 2 | 10 | 1 |
| P5 | 9 | 16 | 4 |

Gantt chart:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 | P4 | P2 | P5 | P3 |

0 11 21 49 65 67

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Process | Burst Time | Completion Time | Turn around Time | Waiting Time |
| P1 | 11 | 3 | 8 | 0 |
| P2 | 28 | 9 | 3 | 2 |
| P3 | 2 | 11 | 9 | 7 |
| P4 | 10 | 12 | 9 | 8 |
| P5 | 16 | 16 | 12 | 8 |

Average waiting Time= 5.0

Average Turn Around Time = 8.2