**Project Report**



School of Computer Science and Engineering

Lovely Professional University.

Phagwara, Punjab (India).

**CSE316 – OPERATING SYSTEMS**

Submitted to:

Dr. Baljit Singh Saini Sir.

Submitted By:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Registration Number | Roll No | Email-Id | GitHub Link |
| **Motupalli Sai Lohith** | **11801913** | **13** | sailohith1439@gmail.com | [**https://github.com/Motupallisailohith/Motupallisailohith/blob/master/multiple\_level\_queue.c**](https://github.com/Motupallisailohith/Motupallisailohith/blob/master/multiple_level_queue.c) |

**Solution Code:**

#include<stdio.h>

#include<conio.h>

int i,n,j=0,k=0,l=0,x,position,q,q1=0,q2=0,q3=0;

int burst\_time1[30],burst\_time2[30],burst\_time3[30],process2[30];

int at1[30], at2[30], at3[30];

int bt[30] , priority[30],priority1[30],at[30],t[30],t1,Total=0,total,counter=0;

int p,waiting\_time1[30],turnaround\_time1[30],sum=0,waiting\_time2[30],turnaround\_time2[30];

float avg\_waiting\_time1=0.0,avg\_turnaround\_time1=0.0,avg\_waiting\_time3,avg\_turnaround\_time3,avg\_waiting\_time2,avg\_turnaround\_time2;

void round\_robin()

{

printf("time quantum for queue3 :: 4 \n\n");

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

{

t[i]=burst\_time3[i];

}

printf(" process number\tburst time\t turnaround time\t waiting time\n\n");

x=j;

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

{

if(t[i]>0 && t[i]<=4)

{

printf("process%d of queue3 is running for %d units \n",i+1,t[i]);

total = total + t[i];

t[i]=0;

counter=1;

}

else if(t[i]>0)

{

printf("process%d of queue 3 is running for 4 units\n",i+1);

t[i]=t[i]-4;

total=total+4;

}

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

{

x--;

printf("process%d\t%d\t%d\t%d \n",i+1,burst\_time3[i],total-at3[i],total-at3[i]-burst\_time3[i]);

avg\_waiting\_time3=avg\_waiting\_time3+total-at3[i]-burst\_time3[i];

avg\_turnaround\_time3=avg\_turnaround\_time3+total-at3[i];

counter = 0;

}

if(i==j-1)

{

i=0;

}

else if(at3[i+1]<=total)

{

i++;

}

else

{

i=0;

}

}

avg\_waiting\_time3=avg\_waiting\_time3/j;

avg\_turnaround\_time3=avg\_turnaround\_time3/j;

printf("\nAverage Waiting Time:%f",avg\_waiting\_time3);

printf("\nAverage Turnaround Time:%f\n",avg\_turnaround\_time3);

}

void priority\_scheduling()

{

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

{

position = i;

for(q=i+1;q<k;q++)

{

if(priority1[q]<priority1[position])

{

position = q;

}

}

t1=priority1[i];

priority1[i]=priority1[position];

priority1[position]=t1;

t1= burst\_time2[i];

burst\_time2[i]=burst\_time2[position];

burst\_time2[position]=t1;

t1=process2[i];

process2[i]=process2[position];

process2[position]=t1;

}

waiting\_time2[0]=0;

for(i=1;i<k;i++)

{

waiting\_time2[i]=0;

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

{

waiting\_time2[i]=waiting\_time2[i]+burst\_time2[j];

}

sum=sum+waiting\_time2[i];

}

avg\_waiting\_time2=sum/k;

sum=0;

printf("\nProcess ID\t\tBurst Time\t Waiting Time\t Turnaround Time\n");

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

{

turnaround\_time2[i]=burst\_time2[i]+waiting\_time2[i];

sum=sum+turnaround\_time2[i];

printf("\nProcess[%d]\t\t%d\t\t %d\t\t %d\n",process2[i],burst\_time2[i],waiting\_time2[i],turnaround\_time2[i]);

}

avg\_turnaround\_time2=sum/k;

printf("\nAverage Waiting Time:\t%f",avg\_waiting\_time2);

printf("\nAverage Turnaround Time:\t%f\n",avg\_turnaround\_time2);

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

{

while(burst\_time2[i]!=0)

{

if(burst\_time2[i]>10)

{

printf("process%d of queue2 is running for 10 units",i+1);

burst\_time2[i]=burst\_time2[i]-10;

}

else if(burst\_time2[i]<=10)

{

printf("process%d of queue2 is running for %d units",i+1,burst\_time2[i]);

burst\_time2[i]=0;

}

}

}

}

void fcfs()

{

waiting\_time1[0] = 0;

for(i=1;i<l;i++)

{

waiting\_time1[i] = 0;

for(p=0;p<l;p++)

{

waiting\_time1[i]=waiting\_time1[i]+burst\_time1[p];

}

}

printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time\n");

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

{

turnaround\_time1[i]=burst\_time1[i]+waiting\_time1[i];

avg\_waiting\_time1=avg\_waiting\_time1+waiting\_time1[i];

avg\_turnaround\_time1=avg\_turnaround\_time1+turnaround\_time1[i];

printf("\nProcess[%d]\t\t%d\t\t%d\t\t%d\n",i+1,burst\_time1[i],waiting\_time1[i],turnaround\_time1[i]);

}

avg\_waiting\_time1=avg\_waiting\_time1/l;

avg\_turnaround\_time1=avg\_turnaround\_time1/l;

printf("\nAverage Waiting Time=%f",avg\_waiting\_time1);

printf("\nAverage Turnaround Time=%f",avg\_turnaround\_time1);

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

{

while(burst\_time1[i]!=0)

{

if(burst\_time1[i]>10)

{

printf("process%d of queue1 is running for 10 units\n",i+1);

burst\_time1[i]=burst\_time1[i]-10;

}

else if(burst\_time1[i]<=10)

{

printf("process%d of queue2 is running for %d units \n",i+1,burst\_time1[i]);

burst\_time1[i]=0;

}

}

}

}

void round\_robin\_whole()

{

printf("time quantum for 3 queues is 10 \n");

for(i=1;i<Total;i = i+10)

{

if(q1>10)

{

printf("queue1 is running for 10 seconds\n");

q1=q1-10;

}

else if(q1<=10 && q1!=0)

{

printf("queue1 is running for %d units \n",q1);

q1=0;

}

if(q2>10)

{

printf("queue2 is running for 10 seconds\n");

q2=q2-10;

}

else if(q2<=10 && q2!=0)

{

printf("queue2 is running for %d units \n",q2);

q2=0;

}

if(q3>10)

{

printf("queue3 is running for 10 seconds \n");

q3=q3-10;

}

else if(q3<=10 && q3!=0)

{

printf("queue3 is running for %d seconds \n",q3);

q3=0;

}

}

}

int main()

{

printf("enter number of processes\n");

scanf("%d",&n);

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

{

printf(" burst time and priority of all process %d \n",i+1);

printf("enter burst time of process \n ");

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

printf("enter priority of processe \n ");

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

printf("enter arrival time of process \n");

scanf("%d",&at[i]);

Total = Total +bt[i];

}

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

{

if(priority[i]>=1 && priority[i]<=5)

{

printf("process%d belongs to queue1 \n",i+1);

at1[j]=at[i];

burst\_time1[j]=bt[i];

j++;

q1=q1+bt[i];

}

else if(priority[i]>6 && priority[i]<= 10)

{

printf("process%d belongs to queue2 \n",i+1);

at2[k]=at[i];

burst\_time2[k]=bt[i];

priority1[k]=priority[i];

process2[k]=k+1;

k++;

q2=q2+bt[i];

}

else if(priority[i]>11)

{

printf("process%d belongs to queue3 \n\n",i+1);

at3[l]=at[i];

burst\_time3[l]=bt[i];

l++;

q3=q3+bt[i];

}

}

round\_robin\_whole();

round\_robin();

fcfs();

priority\_scheduling();

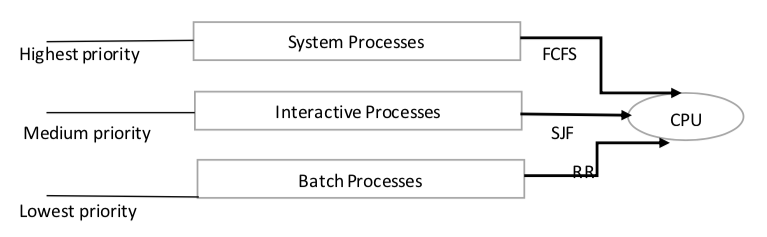
return 0;

}

**Description:**

**The** basic objective of this problem multi level queue scheduling is allotting different set of processes in different queues according to different real life scenarios such that all the queues are scheduled differently using different scheduling algorithms. In the given problem the higher priority processes for example system process in operating system are scheduled according to round robin algorithm with time quantum 4 seconds, medium priority prcosees such as interactive process are scheduled according to priority scheduling and lower priority procees such as batch process are scheduled according to first come first serve basis and as a whole all these queues are scheduled according to round robin algorithm using time quantum 10 seconds and the ordered resulted process enters into running state.

(here we allotted priorites to user prompted processes and they are queued according to their priority and scheduled acoordingly)



**Algorithm:**

**Round Robin:**

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(W.T): Time Difference between turn around time and burst time.  
   Waiting Time = Turn Around Time – Burst Time

* queue that stores processes of high range priority i.e 25,26,27 are considered to be high range priorities is implemented by round robin algorithm (time quantum = 4, mode: premption) .

For example we have three processes implemented using round robin algorithm

1. The execution begins with process P1, which has burst time 4. Here, every process executes for 4 seconds , so P1 completes its execution.
2. At time =4, P2 starts executing which was in ready queue at time = 2 seconds which has burst time of 3 seconds ,i.e 7th to 8th second.
3. Cpu stands idle for 1 second and at 8th second P3 comes to execution it contines with two time slices i.e till 12th second and then 13th second.

**Priority Scheduling:**

1. sort the processes according to arrival time

2. if arrival time is same the acc to priority

3. apply fcfs

* queue that stores processes of medium range priority is implemented by priority scheduling in premptive mode i.e the process which has higher priority in ready queue is executed first by cpu

For example we have three processes implemented using priority scheduling i.e of priorities 6,7,8

1. P1 has priority 5 and burst time 4 seconds P2 has priority and arrives at 4th second 15 P3 has priority 10 and arrives at 2th second
2. At time 0, P1 is running and ends at 4th second
3. At time 4 th second P2,P3 are in ready queue where P2 enters running state because it has higher priority than P3 then P3 comes into running state after completion of P2.

**First come first serve:**

1- Input the processes along with their burst time(b\_t)

and arrival time(a\_t)

2- Find waiting time for all other processes i.e. for

a given process i:

wt\_[i] = (b\_t[0] + b\_t[1] +...... b\_t[i-1]) – a\_t[i]

3- Now find turn around time

= waiting\_time + burst\_time for all processes

4- Average waiting time =

total\_waiting\_time / no\_of\_processes

5- Average turn around time =

total\_turn\_around\_time / no\_of\_processes

* Queue that stores processes of low range priority is implemented by first come first serve basis algorithm in premption mode i.e priorities 2,3,4

For example we have three processes implemented using first come first serve based algorithm.

1. P1 starts running at time 0 till 2 seconds , P2 was in ready queue at time 1 second
2. P2 starts running at time 2 seconds, and completes and prempts at 4th second , P3 arrives at 5th second so Cpu remains idle for 1 second and then P3 starts running.

* All three Queues are now again implemented using round robin algorithm of time quantum 10 seconds.

Example scenario

1. If we have three queues Q1,Q2,Q3 of burst time 20 seconds ,10 seconds and 30 seconds.
2. Q1 arrived at 0th second and runs for 10 seconds, and prempts and enters ready state
3. Next process Q2 in the ready state runs for 10 seconds and terminates .
4. Next Process Q3 in the ready state runs for 10 seconds and terminates
5. Q1 which is in ready state runs for 10 seconds and totally terminates
6. Q2 now runs in two time slices of each 10 seconds.

**Complexity:**

We basically used 4 functions in the programme. So the space complexity for the number of lines of code in each function is as follows

**FUNCTION NLOC COMPLEXITY TOKEN**

round\_robin 50 10 297

Priority\_scheduling 61 11 389

fcfs 40 08 258

round\_robin\_whole 37 11 157

main 50 08 324

total complexity 48

**Boundary Conditions:**

**if(priority[i]>=1 && priority[i]<=5):**

Low range priority that takes values from 1 to 5.

**if(priority[i]>=6 && priority[i]<=10):**

Medium range priority that takes values from 6 to 10

**if(priority[i]>=11):**

High range priority that takes values from 11 and above.

**Test cases:**

1. Enter the no. of processes :-

User needs to enter the number of processes.

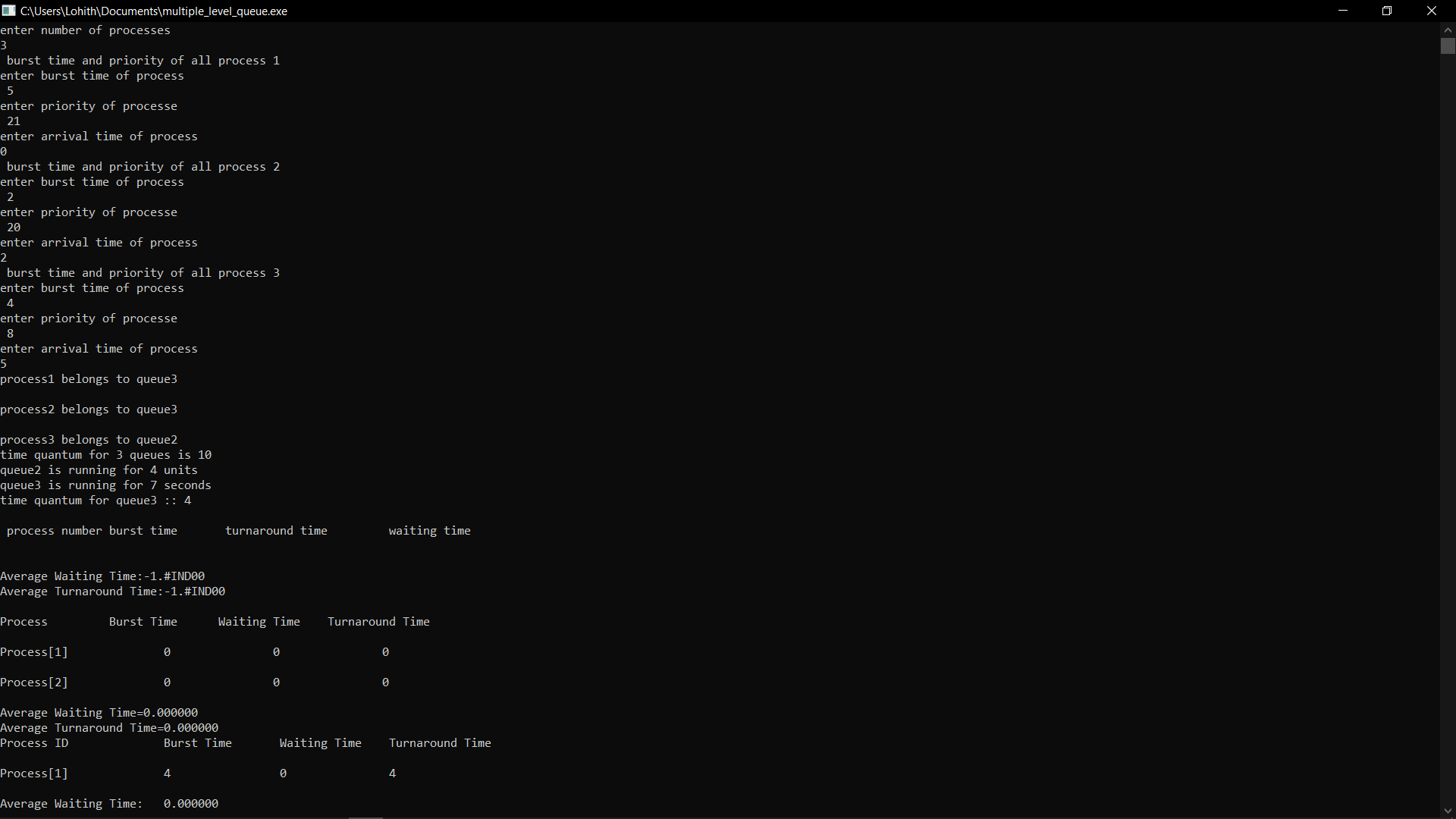
1. Enter Arrival Time and Burst Time of each processes :-

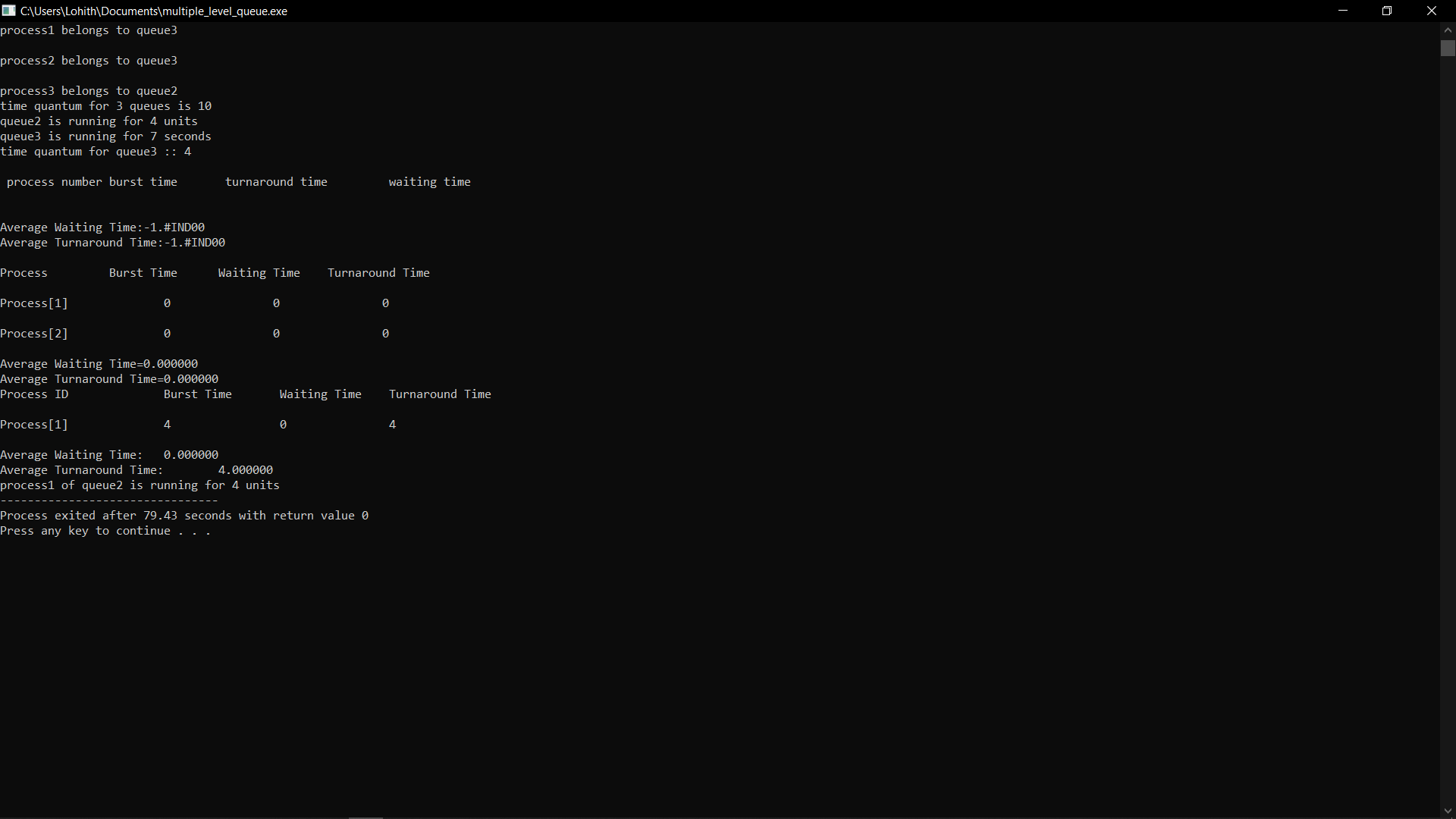
User needs to enter the arrival time and burst time of each processes.

1. Output :-

The starting time and turnaround time of each processes are calculated and they are arranged according to their priority and displayed

**Snapshots:**

****



**All the revisons**

**and information is uploaded in the following github repository**

**Link:** [**https://github.com/Motupallisailohith/Motupallisailohith/blob/master/multiple\_level\_queue.c**](https://github.com/Motupallisailohith/Motupallisailohith/blob/master/multiple_level_queue.c)