**CSE-316**

**CA-ASSIGNMENT**

**Student Name: SAHAJ SHUKLA**

**Student ID: 11806155**

**Email Address: sahajshukla912@gmail.com**

**GitHub Link:** <https://github.com/sahajshukla912/sh/blob/master/mlq%20project.docx>

https://github.com/sahajshukla912/sh

**Code:**

#include<stdio.h>

int arrival\_time1[30],arrival\_time2[30],priority2[30],process2[30],arrival\_time3[30];

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

int Total=0,t1=0,t2=0,t3=0;

int n,i,at[30],bt[30],pr[30],j=0,k=0,l=0;

int total,x,temp[30],counter=0;

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

int p,waiting\_time3[30],turnaround\_time3[30];

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

int position,q,temp1,sum=0,waiting\_time2[30],turnaround\_time2[30];

float avg\_waiting\_time2,avg\_turnaround\_time2;

void round\_robin()

{

printf("Time Quantum for Queue1 is 4\n");

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

{

temp[i]=burst\_time1[i];

}

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

x=j;

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

{

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

{

printf("\nProcess[%d] of Queue1 is running for %d units",i+1,temp[i]);

total=total+temp[i];

temp[i]=0;

counter=1;

}

else if(temp[i]>0)

{

printf("\nProcess[%d] of Queue1 is running for 4 units",i+1);

temp[i]=temp[i]-4;

total=total+4;

}

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

{

x--;

printf("\nProcess[%d]\t%d\t%d\t%d",i+1,burst\_time1[i],total-arrival\_time1[i],total-arrival\_time1[i]-burst\_time1[i]);

avg\_waiting\_time1=avg\_waiting\_time1+total-arrival\_time1[i]-burst\_time1[i];

avg\_turnaround\_time1=avg\_turnaround\_time1+total-arrival\_time1[i];

counter = 0;

}

if(i==j-1)

{

i=0;

}

else if(arrival\_time1[i+1]<=total)

{

i++;

}

else

{

i=0;

}

}

avg\_waiting\_time1=avg\_waiting\_time1/j;

avg\_turnaround\_time1=avg\_turnaround\_time1/j;

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

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

}

void priority()

{

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

{

position=i;

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

{

if(priority2[q]<priority2[position])

{

position=q;

}

}

temp1=priority2[i];

priority2[i]=priority2[position];

priority2[position]=temp1;

temp1=burst\_time2[i];

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

burst\_time2[position]=temp1;

temp1=process2[i];

process2[i]=process2[position];

process2[position]=temp1;

}

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("\nProcess[%d] of Queue2 is running for 10 units",i+1);

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

}

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

{

printf("\nProcess[%d] of Queue2 is running for %d units",i+1,burst\_time2[i]);

burst\_time2[i]=0;

}

}

}

}

void fcfs()

{

waiting\_time3[0] = 0;

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

{

waiting\_time3[i] = 0;

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

{

waiting\_time3[i]=waiting\_time3[i]+burst\_time3[p];

}

}

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

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

{

turnaround\_time3[i]=burst\_time3[i]+waiting\_time3[i];

avg\_waiting\_time3=avg\_waiting\_time3+waiting\_time3[i];

avg\_turnaround\_time3=avg\_turnaround\_time3+turnaround\_time3[i];

printf("\nProcess[%d]\t\t%d\t\t%d\t\t%d\n",i+1,burst\_time3[i],waiting\_time3[i],turnaround\_time3[i]);

}

avg\_waiting\_time3=avg\_waiting\_time3/l;

avg\_turnaround\_time3=avg\_turnaround\_time3/l;

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

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

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

{

while(burst\_time3[i]!=0)

{

if(burst\_time3[i]>10)

{

printf("\nProcess[%d] of Queue3 is running for 10 units",i+1);

burst\_time3[i]=burst\_time3[i]-10;

}

else if(burst\_time3[i]<=10)

{

printf("\nProcess[%d] of Queue2 is running for %d units",i+1,burst\_time3[i]);

burst\_time3[i]=0;

}

}

}

}

void round\_robin1()

{

printf("Time Quantum between the 3 queues is 10\n");

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

{

if(t1>10)

{

printf("Queue1 is running for 10 units\n");

t1=t1-10;

}

else if(t1<=10&&t1!=0)

{

printf("Queue1 is running for %d units\n",t1);

t1=0;

}

if(t2>10)

{

printf("Queue2 is running for 10 units\n");

t2=t2-10;

}

else if(t2<=10&&t2!=0)

{

printf("Queue2 is running for %d units\n",t2);

t2=0;

}

if(t3>10)

{

printf("Queue3 is running for 10 units\n");

t3=t3-10;

}

else if(t3<=10&&t3!=0)

{

printf("Queue3 is running for %d units\n",t3);

t3=0;

}

}

}

int main()

{

printf("Enter the no. of process you want to enter\n");

scanf("%d",&n);

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

{

printf("Enter details of process[%d]\n",i+1);

printf("Arrival Time:");

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

printf("Burst Time:");

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

printf("Priority(1 to 15):");

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

Total=Total+bt[i];

}

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

{

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

{

printf("\n\nProcess[%d] belongs to Queue 1\n",i+1);

arrival\_time1[j]=at[i];

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

j++;

t1=t1+bt[i];

}

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

{

printf("Process[%d] belongs to Queue 2\n",i+1);

arrival\_time2[k]=at[i];

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

priority2[k]=pr[i];

process2[k]=k+1;

k++;

t2=t2+bt[i];

}

else if(pr[i]>=11&&pr[i]<=15)

{

printf("Process[%d] belongs to Queue 3\n\n\n\n",i+1);

arrival\_time3[l]=at[i];

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

l++;

t3=t3+bt[i];

}

}

round\_robin1();

round\_robin();

fcfs();

priority();

return 0;

}

1. **Explain the problem in terms of operating system concept? (Max 200 word)**

**Description:**

Write a program for multilevel queue scheduling algorithm. There must be three queues generated. There must be specific range of priority associated with every queue. Now prompt the user to enter number of processes along with their priority and burst time. Each process must occupy the respective queue with specific priority range according to its priority. Apply Round robin algorithm with quantum time 4 on queue with highest priority range. Apply priority scheduling algorithm on the queue with medium range of priority and First come first serve algorithm on the queue with lowest range of priority. Each and every queue should get a quantum time of 10 seconds. C.P.U will keep on shifting between queues after every 10 seconds i.e. to apply round robin algorithm OF 10 seconds on over all structure.

Calculate Waiting time and turnaround time for every process. The input for number of processes should be given by the user.

Multilevel Queue Scheduling is used where the processes are divided into groups based on property like process type, CPU time, IO access, memory size etc. In a multi-level queue scheduling algorithm there will be ‘n’ no. of queues, where ‘n’ is the no. of groups the processes are classified into. Each queue will be assigned a priority and will have its own scheduling algorithm like FCFS, Round-Robin, SJF, Priority scheduling algorithm. It is to schedule the processes by CPU. With the help of these scheduling algorithm CPU is efficiently schedule the process and decide the order to execute. So that higher priority will execute first. Scheduling algorithm have more than one ready queue generated having different priorities. One running queue is generated.

WE HAVE GIVEN IN THIS QUESTION:

We have to generate 3 queues. In which first one is having higher priority 2nd one is having less than first queue priority and the third queue having very less or low priority.

Round robin having the time quantum of 4 means in this queue there will be context switching in every 4 second or millisecond and also there is 10 units of context switching between the queues.

1. Round-robin scheduling (time quantum=4) Queue 1
2. Priority-Queue scheduling algorithm Queue 2
3. First Come First Serve Queue 3

**Proof of correctness of algorithm:**

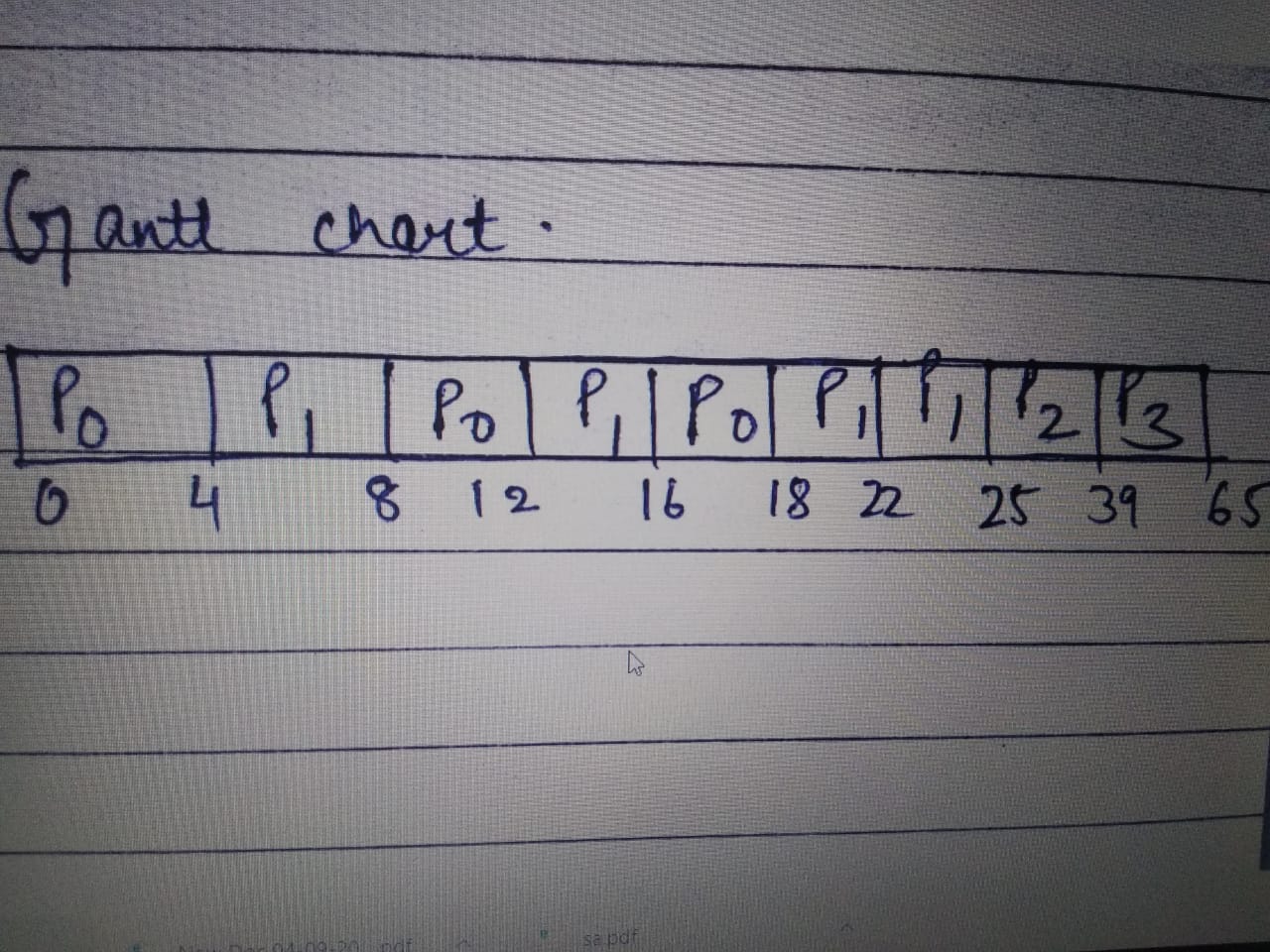
**Algorithm:**

* No. of Processes , Burst time and priority of each process is taken from user .
* Based on the priority the all the process is moved into 3 queues.
* The process with high priority (1) is moved to queue 1 and all the process with low priority are moved to queue 2 and same as in queue 3 .
* First come First serve (FCFS) algorithm is performed on queue 1. If all the process in queue are completed, then the execution will move to queue 2.
* For Round- Robin in queue 3 having time quantum =4 execute it till 10 units and shift to another queue i.e., Priority Queue. After 10 second send it to the 3rd queue ie.,FCFS algo.
* If no more processes in ready queue then stop and exit else repeat 2.
* After the execution of both the queues is completed then we calculate waiting time and turnaround time of each process.

**Complexity analysis of the code**:

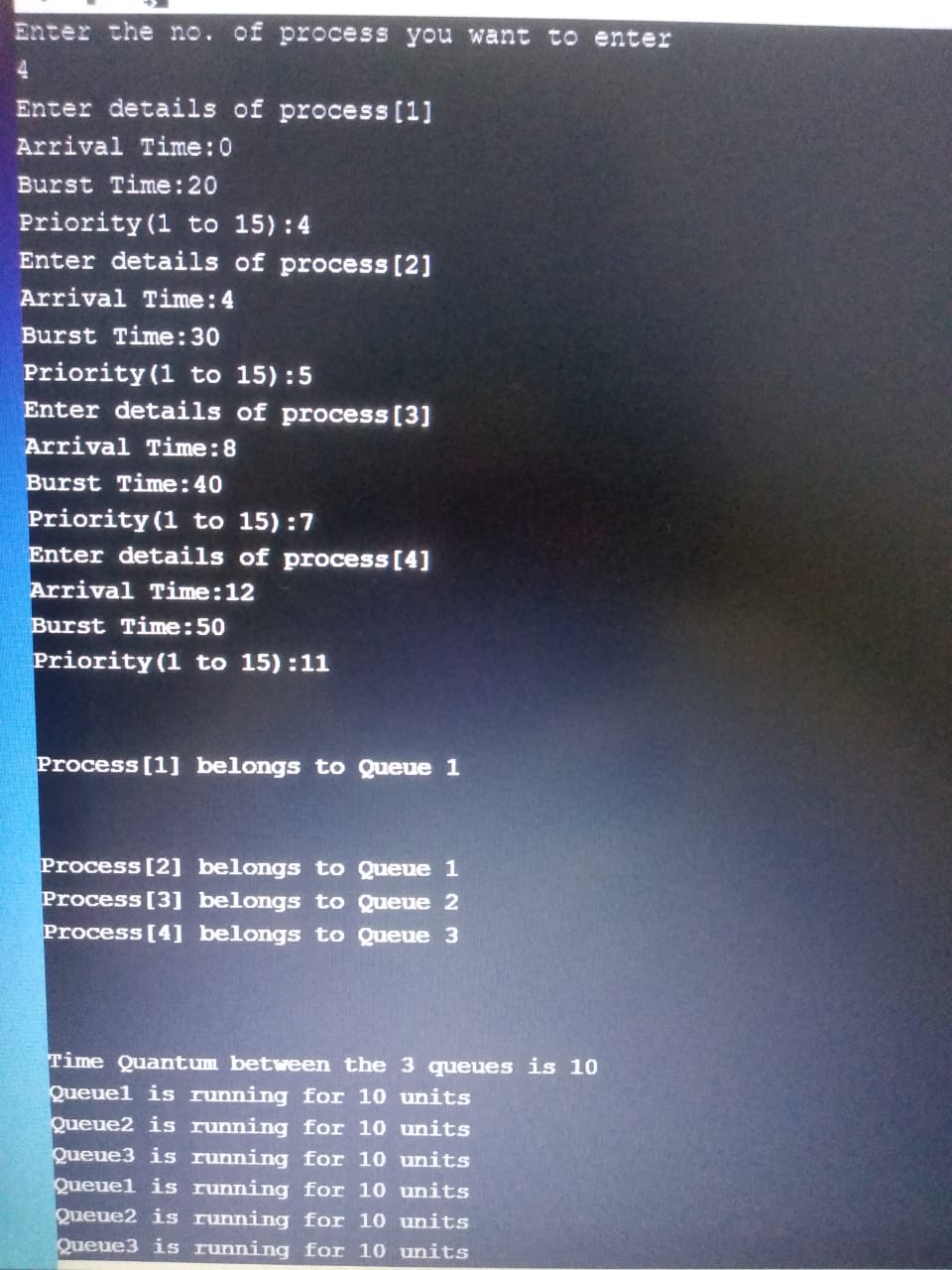
* For every for loop the complexity is calculated in terms of n.
* For every nested for loop the total complexity is calculated in terms of n^m ie., O(n2)
* For every if statement, it is multiplied by 1.
* The complexity of the entire code is (4n (1)) ^2.
* Time taken for code execution: 313 milliseconds= 0.313 seconds (varies from machine to machine)

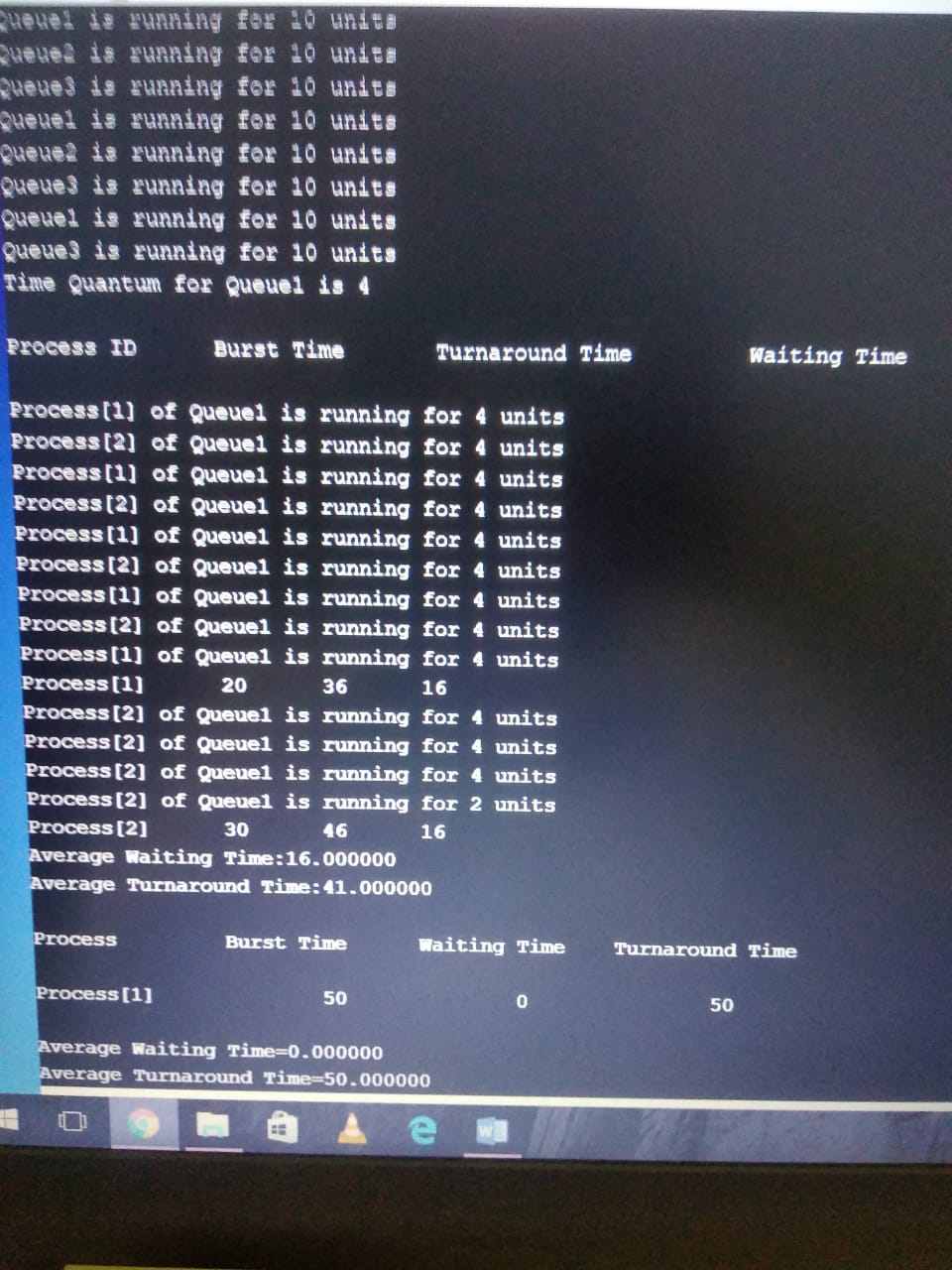
Gantt chart:

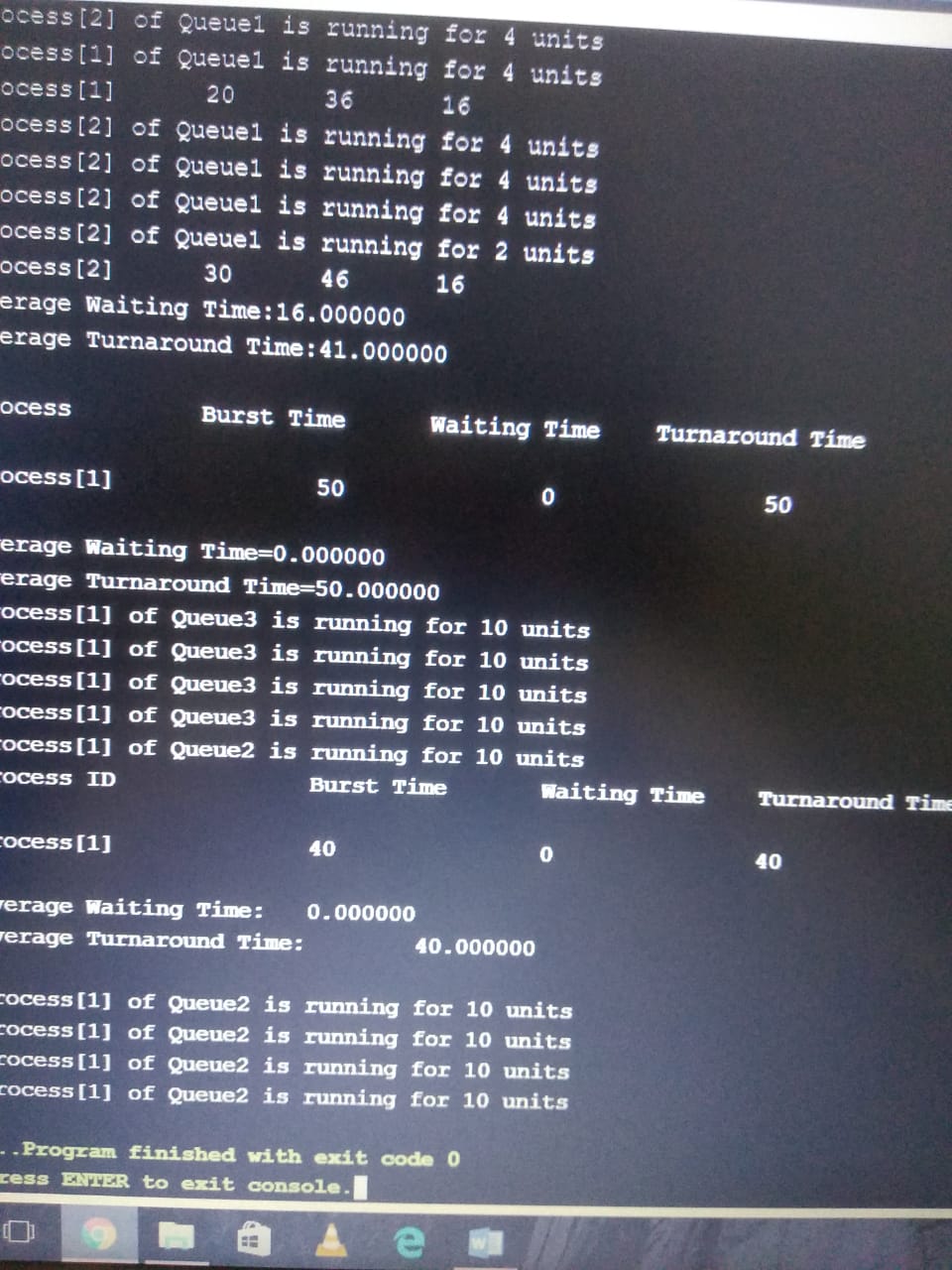


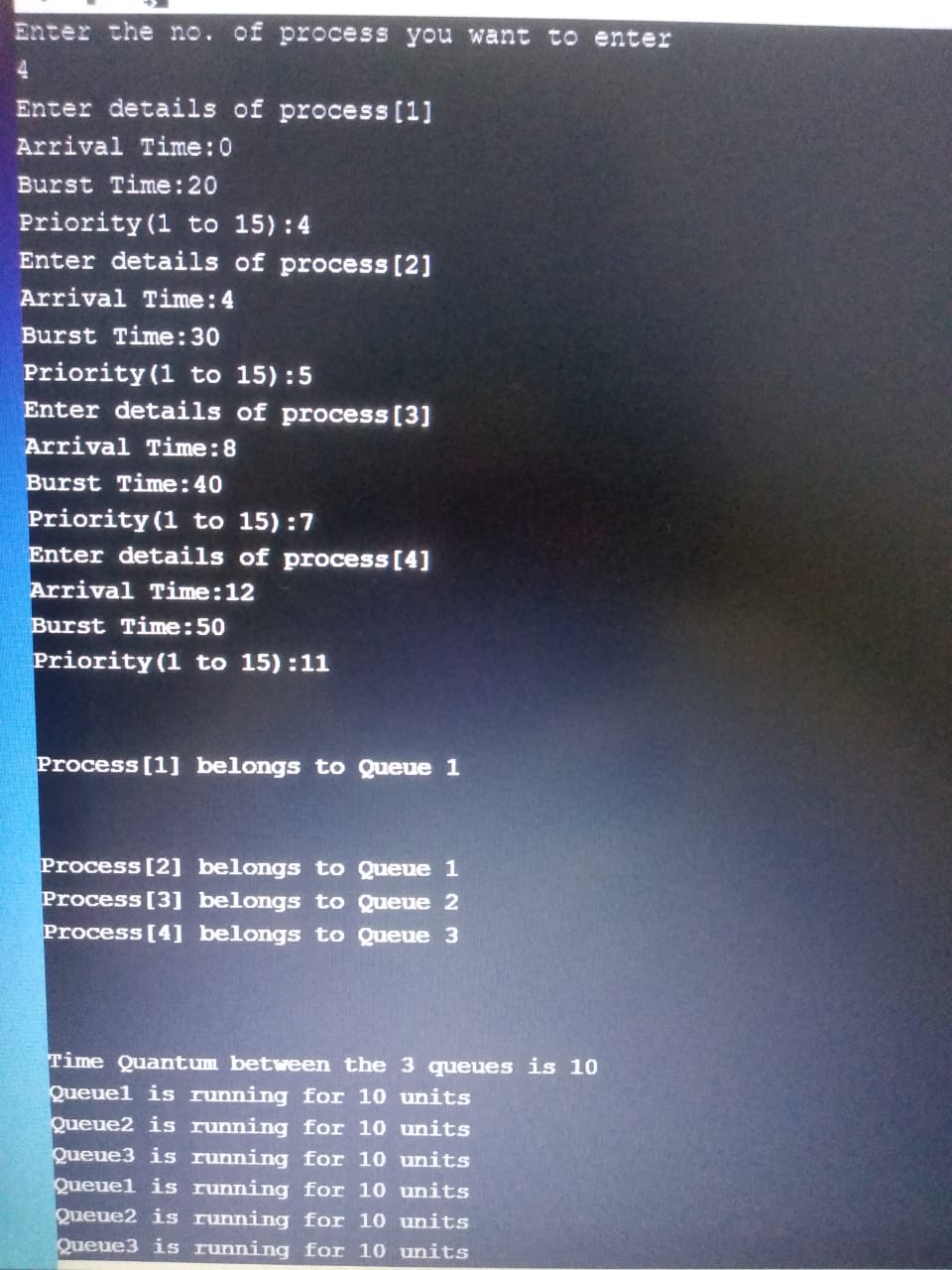
Test cases: 1-

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrival Time | Burst time | Priority |
| 1 | 0 | 20 | 4 |
| 2 | 4 | 30 | 5 |
| 3 | 8 | 40 | 7 |
| 4 | 12 | 50 | 11 |









2 Test-case

