MULTILEVEL QUEUE SCHEDULING(OS)

Name : Nikhil Gupta
Section: K17BG
Roll no: 66.

Question 6:

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. CPU will keep on shifting between queues after every 10 seconds.

Here is the code for my program:

Language: C
Formulas I used in the code:
In P[i]:
Completion for P[i]=Burst Time P[i] + arrival time P[i].
Turnaround Time for P[i]=Burst Time of P[i].
Waiting Time: 0(there will be no waiting Time for the P[i]).
Now for the rest:
Completion Time=completion Time P[i-1]+burst Time P[i].
Turn around Time: Completion Time- Arrival Time;
Waiting Time: Turn Around Time – Burst Time.
Average Waiting Time: addition of the waiting time/no of process.
Average Turn Around Time: addition of the TAT/no of the process.

```
#include <stdio.h>

#include <stdlib.h>

struct process{

        int priority;
```

```
	int bt;

	int pid;

	int wt;

	int tat;

	int rt;

	int at;

	int ct;

};//made a structure so that it would be easy to take the record of process

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

double totalwait=0,totaltat=0,totalCT=0;

void getInput();//to get input

void calcwt(struct process *q,int);//calculate waiting time

void calcCT(struct process *q,int);

void calctat(struct process *q,int);//calculating turnaround

void printQ(struct process *q,int size);//print process details for each queue

void RR();//round robin

void Prior();//priority scheduling

void FCFS();//first come first serve

void printQI(struct process);//print particular process details

void printQ(struct process *,int);//process details for all processes

int q1_size=0,q2_size=0,q3_size=0,n=0; //n=Total Process

struct process *q1,*q2,*q3;

int tq = 4;//time quantum given for RR

void getInput(){

	printf("\n Total Number of Process:\t");
```

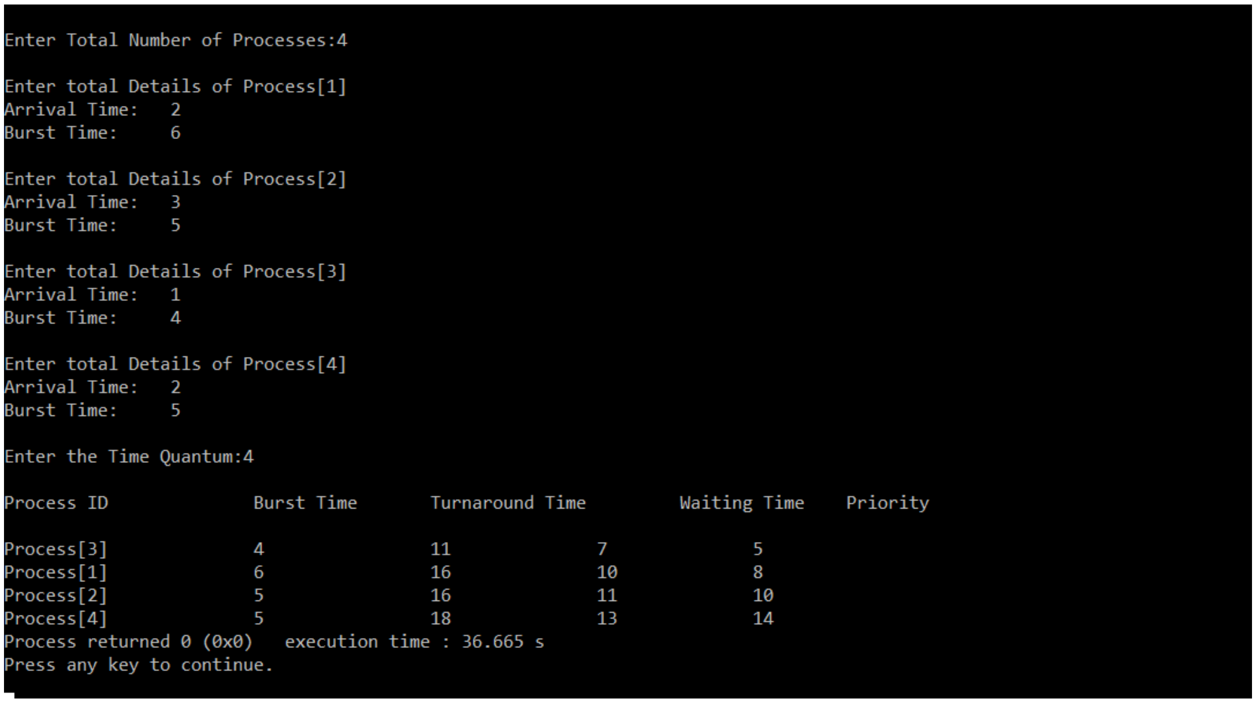
```
        scanf("%d",&n);

        //memory allocation to the queues so that the data of process does not scatter

        q1 = (struct process *)malloc(n*sizeof(struct process));

        q2 = (struct process *)malloc(n*sizeof(struct process));

        q3 = (struct process *)malloc(n*sizeof(struct process));

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

                struct process p;

                printf("\n\t\tProcess
%d\n=============================================\n\n",i+1);



                printf("PId:\t");

                scanf("%d",&p.pid);

                printf("Arrival time:\t");

                scanf("%d",&p.at);

                printf("Priority :\t");

                scanf("%d",&p.priority);

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

                scanf("%d",&p.bt);

                p.rt = p.bt;

    switch(p.priority){

                        case 3:

                                q3[q3_size++] = p;

                                break;

                        case 2:

                                q2[q2_size++] = p;

                                break;
                                case 1:

                                        q1[q1_size++]  = p;

                                        break;

                                default:

         printf("\npriority can only be 1/2/3 nothing else\n");

                        }

                }

        }

void printQ(struct process *q,int size){

                calcCT(q,size);

                calctat(q,size);


                calcwt(q,size);

                        printf("\nPId| Priority| Burst_Time| Waiting_Time| TurnAround_Time| Arrival_Time|
Completion_time|");


printf("\n=================================================================================
===\n");

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

                        printQI(q[i]);

                }

                printf("\n\n");

        }

void printQI(struct process p){

                printf("\n%d\t%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d",p.pid,p.priority,p.bt,p.wt,p.tat,p.at,p.ct);

        }

void calcwt(struct process *q,int size){
        q[0].wt = 0;

        for(int i=1;i<size;i++){

                q[i].wt = q[i].tat - q[i].bt;

                totalwait+=q[i].wt;

        }

        //printf(q.wt);

}

void calcCT(struct process *q,int size){

        q[0].ct = q[0].bt +q[0].at;

        for(int i=1;i<size;i++){

                q[i].ct = q[i-1].ct + q[i].bt;

                totalCT+=q[i].ct;

        }

}

void calctat(struct process *q,int size){

        q[0].tat = q[0].bt;

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

                q[i].tat = q[i].ct - q[i].at;

                totaltat+=q[i].tat;

        }

        //printf("the total turnaround tiq.tat);

}

void FCFSq1(struct process *q,int size){

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

                for(int j=0;j<size;j++){
                    if(q[j].at>q[i].at){

                        struct process t = q[i];

                        q[i] = q[j];

                        q[j] = t;


                    }

                }t3+=q[i].ct;

        }

}

void Priorq2(struct process *q,int size){

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

                for(int j=0;j<size;j++){

                    if(q[j].priority>q[i].priority){

                        struct process t = q[i];

                        q[i] = q[j];

                        q[j] = t;

                    }

                }t2+=q[i].ct;

        }

}

void RRq3(struct process *q,int size){

                int time=0,i=0,remain=size,flag=0,wait_time=0,tat_time=0,total_times=0;

        for(time=0,i=0;remain!=0;){

                struct process p = q[i];

                if(p.rt<=tq && p.rt>0){
                }

                        time += p.rt;

                        p.rt = 0;

                        flag = 1;

                }else if(p.rt>tq){

                        p.rt -= tq;

                        time += tq;

                }

                if(p.rt==0 && flag==1){

                        remain--;

        printf("\n%d\t%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d",p.pid,p.priority,p.bt,p.wt,p.tat,p.at,p.ct);

                        wait_time += time -p.at - p.bt;

                        totalwait+=wait_time;

                        //printf("wait test: %f\n",totalwait);

                        tat_time += time -p.at;

                        totaltat+=tat_time;

                        flag = 0;

                }


                if(i==remain-1){

                        i=0;

                }else if(q[i+1].at<time){

                        i++;

                }else{

                        i=0;

                }
```

```
                    q[i] = p;

            }t1+=q[i].ct;

            //p.wt+=wait_time;

            //p.tat+=tat_time;


}

void FCFS(){

            printf("\n\n======================================================================
======");

            printf("\n\t\tFirst Come First Serve\t");

            printf("\n=======================================================================
====\n\n");

            FCFSq1(q1,q1_size);

            printQ(q1,q1_size);

}

void Prior(){

            printf("\n\n======================================================================
======");

            printf("\n\t\tPriority Scheduling\t");

            printf("\n=======================================================================
====\n\n");

            Priorq2(q2,q2_size);

            printQ(q2,q2_size);

}

void RR(){
            printf("\n\n=====================================================================
======");

            printf("\n\t\tRound Robin\t");

            printf("\n=======================================================================
====\n\n");


            printf("\nPId| Priority| Burst_Time| Waiting_Time| TurnAround_Time| Arrival_Time|
Completion_time|");

            printf("\n=======================================================================
===============\n");

  calcCT(q3,q3_size);


            calctat(q3,q3_size);

            calcwt(q3,q3_size);

            RRq3(q3,q3_size);




}

void round_robin1()

{

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

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

            {

                    if(t1>10)

                    {

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

                            t1=t1-10;
```

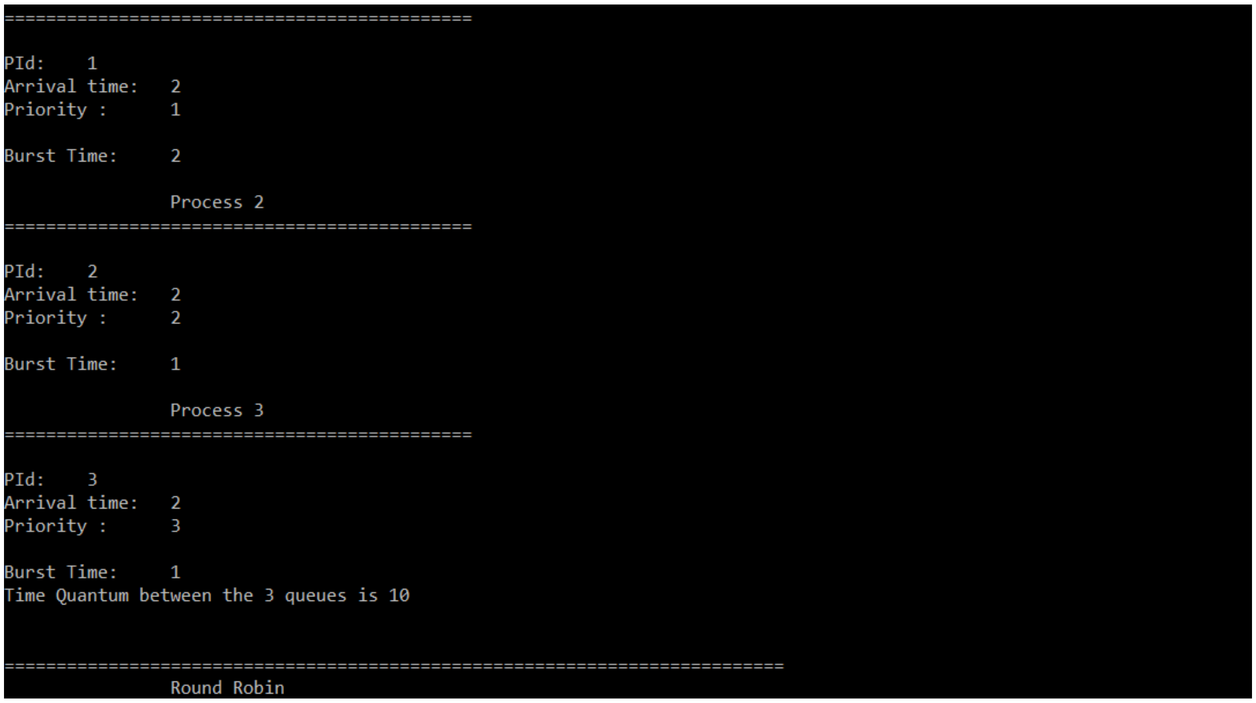
```
            t1=0;

    }
    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(){
            getInput();
            int i=1;
            round_robin1();
            RR();
   Prior();
   FCFS();
            printf("\n\n");
            double p= totalwait/n;
            if(p<0)
       p=0;
            printf("\n Average Waiting Time= %f\n",p);
            printf(" Average Turnaround Time = %f\n",totaltat/n);
}
```

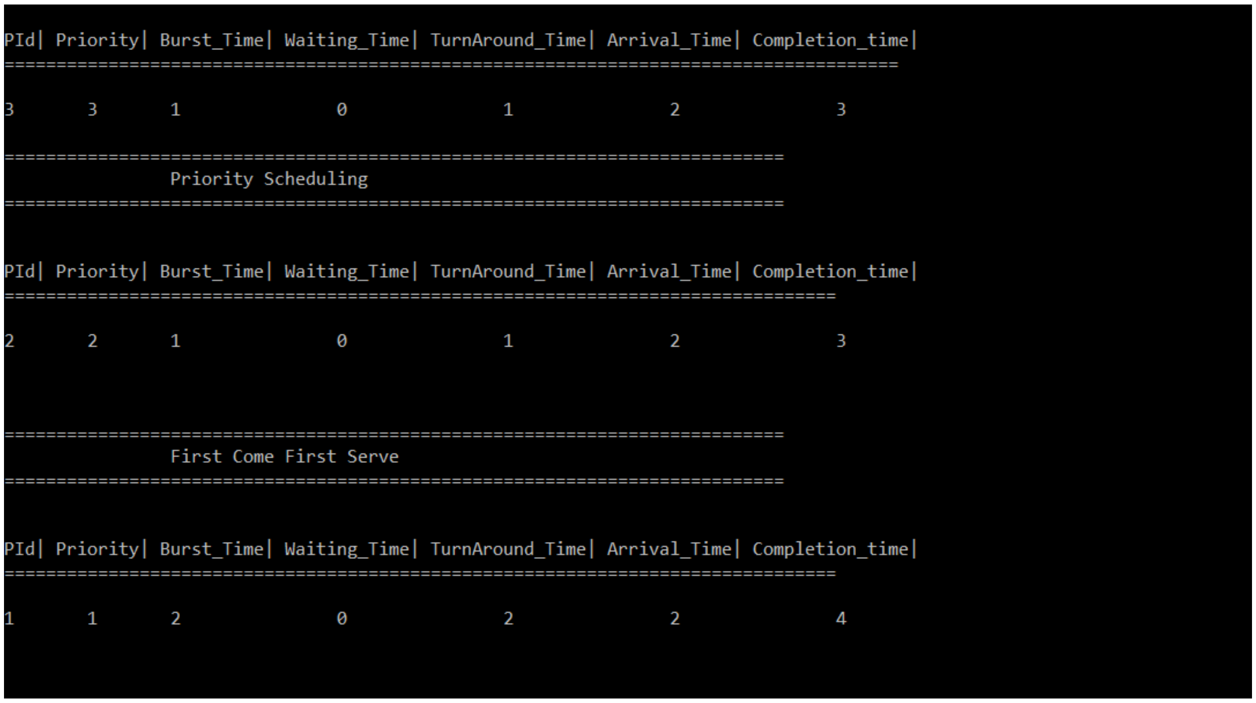
Screen Shot:
This program will ask the user to run the desire no of the process.
I've given P1 P2 P3 for 3 process.



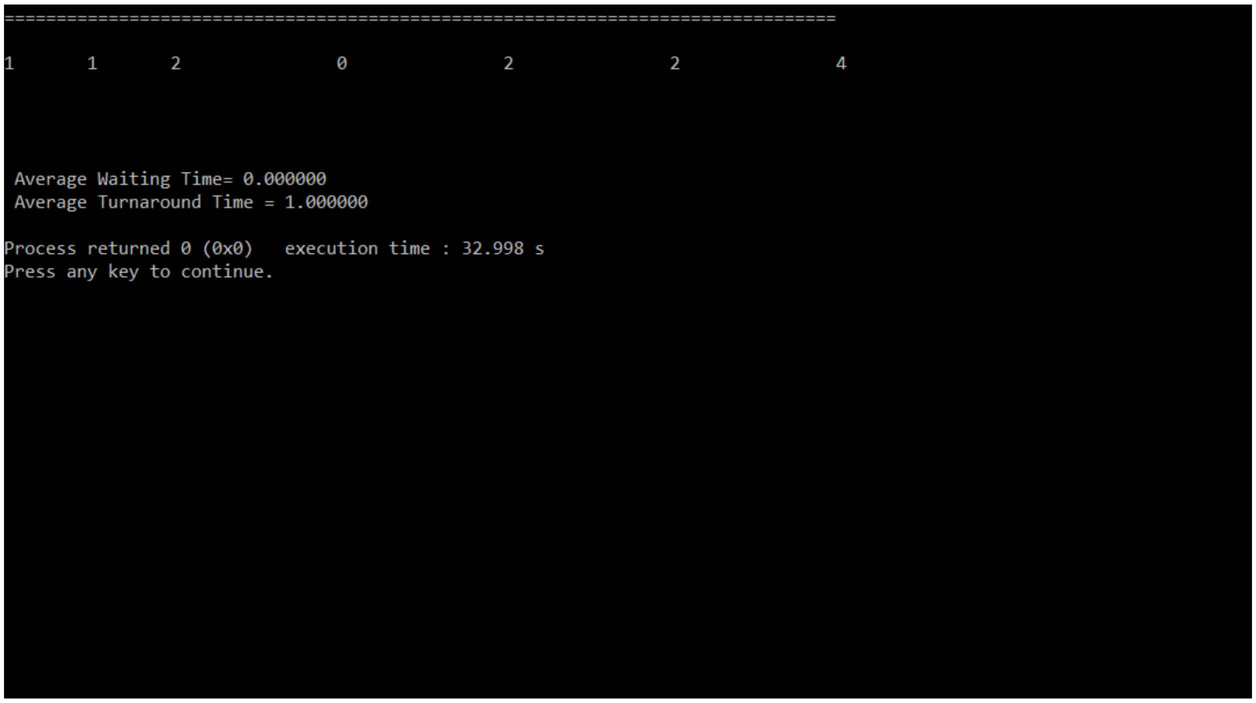
```
Enter Total Number of Processes:4

Enter total Details of Process[1]
Arrival Time:   2
Burst Time:     6

Enter total Details of Process[2]
Arrival Time:   3
Burst Time:     5

Enter total Details of Process[3]
Arrival Time:   1
Burst Time:     4

Enter total Details of Process[4]
Arrival Time:   2
Burst Time:     5

Enter the Time Quantum:4

Process ID              Burst Time       Turnaround Time         Waiting Time    Priority

Process[3]              4                11              7              5
Process[1]              6                16              10             8
Process[2]              5                16              11             10
Process[4]              5                18              13             14
Process returned 0 (0x0)   execution time : 36.665 s
Press any key to continue.
```



```
============================================

PId:    1
Arrival time:   2
Priority :      1

Burst Time:     2

                Process 2
============================================

PId:    2
Arrival time:   2
Priority :      2

Burst Time:     1

                Process 3
============================================

PId:    3
Arrival time:   2
Priority :      3

Burst Time:     1
Time Quantum between the 3 queues is 10


===========================================================================
                Round Robin
```



```
PId| Priority| Burst_Time| Waiting_Time| TurnAround_Time| Arrival_Time| Completion_time|
=======================================================================================

3       3       1               0               1               2               3

===========================================================================
                Priority Scheduling
===========================================================================


PId| Priority| Burst_Time| Waiting_Time| TurnAround_Time| Arrival_Time| Completion_time|
================================================================================

2       2       1               0               1               2               3



===========================================================================
                First Come First Serve
===========================================================================


PId| Priority| Burst_Time| Waiting_Time| TurnAround_Time| Arrival_Time| Completion_time|
================================================================================

1       1       2               0               2               2               4
```



```
================================================================================

1       1       2               0               2               2               4




 Average Waiting Time= 0.000000
 Average Turnaround Time = 1.000000

Process returned 0 (0x0)   execution time : 32.998 s
Press any key to continue.
```

Rememberings:

☹I've tried to use RR(10) for queue scheduling. I've tried to use sleep(10) but it didn't proide the desired.

This program will not allow to enter priority other than 1,2&3 . I've worked till late for this.If unable to run please feel free to check the code and if any errors or improvement needed let me know. ☺