Operating Systems Simulation Based Assignment

Name	**K.S.S Teja Srinivas**
Registration No	**12104702**
Section	**K21CH**
Roll No	**RK21CHB55**
Course Code	**CSE - 316**
Submitted to	**Cherry Khosla mam**
GitHub Link	https://github.com/tejasrinuvas/Os-Simulation
Email	tejasrinivas04@gmail.com

“SCHOOL OF COMPUTER SCIENCE AND ENGINEERING”

LOVELY PROFESSIONAL UNIVERSITY PUNJAB (INDIA)

Phagwara, Punjab.

TABLE OF CONTENTS

6. Snapshots of Output
7. Acknowledgement

Question Assigned:

Q6. 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.

- **Description:**

It may happen that processes in the ready queue can be divided into different classes where each class has its own scheduling needs. For example, a common division is a **foreground (interactive)** process and a **background (batch)** process. These two classes have different scheduling needs. For this kind of situation, **Multilevel Queue Scheduling** is used.
Now, let us see how it works.

Complexity Of Algorithm :

- **Round Robin :** The time complexity of round-robin scheduling algorithms is O(1). It is easy to realize and is suitable to use in high-speed networks.
- **Priority Scheduling Algorithm :** The time and space complexity for NonPre-emptive Priority CPU Scheduling Algorithm :

  Worst case time complexity: $\Theta(n^2)$

  Average case time complexity: $\Theta(n^2)$

  Best case time complexity: $\Theta(n)$
- **First Come First Serve :** The time and space complexity for First Come First Serve Algorithm :

  Worst case time complexity: $\Theta(n^2)$

  Average case time complexity: $\Theta(n^2)$

  Best case time complexity: $\Theta(n)$

Code Snippet of Assigned Question:

```
//ALGORITHM
    //Initiate 3 queues and associate specific range of priority with every queue
    //Enter number of processes along with their priority and burst time.
    //Each process should occupy respective queue
    //Apply Round Robin Algorithm (q=4) with highest priority range
    //Apply Priority Scheduling Algorithm on medium priority range
    //First Come First Serve on lowest priority range
    //Each queue will only get 10 seconds
    //Round Robin on overall structure

    //q1 : p1,p2,p3  |RR(4)    |
    //q2 : p4,p5,p6  |PS    ---| Round Robin (10)
    //q3 : p7,p8,p9  |FCFS     |

#include <iostream>
using namespace std;

struct process{
    int priority;
    int burst_time;
    int tt_time;
    int total_time=0;
};

struct queues{
    int priority_start;
    int priority_end;
    int total_time=0;
    int length = 0;
    process *p;
    bool executed = false;
};

bool notComplete(queues q[]){
    bool a=false;
    int countInc=0;
        for(int i=0;i<3;i++){
                countInc=0;
            for(int j=0;j<q[i].length;j++){
                if(q[i].p[j].burst_time != 0){
                    a=true;
                }
                else{
                    countInc+=1;
                }
            }
            if(countInc==q[i].length){

                q[i].executed = true;
            }
        }
        return a;
}



void sort_ps(queues q){
    //Queue q has to be sorted according to priority of processes
    for(int i=1;i<q.length;i++){
        for(int j=0;j<q.length-1;j++){
            if(q.p[j].priority<q.p[j+1].priority){
                process temp = q.p[j+1];
                q.p[j+1] = q.p[j];
                q.p[j] = temp;
            }
        }
    }
}


void checkCompleteTimer(queues q[]){
    bool a = notComplete(q);
    for(int i=0;i<3;i++){
        if(q[i].executed==false){
            for(int j=0;j<q[i].length;j++){
                if(q[i].p[j].burst_time!=0){
                    q[i].p[j].total_time+=1;
                }
            }
            q[i].total_time+=1;
        }
    }
}

main(){

    //Initializing 3 queues
    queues q[3];
    q[0].priority_start = 7;
    q[0].priority_end = 9;
    q[1].priority_start = 4;
    q[1].priority_end = 6;
    q[2].priority_start = 1;
    q[2].priority_end = 3;

    int no_of_processes,priority_of_process,burst_time_of_process;
    //Prompt User for entering Processes and assigning it to respective queues.
    cout<<"Enter the number of processes\n";
    cin>>no_of_processes;
    process p1[no_of_processes];

    for(int i=0;i<no_of_processes;i++){
        cout<<"Enter the priority of the process\n";
        cin>>priority_of_process;
        cout<<"Enter the burst time of the process\n";
        cin>>burst_time_of_process;
        p1[i].priority = priority_of_process;
        p1[i].burst_time = burst_time_of_process;
        p1[i].tt_time = burst_time_of_process;
        for(int j=0;j<3;j++){
        if(q[j].priority_start<=priority_of_process && priority_of_process<=q[j].priority_end){
            q[j].length++;
        }
        }
    }

    for(int i =0;i<3;i++){
        int len = q[i].length;
        q[i].p = new process[len];
    }


    int a=0;
    int b=0;
    int c=0;

    for(int i =0;i<3;i++){
        for(int j=0;j<no_of_processes;j++){
            if((q[i].priority_start<=p1[j].priority) && (p1[j].priority<=q[i].priority_end)){
                if(i==0){
                    q[i].p[a++] = p1[j];

                    }
                else if(i==1){
                    q[i].p[b++] = p1[j];
                    }
                else{
                    q[i].p[c++] = p1[j];
                    }
            }
        }
    }

    a--;b--;c--;
    for(int i=0;i<3;i++){
        cout<<"Queue "<<i+1<<" : \t";
        for(int j=0;j<q[i].length;j++){
            cout<<q[i].p[j].priority<<"->";
        }
        cout<<"NULL\n";
    }


    //While RR on multiple queues is not complete, keep on repeating
    int timer = 0;
    int l =-1;
    int rr_timer = 4;
    int counter=0;
    int counterps=0;
    int counterfcfs=0;
    while(notComplete(q)){
        if(timer == 10){
            timer = 0;
        }
```

```
        l+=1;
        if(l>=3){
            l=l%3;
        }

        //Process lth queue if its already not executed
        //If its executed change the value of l
        if(q[l].executed == true){
                cout<<"Queue "<<l+1<<" completed\n";
            l+=1;
            if(l>=3){
                l=l%3;
            }
            continue;
        }

        //Finally you now have a queue which is not completely executed
        //Process the incomplete processes over it

        if(l==0){
            cout<<"Queue "<<l+1<<" in hand\n";
            //Round Robin Algorithm for q=4
            if(rr_timer == 0){
                rr_timer = 4;
            }

            for(int i=0;i<q[l].length;i++){
                if(q[l].p[i].burst_time==0){
                    counter++;
                    continue;
                }
                if(counter == q[l].length){
                    break;
                }
                while(rr_timer>0 && q[l].p[i].burst_time!=0 && timer!=10){
                    cout<<"Executing queue 1 and "<<i+1<<" process for a unit time. Process has priority of "<<q[l].p[i].priority<<"\n";
                    q[l].p[i].burst_time--;
                    checkCompleteTimer(q);
                    rr_timer--;
                    timer++;

                }
                if(timer == 10){
                    break;
                }
                if(q[l].p[i].burst_time==0 && rr_timer ==0){
                    rr_timer = 4;
                    if(i == (q[i].length-1)){
                        i=-1;
                    }
                    continue;
                }
                if(q[l].p[i].burst_time==0 && rr_timer > 0){
                    if(i == (q[i].length-1)){
                        i=-1;
                    }
                    continue;
                }
                if(rr_timer <= 0){
                    rr_timer = 4;
                    if(i == (q[i].length-1)){
                        i=-1;
                    }
                    continue;
                }

            }
        }


        else if(l==1){
            cout<<"Queue "<<l+1<<" in hand\n";
            sort_ps(q[l]);
            //Priority Scheduling
            for(int i=0;i<q[l].length;i++){
                if(q[l].p[i].burst_time==0){
                    counterps++;
                    continue;
                }
                if(counterps == q[l].length){
                    break;
                }
                while(q[l].p[i].burst_time!=0 && timer!=10){
                    cout<<"Executing queue 2 and "<<i+1<<" process for a unit time. Process has
priority of "<<q[l].p[i].priority<<"\n";
                    q[l].p[i].burst_time--;
                    checkCompleteTimer(q);
                    timer++;

                }
                if(timer == 10){
                    break;
                }
                if(q[l].p[i].burst_time==0){
                        continue;
                }

            }
        }
        else{
            cout<<"Queue "<<l+1<<" in hand\n";
            //FCFS
            for(int i=0;i<q[l].length;i++){
                if(q[l].p[i].burst_time==0){
                    counterfcfs++;
                    continue;
                }
                if(counterfcfs == q[l].length){
                    break;
                }
                while(q[l].p[i].burst_time!=0 && timer!=10){
                    cout<<"Executing queue 3 and "<<i+1<<" process for a unit time. Process has priority of "<<q[l].p[i].priority<<"\n";
                    q[l].p[i].burst_time--;
                    checkCompleteTimer(q);

                    timer++;
                }
                if(timer == 10){
                    break;
                }
                if(q[l].p[i].burst_time==0){
                        continue;
                }

            }

        }
        cout<<"Broke from queue "<<l+1<<"\n";
    }

    for(int i=0;i<3;i++){
            cout<<"\nTime taken for queue "<<i+1<<" to execute: "<<q[i].total_time<<"\n";
        for(int j=0;j<q[i].length;j++){
            cout<<"Process "<<j+1<<" of queue "<<i+1<<" took "<<q[i].p[j].total_time<<"\n";
        }
    }
    int sum_tt=0;
    int sum_wt=0;

    cout<<"\n\nProcess     | Turn Around Time | Waiting Time\n";
    for(int i=0;i<3;i++){
            cout<<"Queue "<<i+1<<"\n";
        for(int j=0;j<q[i].length;j++){
            cout<<"Process P"<<j+1<<"\t"<<q[i].p[j].total_time<<"\t\t    "<<q[i].p[j].total_time-
q[i].p[j].tt_time<<"\n";
            sum_tt+=q[i].p[j].total_time;
            sum_wt+=q[i].p[j].total_time-q[i].p[j].tt_time;
        }
    }

    cout<<"\n The average turnaround time is : "<<sum_tt/no_of_processes<<endl;
    cout<<"\n The average waiting time is : "<<sum_wt/no_of_processes<<endl;

}
```

○ Characteristics, Disadvantages of Algorithm:

- **Advantages of MLQ algorithm:** With the help of this scheduling, we can apply various kind of scheduling for different kind of processes:

  For System Processes: First Come First Serve(FCFS) Scheduling.
  For Interactive Processes: Shortest Job First (SJF) Scheduling.
  For Batch Processes: Round Robin(RR) Scheduling
  For Student Processes: Priority Scheduling

- **Disadvantages of Multilevel Queue Scheduling:** The main disadvantage of Multilevel Queue Scheduling is the problem of starvation for lower-level processes.

  Starvation: Due to starvation lower-level processes either never execute or have to wait for a long amount of time because of lower priority or higher priority process taking a large amount of time.

- **Advantages of Round Robin Scheduling Algorithm:**

  While performing this scheduling algorithm, a particular time quantum is allocated to different jobs.
  In terms of average response time, this algorithm gives the best performance.

With the help of this algorithm, all the jobs get a fair allocation of CPU.
In this algorithm, there are no issues of starvation or convoy effect.
This algorithm deals with all processes without any priority.
Also, in this, a round-robin scheduler generally employs time-sharing which means providing each job a time slot or quantum.
In this scheduling algorithm, each process gets a chance to reschedule after a particular quantum time.

- **Disadvantages of Round Robin Scheduling Algorithm:**

This algorithm spends more time on context switches.
For small quantum, it is time-consuming scheduling.
This algorithm offers a larger waiting time and response time.
In this, there is low throughput.
If time quantum is less for scheduling, then its Gantt chart seems to be too big.

○ Testcases of Algorithm: Code Compiled

- **Test 1(t1.txt):**

```
Enter the number of processes
4
Enter the priority of the process
3
Enter the burst time of the process
4
Enter the priority of the process
2
Enter the burst time of the process
5
Enter the priority of the process
1
Enter the burst time of the process
7
Enter the priority of the process
4
```

```
Enter the burst time of the process
8
Queue 1 :       NULL
Queue 2 :       4->NULL
Queue 3 :       3->2->1->NULL
Queue 1 completed
Queue 3 in hand
Executing queue 3 and 1 process for a unit time. Process has priority of 3
Executing queue 3 and 1 process for a unit time. Process has priority of 3
Executing queue 3 and 1 process for a unit time. Process has priority of 3
Executing queue 3 and 1 process for a unit time. Process has priority of 3
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 3 process for a unit time. Process has priority of 1
Broke from queue 3
Queue 1 completed
Queue 3 in hand
Executing queue 3 and 3 process for a unit time. Process has priority of 1
Executing queue 3 and 3 process for a unit time. Process has priority of 1
Executing queue 3 and 3 process for a unit time. Process has priority of 1
Executing queue 3 and 3 process for a unit time. Process has priority of 1
Executing queue 3 and 3 process for a unit time. Process has priority of 1
Executing queue 3 and 3 process for a unit time. Process has priority of 1
Broke from queue 3
Queue 1 completed
Queue 3 completed
Queue 2 in hand
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 1 process for a unit time. Process has priority of 4
```

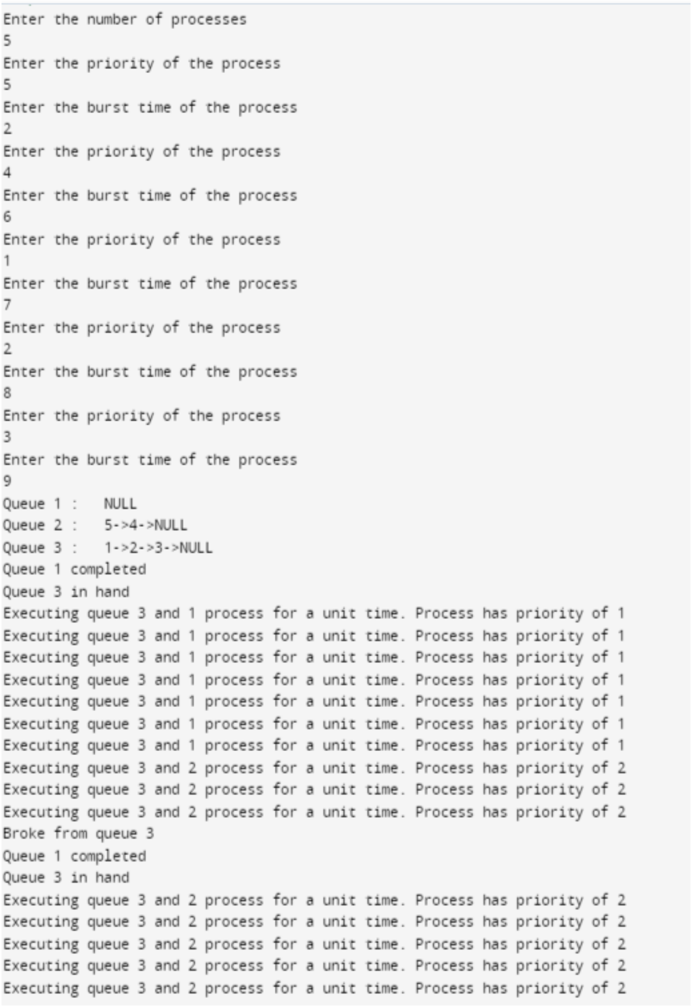
```
Broke from queue 2
Queue 3 completed
Queue 2 in hand
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Broke from queue 2

Time taken for queue 1 to execute: 0

Time taken for queue 2 to execute: 23
Process 1 of queue 2 took 23

Time taken for queue 3 to execute: 15
Process 1 of queue 3 took 3
Process 2 of queue 3 took 8
Process 3 of queue 3 took 15



Process     | Turn Around Time | Waiting Time
Queue 1
Queue 2
Process P1      23                  15
Queue 3
Process P1      3                   -1
Process P2      8                   3
Process P3      15                  8

 The average turnaround time is : 12

 The average waiting time is : 6
```

- **Test 2(t2.txt):**

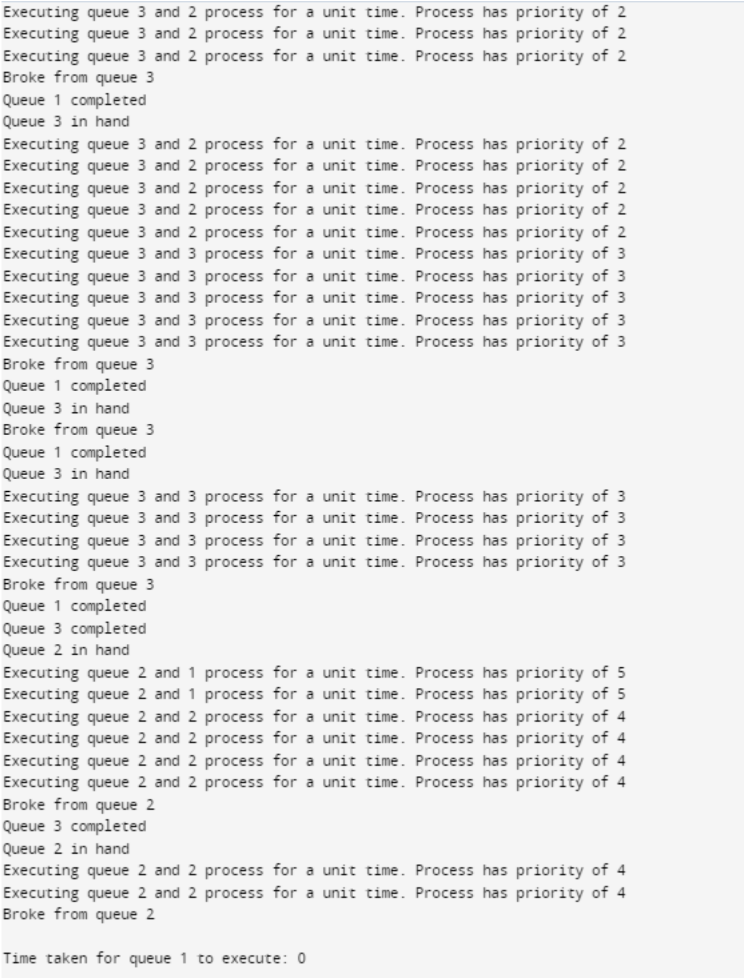
```
Enter the number of processes
5
Enter the priority of the process
5
Enter the burst time of the process
2
Enter the priority of the process
4
Enter the burst time of the process
6
Enter the priority of the process
1
Enter the burst time of the process
7
Enter the priority of the process
2
Enter the burst time of the process
8
Enter the priority of the process
3
Enter the burst time of the process
9
Queue 1 :       NULL
Queue 2 :       5->4->NULL
Queue 3 :       1->2->3->NULL
Queue 1 completed
Queue 3 in hand
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Broke from queue 3
Queue 1 completed
Queue 3 in hand
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Broke from queue 3
Queue 1 completed
Queue 3 in hand
Broke from queue 3
Queue 1 completed
Queue 3 in hand
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Broke from queue 3
Queue 1 completed
Queue 3 completed
Queue 2 in hand
Executing queue 2 and 1 process for a unit time. Process has priority of 5
Executing queue 2 and 1 process for a unit time. Process has priority of 5
Executing queue 2 and 2 process for a unit time. Process has priority of 4
Executing queue 2 and 2 process for a unit time. Process has priority of 4
Executing queue 2 and 2 process for a unit time. Process has priority of 4
Executing queue 2 and 2 process for a unit time. Process has priority of 4
Broke from queue 2
Queue 3 completed
Queue 2 in hand
Executing queue 2 and 2 process for a unit time. Process has priority of 4
Executing queue 2 and 2 process for a unit time. Process has priority of 4
Broke from queue 2

Time taken for queue 1 to execute: 0

Time taken for queue 2 to execute: 31
Process 1 of queue 2 took 25
Process 2 of queue 2 took 31

Time taken for queue 3 to execute: 23
Process 1 of queue 3 took 6
Process 2 of queue 3 took 14
Process 3 of queue 3 took 23



Process     | Turn Around Time | Waiting Time
Queue 1
Queue 2
Process P1      25                 23
Process P2      31                 25
Queue 3
Process P1      6                  -1
Process P2      14                 6
Process P3      23                 14

 The average turnaround time is : 19
```

The average waiting time is : 13

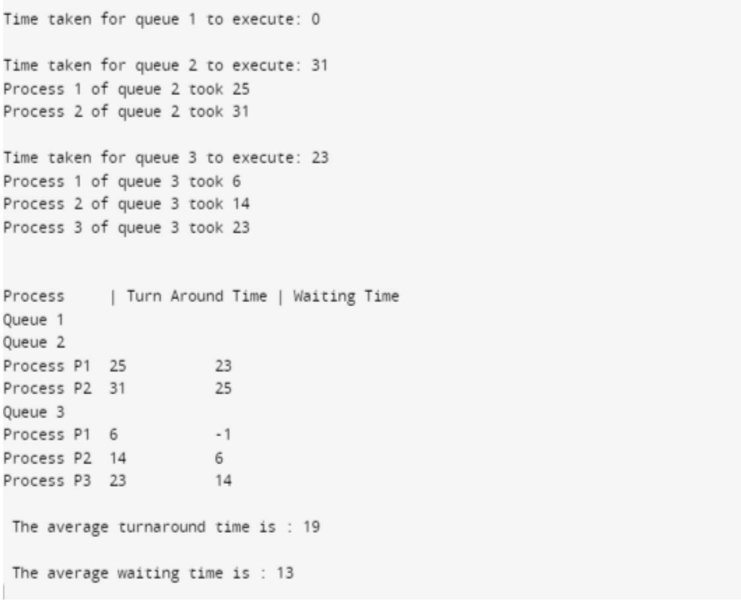
Snapshots of the Output:



```
Enter the number of processes
5
Enter the priority of the process
5
Enter the burst time of the process
2
Enter the priority of the process
4
Enter the burst time of the process
6
Enter the priority of the process
1
Enter the burst time of the process
7
Enter the priority of the process
2
Enter the burst time of the process
8
Enter the priority of the process
3
Enter the burst time of the process
9
Queue 1 :   NULL
Queue 2 :   5->4->NULL
Queue 3 :   1->2->3->NULL
Queue 1 completed
Queue 3 in hand
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 1 process for a unit time. Process has priority of 1
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Broke from queue 3
Queue 1 completed
Queue 3 in hand
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
```



```
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Broke from queue 3
Queue 1 completed
Queue 3 in hand
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Broke from queue 3
Queue 1 completed
Queue 3 in hand
Broke from queue 3
Queue 1 completed
Queue 3 in hand
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Executing queue 3 and 3 process for a unit time. Process has priority of 3
Broke from queue 3
Queue 1 completed
Queue 3 completed
Queue 2 in hand
Executing queue 2 and 1 process for a unit time. Process has priority of 5
Executing queue 2 and 1 process for a unit time. Process has priority of 5
Executing queue 2 and 2 process for a unit time. Process has priority of 4
Executing queue 2 and 2 process for a unit time. Process has priority of 4
Executing queue 2 and 2 process for a unit time. Process has priority of 4
Executing queue 2 and 2 process for a unit time. Process has priority of 4
Broke from queue 2
Queue 3 completed
Queue 2 in hand
Executing queue 2 and 2 process for a unit time. Process has priority of 4
Executing queue 2 and 2 process for a unit time. Process has priority of 4
Broke from queue 2

Time taken for queue 1 to execute: 0
```



```
Time taken for queue 1 to execute: 0

Time taken for queue 2 to execute: 31
Process 1 of queue 2 took 25
Process 2 of queue 2 took 31

Time taken for queue 3 to execute: 23
Process 1 of queue 3 took 6
Process 2 of queue 3 took 14
Process 3 of queue 3 took 23


Process     | Turn Around Time | Waiting Time
Queue 1
Queue 2
Process P1  25          23
Process P2  31          25
Queue 3
Process P1  6           -1
Process P2  14          6
Process P3  23          14

 The average turnaround time is : 19

 The average waiting time is : 13
```

ACKNOWLEDGEMENT:

I would like to express my special thanks to our lectural Cherry Khosla for her time and efforts she provided throughout the Sem. Your useful advice and suggestions were really helpful to me during the project's completion. In this aspect, I am eternally grateful to you.

Thankyou.