SVKM'S NMIMS Deemed-to-be-University
Mukesh Patel School of Technology Management & Engineering
Department of Computer Engineering

Course Code		Program	B.Tech.
Semester	V	Year	II
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

PART A
(PART A : TO BE REFFERED BY STUDENTS)

Experiment No. 05

A.1—Aim:
Study various Process Scheduling Algorithm and implementation of **Round Robin** algorithm for scheduling using 5 Process count.

A.2--- Prerequisite:
Concepts of Process & Process Scheduling

A.3--- Outcome:
After successful completion of this experiment students will be able to:
1. Understand the basics of Process & Process Scheduling.
2. Implement Round Robin Process Scheduling Algorithm

A.4--- Theory:

One of the oldest, simplest, fairest and most widely used algorithm is round robin (RR).

In the round robin scheduling, processes are dispatched in a FIFO manner but are given a limited amount of CPU time called a time-slice or a quantum.

If a process does not complete before its CPU-time expires, the CPU is pre-empted and given to the next process waiting in a queue. The pre-empted process is then placed at the back of the ready list. Round Robin Scheduling is pre-emptive (at the end of time-slice) therefore it is effective in time sharing environments in which the system needs to guarantee reasonable response times for interactive users.

The only interesting issue with round robin scheme is the length of the quantum. Setting the quantum too short causes too many context switches and lower the CPU efficiency. On the other hand, setting the quantum too long may cause poor response time and approximates FCFS.

In any event, the average waiting time under round robin scheduling is often quite long.

A.5--- Procedure:
Task:
1. Study Round Robin Process Scheduling Algorithm
2. Implement Round Robin Algorithm with 5 processes.
3. Save and close the file and name it as **EXP5_ your Roll no.**

SVKM'S NMIMS Deemed-to-be-University
Mukesh Patel School of Technology Management & Engineering
Department of Computer Engineering

Course Code		Program	B.Tech.
Semester	V	Year	II
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

Roll No:B090	Name:Vedant Sahai
Class:B.tech CE	Batch: B2
Date of Experiment	Date of Submission
Grade:	

B.1 Work done by student

```
       package os;

import java.util.*;

public class RoundRobin{
    private static Scanner sc = new Scanner(System.in);
    //Driver Code
    public static void main(String[] args){
        int n,tq, timer = 0, maxProccessIndex = 0;
        float avgWait = 0, avgTT = 0;
        System.out.print("\nEnter the time quanta : ");
        tq = sc.nextInt();
        System.out.print("\nEnter the number of processes : ");
        n = sc.nextInt();
        String pid[] = new String[n];
        int arrival[] = new int[n];
        int burst[] = new int[n];
        int wait[] = new int[n];
        int turn[] = new int[n];
        int queue[] = new int[n];
        int temp_burst[] = new int[n];
        int completiontime[] = new int[n];
        boolean complete[] = new boolean[n];
        for (int i=0;i<n;i++)
        {
            pid[i]= "P"+(i+1);
            System.out.println ("enter process " +(i+1)+ " arrival
time:");
```

SVKM'S NMIMS Deemed-to-be-University
Mukesh Patel School of Technology Management & Engineering
Department of Computer Engineering

Course Code		Program	B.Tech.
Semester	V	Year	II
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

```
            arrival[i]= sc.nextInt();
            System.out.println("enter process " +(i+1)+ " burst
time:");
            burst[i]= sc.nextInt();
            temp_burst[i] = burst[i];
        }

        for(int i = 0; i < n; i++){    //Initializing the queue and
complete array
            complete[i] = false;
            queue[i] = 0;
        }
        while(timer < arrival[0])    //Incrementing Timer until the
first process arrives
            timer++;
        queue[0] = 1;

        while(true){
            boolean flag = true;
            for(int i = 0; i < n; i++){
                if(temp_burst[i] != 0){
                    flag = false;
                    break;
                }
            }
            if(flag)
                break;

            for(int i = 0; (i < n) && (queue[i] != 0); i++){
                int ctr = 0;
                while((ctr < tq) && (temp_burst[queue[0]-1] > 0)){
                    temp_burst[queue[0]-1] -= 1;
                    timer += 1;
                    ctr++;

                    //Updating the ready queue until all the
processes arrive
                    checkNewArrival(timer, arrival, n,
maxProccessIndex, queue);
                }
                if((temp_burst[queue[0]-1] == 0) &&
(complete[queue[0]-1] == false)){
                    turn[queue[0]-1] = timer;        //turn
```

SVKM'S NMIMS Deemed-to-be-University
Mukesh Patel School of Technology Management & Engineering
Department of Computer Engineering

Course Code		Program	B.Tech.
Semester	V	Year	II
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

```
currently stores exit times
                    complete[queue[0]-1] = true;
                }

                //checks whether or not CPU is idle
                boolean idle = true;
                if(queue[n-1] == 0){
                    for(int k = 0; k < n && queue[k] != 0; k++){
                        if(complete[queue[k]-1] == false){
                            idle = false;
                        }
                    }
                }
                else
                    idle = false;

                if(idle){
                    timer++;
                    checkNewArrival(timer, arrival, n,
maxProccessIndex, queue);
                }

                //Maintaining the entries of processes after each
premption in the ready Queue
                queueMaintainence(queue,n);
            }
        }

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

            turn[i] = turn[i] - arrival[i];
            wait[i] = turn[i] - burst[i];
            completiontime[i] = turn[i]+arrival[i];
        }
        System.out.println("\nProcessid Arrival time(AT) burst
time(BT) completion time(CT) turnaround time(TAT) Waiting
time(WT)");
        for(int i = 0 ;i < n;i++)
        {
            System.out.println("\t"+pid[i] +"\t\t\t"+
arrival[i]+"\t\t\t\t"+burst[i]+"\t\t\t\t"+completiontime[i]+"\t\t\t
\t\t"+turn[i]+"\t\t\t\t\t\t"+wait[i]);
        }
```

Course Code		Program	B.Tech.
Semester	V	Year	II
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

```
        System.out.println();
        for(int i =0; i< n; i++){
            avgWait += wait[i];
            avgTT += turn[i];
        }
        System.out.print("\nAverage wait time : "+(avgWait/n)
                +"\nAverage Turn Around Time : "+(avgTT/n));
    }
    public static void queueUpdation(int queue[],int timer,int
arrival[],int n, int maxProccessIndex){
        int zeroIndex = -1;
        for(int i = 0; i < n; i++){
            if(queue[i] == 0){
                zeroIndex = i;
                break;
            }
        }
        if(zeroIndex == -1)
            return;
        queue[zeroIndex] = maxProccessIndex + 1;
    }

    public static void checkNewArrival(int timer, int arrival[],
int n, int maxProccessIndex,int queue[]){
        if(timer <= arrival[n-1]){
            boolean newArrival = false;
            for(int j = (maxProccessIndex+1); j < n; j++){
                if(arrival[j] <= timer){
                    if(maxProccessIndex < j){
                        maxProccessIndex = j;
                        newArrival = true;
                    }
                }
            }
            if(newArrival)    //adds the index of the arriving
process(if any)
                queueUpdation(queue,timer,arrival,n,
maxProccessIndex);
        }
    }

    public static void queueMaintainence(int queue[], int n){
```

SVKM'S NMIMS Deemed-to-be-University
Mukesh Patel School of Technology Management & Engineering
Department of Computer Engineering

Course Code		Program	B.Tech.
Semester	V	Year	II
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

```
        for(int i = 0; (i < n-1) && (queue[i+1] != 0) ; i++){
            int temp = queue[i];
            queue[i] = queue[i+1];
            queue[i+1] = temp;
        }
    }
}
```

```
Enter the time quanta : 2

Enter the number of processes : 5
enter process 1 arrival time:
0
enter process 1 burst time:
5
enter process 2 arrival time:
1
enter process 2 burst time:
3
enter process 3 arrival time:
2
enter process 3 burst time:
1
enter process 4 arrival time:
3
enter process 4 burst time:
2
enter process 5 arrival time:
4
enter process 5 burst time:
3

Processid Arrival time(AT) burst time(BT) completion time(CT) turnaround time(TAT) Waiting time(WT)
    P1          0               5               13                  13                      8
    P2          1               3               12                  11                      8
    P3          2               1               5                   3                       2
    P4          3               2               9                   6                       4
    P5          4               3               14                  10                      7
```

Course Code		Program	B.Tech.
Semester	V	Year	II
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

```
Average wait time : 5.8
Average Turn Around Time : 8.6
Process finished with exit code 0
```

B.2 Conclusion:

Successfully learnt and implemented RoundRobin algorithm.

B.3 Questions of Curiosity:

Q1. Justify "The performance of Round Robin is heavily dependent on the size of the time quantum."

The performance of time slicing policy is heavily dependent on the size/duration of the time quantum. When the time quantum is very large, the Round Robin policy becomes a FCFS policy. Too short quantum causes too many process/context switches and reduces CPU efficiency.

In round robin algorithm, the size of time quanta plays a very important role as:
If size of quanta is too small: Context switches will increase and it is counted as the waste time, so CPU utilization will decrease.
If size of quanta is too large: Larger time quanta will lead to Round robin regenerated into FCFS scheduling algorithm.

Q2. What is the Average Waiting time and Turnaround time for the implemented example?

```
Average wait time : 5.8
Average Turn Around Time : 8.6
Process finished with exit code 0
```