Practical 6

1) Write a java program that implements the RR algorithm.

```
package com.mycompany.rr;
import java.util.Scanner;
```

package com.mycompany.rr;: This declares that the class RR is part of the com.mycompany.rr package.

import java.util.Scanner;: This imports the Scanner class, which is used for reading user input from the console.

```
public class RR {
```

public class RR: This declares a public class named RR.

```
public static void main(String[] args) {
```

public static void main(String[] args): This is the entry point of the program. The main method is where execution begins.

4. **Variable Declarations**

```
int n, i, qt, count = 0, temp, sq = 0, bt[], wt[], tat[], rem_bt[];
float awt = 0, atat = 0;
```

int n, i, qt, count = 0, temp, sq = 0, bt[], wt[], tat[], rem_bt[];:

- n is the number of processes.
- i is used for looping.
- qt is the quantum time (time slice for each process).

- count is used to count the number of completed processes.
- temp is used for temporary storage of remaining burst time.
- sq is used to track the system time.
- bt is an array to store the burst times of processes.
- wt is an array to store the waiting times of processes.
- tat is an array to store the turnaround times of processes.
- rem_bt is an array to store the remaining burst times of processes.

  float awt = 0, atat = 0;:
- awt is the average waiting time.
- atat is the average turnaround time.

5. **Array Initialization**

```
bt = new int[10];
wt = new int[10];
tat = new int[10];
rem_bt = new int[10];
```

Arrays bt, wt, tat, and rem_bt are initialized with a size of 10 to store burst times, waiting times, turnaround times, and remaining burst times respectively.

6. **Input from User**

```
Scanner s = new Scanner(System.in);
System.out.print("Enter the number of process (maximum 10) = ");
n = s.nextInt();
```

```
System.out.print("Enter the burst time of the process\n");
for (i = 0; i < n; i++) {
    System.out.print("P" + i + " = ");
    bt[i] = s.nextInt();
    rem_bt[i] = bt[i];
}
System.out.print("Enter the quantum time: ");
qt = s.nextInt();
```

A Scanner object s is created to read input.

The user is prompted to enter the number of processes n (with a maximum of 10).

The burst times for each process are entered by the user and stored in bt and rem_bt arrays.

The quantum time qt is also entered by the user.

7. **Round Robin Scheduling Algorithm**

```
while (true) {
    for (i = 0, count = 0; i < n; i++) {
        temp = qt;
        if (rem_bt[i] == 0) {
            count++;
            continue;
        }
        if (rem_bt[i] > qt) {
```

```
            rem_bt[i] = rem_bt[i] - qt;
        } else if (rem_bt[i] >= 0) {
            temp = rem_bt[i];
            rem_bt[i] = 0;
        }
        sq = sq + temp;
        tat[i] = sq;
    }
    if (n == count) {
        break;
    }
}
```

The while (true) loop runs until all processes are completed.

For each process, if the remaining burst time rem_bt[i] is 0, it is counted as completed.

If the remaining burst time is more than the quantum time, it is reduced by the quantum time. If it is less, it is set to 0 and temp is set to the remaining burst time.

sq (the system time) is updated by adding temp, and the turnaround time for the process tat[i] is set.

If all processes are completed (count == n), the loop breaks.

8. **Output Results**

```
System.out.print("--------------------------------------------------------------------------
");
```

```
System.out.print("\nProcess\t Burst Time\t Turnaround Time\t Waiting Time\n");
System.out.print("--------------------------------------------------------------------------------");
for (i = 0; i < n; i++) {
    wt[i] = tat[i] - bt[i];
    awt = awt + wt[i];
    atat = atat + tat[i];
    System.out.print("\n " + (i + 1) + "\t " + bt[i] + "\t\t " + tat[i] + "\t\t " + wt[i] + "\n");
}
awt = awt / n;
atat = atat / n;
System.out.println("\nAverage waiting Time = " + awt + "\n");
System.out.println("Average turnaround time = " + atat);
```

The results are printed in a formatted table showing process number, burst time, turnaround time, and waiting time.

Average waiting time and average turnaround time are calculated and printed.

```
package com.mycompany.rr;

import java.util.Scanner;
public class RR {

    public static void main(String[] args) {
        int n,i,qt,count=0,temp,sq=0,bt[],wt[],tat[],rem_bt[];
float awt=0,atat=0;
bt = new int[10];
wt = new int[10];
tat = new int[10];
rem_bt = new int[10];
Scanner s=new Scanner(System.in);
System.out.print("Enter the number of process (maximum 10) = ");
n = s.nextInt();
System.out.print("Enter the burst time of the process\n");
for (i=0;i<n;i++)
{
System.out.print("P"+i+" = ");
bt[i] = s.nextInt();
rem_bt[i] = bt[i];
}
System.out.print("Enter the quantum time: ");
qt = s.nextInt();
while(true)
```

```
{
for (i=0,count=0;i<n;i++)
{
temp = qt;
if(rem_bt[i] == 0)
{
count++;
continue;
}
if(rem_bt[i]>qt)
rem_bt[i]= rem_bt[i] - qt;
else
if(rem_bt[i]>=0)
{
temp = rem_bt[i];
rem_bt[i] = 0;
}
sq = sq + temp;
tat[i] = sq;
}
if(n == count)
break;
}
System.out.print("--------------------------------------------------------------------------------
");
```

```
System.out.print("\nProcess\t Burst Time\t Turnaround Time\t Waiting Time\n");
System.out.print("-----------------------------------------------------------------------------------");
for(i=0;i<n;i++)
{
wt[i]=tat[i]-bt[i];
awt=awt+wt[i];
atat=atat+tat[i];
System.out.print("\n "+(i+1)+"\t "+bt[i]+"\t\t "+tat[i]+"\t\t "+wt[i]+"\n");
}
awt=awt/n;
atat=atat/n;
System.out.println("\nAverage waiting Time = "+awt+"\n");
System.out.println("Average turnaround time = "+atat);
}
}
```