Ex. No.: 6d)
Date

ROUND ROBIN SCHEDULING

Aim:

To implement the Round Robin (RR) scheduling technique

Algorithm:

- 1. Declare the structure and its elements.
- 2. Get number of processes and Time quantum as input from the user.
- 3. Read the process name, arrival time and burst time
- 4. Create an array **rem_bt[]** to keep track of remaining burst time of processes which is initially copy of bt[] (burst times array)
- 5. Create another array $\mathbf{wt}[]$ to store waiting times of processes. Initialize this array as 0. 6. Initialize time: t = 0
- 7. Keep traversing the all processes while all processes are not done. Do following for i'th process if it is not done yet.

```
a- If rem_bt[i] > quantum
```

- (i) t = t + quantum
- (ii) bt rem[i] -= quantum;
- b- Else // Last cycle for this process
- (i) t = t + bt rem[i];
- (ii) wt[i] = t bt[i]
- (iii) bt rem[i] = 0; // This process is over
- 8. Calculate the waiting time and turnaround time for each process.
- 9. Calculate the average waiting time and average turnaround time.
- 10. Display the results.

Program Code:

```
int main()
{
    int i, limit, total = 0, x, counter = 0, time_quantum;
    int wait_time = 0, turnaround_time = 0, arrival_time[10], burst_time[10], temp[10];
    float average_wait_time, average_turnaround_time;
    printf("Enter Total Number of Processes:t");
    scanf("%d", &limit);
    x = limit;
    for(i = 0; i < limit; i++)
    {
        printf("Enter Details of Process[%d]", i + 1);
        printf("Arrival Time:");
        scanf("%d", &arrival_time[i]);
        printf("Burst Time:");
        scanf("%d", &burst_time[i]);
        temp[i] = burst_time[i];
}</pre>
```

```
}
printf("Enter Time Quantum:t");
scanf("%d", &time quantum);
printf("\nProcess ID\t\tBurst Time\t Turnaround\t Time Waiting Time\n");
for(total = 0, i = 0; x != 0;)
   if(temp[i] \le time quantum && temp[i] > 0)
       total = total + temp[i];
       temp[i] = 0;
       counter = 1;
   else if(temp[i] > 0)
       temp[i] = temp[i] - time_quantum;
       total = total + time quantum;
   if(temp[i] == 0 \&\& counter == 1)
       X--;
       printf("Process[%d]\t\t%d\t\t %d\t\t%d\n\n", i + 1, burst time[i], total - arrival time[i],
total - arrival time[i] - burst time[i]);
       wait time = wait time + total - arrival time[i] - burst time[i];
       turnaround time = turnaround time + total - arrival time[i];
       counter = 0;
   if(i == limit - 1)
       i = 0;
   else if(arrival time[i + 1] <= total)
       i++;
   else
       i = 0;
}
average wait time = wait time * 1.0 / limit;
average turnaround time = turnaround time * 1.0 / limit;
printf("\n\nAverage Waiting Time:%f\n", average wait time);
printf("\nAvg Turnaround Time:%f\n", average turnaround time);
return 0;
```

Sample Output:

}

```
C:\WINDOWS\SYSTEM32\cmd.exe
 nter Total Number of Processes:
Arrival Time: 0
Burst Time:
Enter Details of Process[2]
Arrival Time:
Burst Time:
Enter Details of Process[3]
Arrival Time:
Burst Time:
inter Details of Process[4]
Arrival Time:
Burst Time:
Enter Time Quantum:
rocess ID
                        Burst Time
                                         Turnaround Time
                                                                  Waiting Time
rocess[3]
rocess[4]
Average Waiting Time:
                        11.500000
Avg Turnaround Time:
                        17.000000
```

Output:

```
Student@localhost ~|$ cc rr.c
[student@localhost ~|$ ./a.out
Enter Total Number of Processes:t4
Enter Details of Process[]

Arrival Time:0

Burst Time:4
Enter Details of Process[2]

Arrival Time:1

Burst Time:7
Enter Details of Process[3]

Arrival Time:2

Burst Time:5
Enter Details of Process[4]

Arrival Time:3

Burst Time:6
Enter Time Quantum:t3

Process ID

Process[1]

Process[3]

Burst Time

13

Process[4]

Arrival Time:4

Enter Details of Process[4]

Arrival Time:5
Enter Details of Process[4]

Arrival Time:6
Enter Time Quantum:t3

Process ID

Process[1]

Authority Time

11

Process[2]

Average Waiting Time:11.500000

Avg Turnaround Time:17.000000

[student@localhost ~]$
■
```

Result:

Program is executed successfully and output is verified.