Ex. No.: 6d)

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 **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:

```c
#include <stdio.h>

int main() {
    int i, n, tq, bt[10], rt[10], wt[10] = {0}, tat[10] = {0};
    int time = 0, remain, flag = 0;

    printf("Enter total number of processes: ");
    scanf("%d", &n);
    remain = n;

    for(i = 0; i < n; i++) {
        printf("Enter burst time for process %d: ", i + 1);
        scanf("%d", &bt[i]);
        rt[i] = bt[i];
    }

    printf("Enter time quantum: ");
    scanf("%d", &tq);
```

```c
    while(remain != 0) {
        for(i = 0; i < n; i++) {
            if(rt[i] > 0) {
                if(rt[i] <= tq) {
                    time += rt[i];
                    tat[i] = time;
                    wt[i] = tat[i] - bt[i];
                    rt[i] = 0;
                    remain--;
                } else {
                    time += tq;
                    rt[i] -= tq;
                }
            }
        }
    }

    float avg_wt = 0, avg_tat = 0;
    printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for(i = 0; i < n; i++) {
        avg_wt += wt[i];
        avg_tat += tat[i];
        printf("P%d\t%d\t\t%d\t\t%d\n", i + 1, bt[i], wt[i], tat[i]);
    }

    avg_wt /= n;
    avg_tat /= n;

    printf("\nAverage Waiting Time = %.2f", avg_wt);
    printf("\nAverage Turnaround Time = %.2f\n", avg_tat);

    return 0;
}
```

Sample Output:

```
C:\WINDOWS\SYSTEM32\cmd.exe

Enter Total Number of Processes:        4

Enter Details of Process[1]
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:     3

Process ID              Burst Time      Turnaround Time     Waiting Time

Process[1]              4               13                  9
Process[3]              5               16                  11
Process[4]              6               18                  12
Process[2]              7               21                  14

Average Waiting Time:   11.500000
Avg Turnaround Time:    17.000000
```

 Enter total number of processes: 4
Enter burst time for process 1: 5
Enter burst time for process 2: 15
Enter burst time for process 3: 4
Enter burst time for process 4: 3
Enter time quantum: 5

Process	Burst Time	Waiting Time	Turnaround Time
P1	5	0	5
P2	15	18	33
P3	4	5	9
P4	3	9	12

Average Waiting Time = 8.00
Average Turnaround Time = 14.75

Result:

The Round Robin (RR) scheduling technique has been implemented successfully and the output has been verified.