Ex. No.: 6d)
Date 20/3/25 **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:

```
#include <stdio.h>
int main() {
    int n;
    printf("Enter Total number of process:");
    scanf("%d", &n);
    int wait_time=0, ta_time=0, arr_time[n], burst_time[n],
                                    temp=burst_time[n];
    int x=n;
    for(int i=0; i<n; i++) {
        printf("Enter details of process %d\n", i+1);
        printf("Arrival time:");
        scanf("%d", &arr_time[i]);
        printf("Burst time");
```

```
scanf("%d", &burst_time[i]);
temp_burst_time[i] = burst_time[i]; }
int time_slot;
printf("Enter time slot:");
scanf("%d", &time_slot);
int total=0, counter=0, i;
printf("Process ID Burst Time Turnaround Time
                                waiting Time \n");
for(total=0, i=0; x!=0;){
  if(temp_burst_time[i] <= time_slot && temp_burst_
                                time[i]>0){
    total=total+temp_burst_time[i];
    temp_burst_time[i]=0;
    counter=1;}
  else if(temp_burst_time[i]>0){
    temp_burst_time[i]=temp_burst_time[i]-
                        time_slot;
    total_t=time_slot;}
  if(temp_burst_time[i]==0 && counter==1){
      x--;
      printf("\n Process No %d \t\t %d \t\t\t
                        %d \t\t\t %d",
        i+1, burst_time[i], total_arr_time[i],
          total_arr_time[i]-burst_time[i]);
    wait_time=wait_time+total_arr_time[i]-
                        burst_time[i];
    ta_time_t=total_arr_time[i];
    counter=0;}
```

```
if (i==n-1){
    i=0; }
else if (arr_time[i+1] <= total) {
    i++ ; }
else {
    i=0; }
}
float average_wait_time = wait_time * 1.0/n;
float average_turnaround_time = ta_time * 1.0/n;
printf("\n Average waiting time : %f", average_wait_time);
printf("\n Avg Turnaround Time : %f", average_turnaround
                                    _time);
return 0;
}
```

Output:

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 slot : 2

Process Id	Burst time	Turnaround Time	waiting
Process no 1	4	10	6
Process no 2	5	17	12
Process no 3	6	18	12
Process no 4	7	21	14

Average waiting Time : 11.000000
Avg Turnaround Time : 16.500000

Result:

Thus, the Round Robin scheduling technique is achieved sucessfully