Ex. No.: 6d)
Date 21|3|25

ROUND ROBIN SCHEDULING

Aim:

To implement the Round Robin (RR) scheduling technique

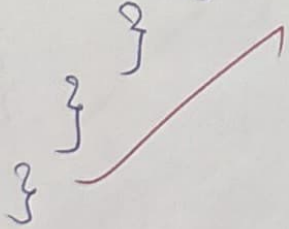
Algorithm:

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, time_quantum;
  printf("Enter no of processes: ");
  scanf("%d", &n);
  int b[n], t[n], p[n], ta[n], w[n], cur[n], c[n];
  printf("Enter arrival time:");
for (int i=0; i<n; i++)
{
  scanf("%d", &ar);
}
```

```
Printf ("Enter time quantum:");
scanf("%d", &time_quantum);
int remaining=n, time=0;
int done[n];
for(int i=0; i<n; i++) done[i]=0;
while (remaining>0)
{
  for(int i=0; i<n; i++)
  { if(t[i]>0) // if
    {
      if(t[i]> time_quantum)
      {
        time+=time_quantum;
        t[i]-= time_quantum;
      }
      else
      {
        time+=t[i];
        C[i]=time;
        t[i]=0;
        done[i]=1;
        remaining--;
      }
  }
}
}
```



```
float sum_ta=0, sum_w=0;
for(int i=0; i<n; i++)
{
    ta[i]=c[i]-arr[i];
    w[i]=ta[i]-b[i];

    sum_ta+=ta[i];
    sum_w+=w[i];
}
float avg_ta = sum_ta/n;
float avg_w = sum_w/n;
printf("Process\t Burst Time\t Arrival Time\t
 Waiting Time\ Turnaround Time\n");
for(int i=0; i<n; i++)
{
  printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p[i], b[i],
  arr[i], w[i], ta[i]);
}
  printf("Avg waiting time is: %.2f\n", avg_w);
  printf("Avg Turnaround Time is: %.2f\n", avg_ta);
  return 0;
}
```

Enter no of processes: 4

Enter arrival time : 0

Enter burst time : 5 4 6 2

Enter quantum time: 2

Process	Burst Time	Arrival Time	Waiting Time	Turnaround Time
1	5	0	10	15
2	4	0	8	12
3	6	0	11	17
4	2	0	6	8

The average waiting Time is: 8.75

The average turn around Time is : 13.00

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
```

Result:

Hence the C program for Round Robin Scheduling is executed and written successfully.