Ex. No.: 6d)
Date 2013|25

ROUND ROBIN SCHEDULING

Aim:

To implement the Round Robin (RR) scheduling technique

Algorithm:

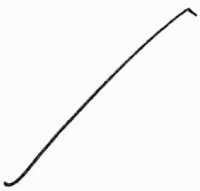
1. Declare the structure and its elements.
2. Get number of processes and Time quantum as input from the user.
3. Read the process name, arrival time and burst time
4. Create an array **rem_bt[]** to keep track of remaining burst time of processes which is initially copy of bt[] (burst times array)
5. 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:

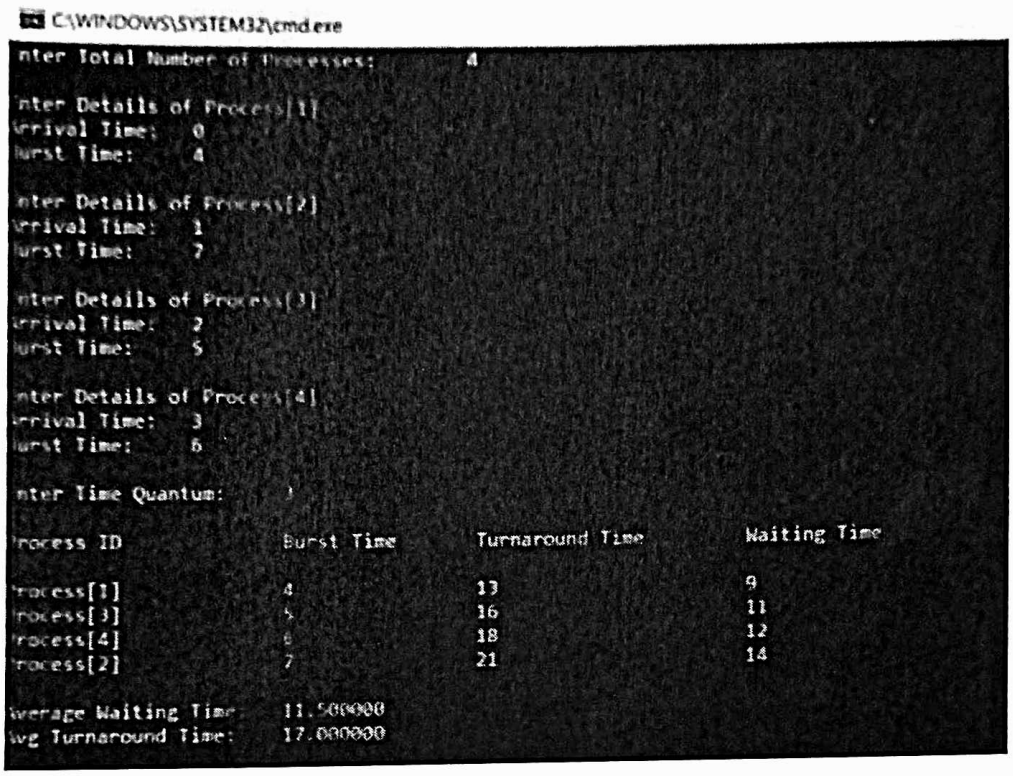
```
# include <stdio.h>
int main()
{
    int n;
    printf("Enter Total no of proces : ");
    Scanf ("%d", &n);
    int wait = 0, turn arro = 0, arr[n], burst[n], temp[n];
    int x=n;
    for (int i=0; i<n; i++) {
        Printf("Enter details %d\n", i+1);
        printf ("A rrival time :");
        Scanf ("%d", &arr[i]);
```

```
        Printf("Burst Time : ");
        Scanf ("%d", & burst[i]);
}
int time_quant;
printf("Enter Quant: ");
Scanf ("%d", &time_Quant);
int total = 0; counter = 0, i;
printf (" Process ID Burst time Turn Around Time Waiting Time \n");
for (total = 0; i = 0 ; x != 0) {
    if (temp [i] <= time_quant && temp[i] >0) {
            total = total + temp [i];
            temp [i] = 0;
            counter = 1;
    }
    else if (temp[i] >0) {
            temp [i] = temp [i] - time_quant;
            total + = time_quant;
    }
    if (temp[i] == 0 && counter == 1) {
            x--;
            printf ("\n Process No %d \t \t %d \t \t %d ", i+1)
            burst [i], total_arr [i], total_arr[i] - burst(i)
            wait = wait + total_arr[i] - burst[i];
            turnaro = total arr[i];
            counter = 0;
    }
    if (i == n-1)
        i = 0;
    else if (arr [i+1] <= total)
        i + 1
    else
        i = 0;
}
```

```
float avg w = (float) wait \n;
float avg t = (float) turnaround \n;
printf ("\n Average Waiting Time : %.f ", avg w);
printf (" \n Average Turn Around Time : %.f , avg t);
return 0;
}
```



Sample Output:



Output:

Enter Total no. of Process : 3
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 Time Quantum : 2

Process ID	Burst time	Turn Around time	Waiting Time
1	4	8	4
3	5	13	8
2	7	15	8

Average Waiting Time : 6.66 ms
Average Turn Around time : 12.00 ms

Result:

Thus the Round Robin Algorithm is executed