

Ex. No.: 6d)
Date 20/03/2025

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 Process : ");
    scanf("%d", &n);
    int wait = 0, TAT = 0, arr[n], b[n], temp[n];
    int x = n;
    for (int i = 0; i < n; i++) {
        printf("Enter details %d\n", i + 1);
        printf("Arrival Time : ");
        scanf("%d", &arr[i]);
        printf("Burst Time : ");
        scanf("%d", &b[i]);
    }
}
```

```

int tq;
printf("Enter Time Quantum : ");
scanf("%d", &tq);
int total=0; count=0, i;
printf(" Process ID Burst time TAT  

Waiting Time \n");
for (total=0; i=0; x!=0){
    if (temp[i] <= tq && temp[i] > 0){
        total = total + temp[i];
        temp[i] = 0;
        counter = 1;
    }
    else if (temp[i] > 0){
        temp[i] = temp[i] - tq;
        total += tq;
    }
    if (temp[i] == 0 && counter == 1){
        printf("\n Process NO %d It %d It %d It %d  

%d ", i+1,
        b[i], arr[i], arr[i]-b[i]);
        wait = wait + arr[i]-burst[i];
        tAT = arr[i];
        count = 0;
    }
    if (i == n-1)
        i=0;
    else if (arr[i+1] <= total)
        i++;
    else
        i=0;
}

```

```
float avgw = (float)wait / n;  
float avgt = (float)tAT / n;  
printf("In Average waiting Time : %f",  
      avgw);  
printf("In Average TAT : %f", avgt);
```

}



OF policy

P

S

S

E

E

F

F

R

R

R

R

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 Quant : 2

| Process | Burst Time | Turn Around Time | Waiting Time |
|---------|------------|------------------|--------------|
| 1 | 4 | 8 | 4 |
| 2 | 5 | 13 | 8 |
| 3 | 7 | 15 | 8 |

Average Waiting Time : 6.66 ms

Average Turn Around Time : 12.00 ms.

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:

Thus the Round Robin Algorithm is executed
✓
8th.