**Ex. No.: 6d)**
**Date** 25.3.25

## 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.
```

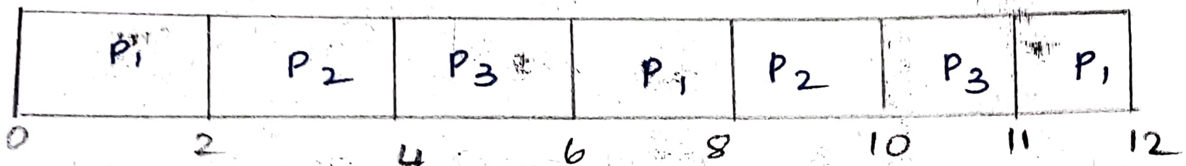
**Program Code:**

```
#include <stdio.h>
int main()
{
  int n,q;
  printf("Enter the no. of processes : ");
  scanf(" %d", &n);
  int bt[n] ,at [n], wt[n], tat[n], rt[n], ct[n], comp=0,
    t=0;
   float total_tat=0;  total_wt=0;
   for (int i=0; i<n; i++)
   {  printf(" Process %d Burst time: ", i+1);
      scanf("%d ", &bt [i]);
      Printf(" process %d Arrival Time: ", i+1);
      scanf("%d", &at(i]) ;
      rt[i] = bt[i];
   }
```

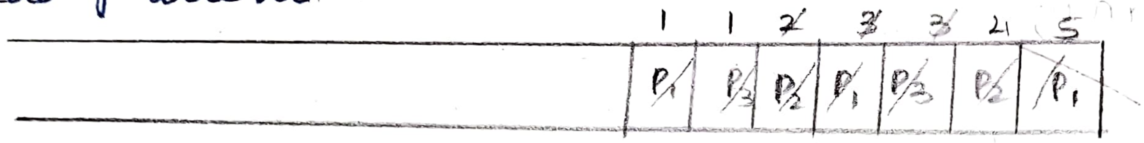
```
Printf ("Enter the Time quantum:");
 scanf ("%d", &q);
  while (comp < n)
 {
      int  done = 1;
      For (int i=0; i<n; i++)
      {
        if (rt[i]>0 && at[i]<= t)
        for (int i=0 ; i<n; i++)
        {
          if (rt [i]>0 && at [i] <= time){
           done=0;
           if (rt[i] > q){
               t+ =q;
               rt[i]  =q;
           }
           else {

               t+ =rt[i]
               rt[i]=0
               ct [i] =t
               tat(i] =ct[i] - at[i];
               wt(i] =tat (i] - bt [i];
                total-tat += tat[i];
               total - wt + =wt(i];
               comp++;

        }  }
   }
    if(done) time++;
 }
 float  avg-tat = total_tat/n;
 float  avg-wt  = total- wt /n;
 printf(" Process  Burst Time      ArrivalTime  TurnAroundTime
     waiting Time\n");
for (int i=0; i<n; i++)
{
    printf ("%d %d %d %d %d", i+1, bt[i], at[i],
        tat[i], wt[i]);
}
printf ("Average TurnAround Time = %.2f", avg-tat);
printf ("Average Waiting Time = %.2f", avg-wt);
return 0;
}
```

Time quantum : 2

Gantt chart:



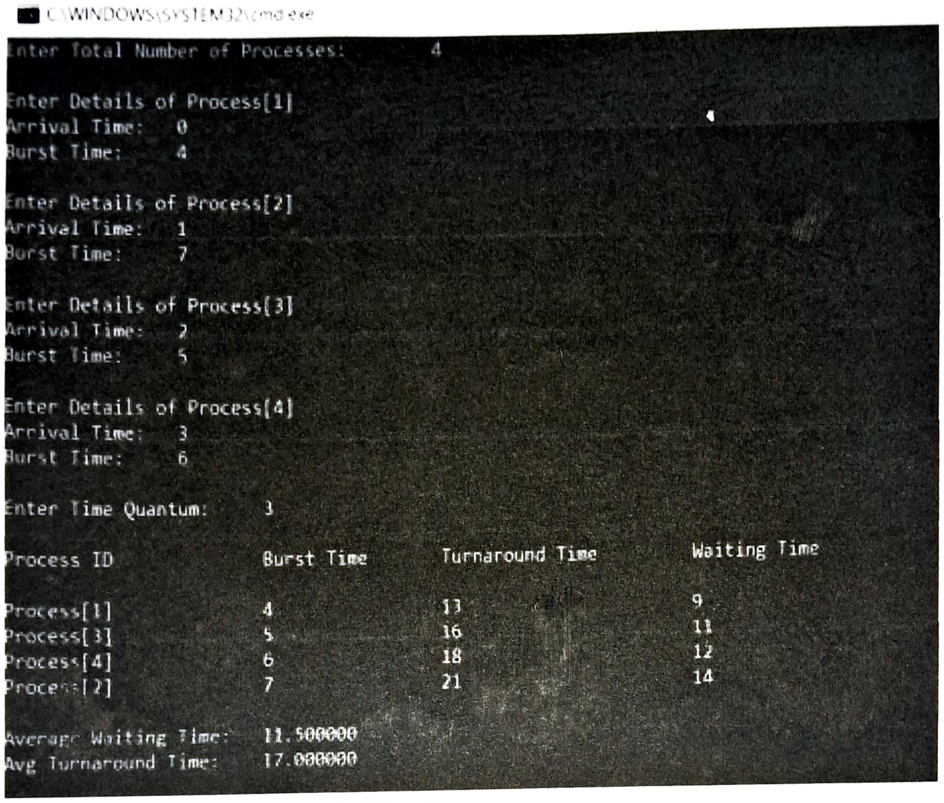
Ready Queue :



Tabulation :

| PROCESS | BT (ms) | AT (ms) | CT (ms) | TAT = CT - AT (ms) | WT = TAT - BT (ms) |
|---|---|---|---|---|---|
| 1 | 5 | 0 | 12 | 12 | 7 |
| 2 | 4 | 1 | 10 | 9 | 5 |
| 3 | 3 | 2 | 11 | 9 | 6 |

**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 the no. of processes : 3

Enter process1 Burst time : 5

Enter process2 Burst time : 4

Enter process3 Burst time : 3

Enter process 1 Arrival Time : 0

Enter process 2 Arrival Time : 1

Enter process 3 Arrival Time : 2

Enter the Time Quantum : 2

| Process | Burst Time | Arrival Time | Turn Around Time | Waiting Time |
|---|---|---|---|---|
| 1 | 5 | 0 | 12 | 7 |
| 2 | 4 | 1 | 9 | 5 |
| 3 | 3 | 2 | 9 | 6 |

Average Turn Around Time =10.00
Average waiting Time = 6.00

**Result:**

Thus the implementation of Round Robin CPU scheduling has been successfully executed.

[illegible]