Ex. No.: 6d)
Date 6. 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.
```

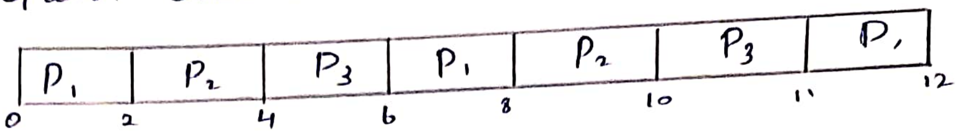
Program Code:

```
#include <Stdio.h>.
int main()
{
    int n,q;
    printf ("Enter the number of processes:");
    Scanf (" %d", &n);
    int bt[n], at [n], wt [n], tat [n], rt [n], ct[n],
    Comp =0, t =0;
    float ~~(int i=0; i<n; i++~~) 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];
    }.
```

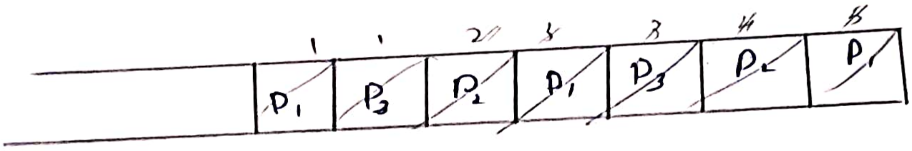
```
Printf ("Enter the time quantum: ");
Scanf (" %d", &q);
while (comp<n)
{
    int done =1;
    for (int i=0; i<n; i++)
    {
        if (rt [i] > 0 && at[i] <= time){

           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 ("program  Burst time  Arrival time  turn around time  waiting tim

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 turn around time = %.2f", avg_tat);
printf (Average waiting time = %.2F", avg_wt);
return 0;
}.
```

time quantum : 2

Gantt chart :



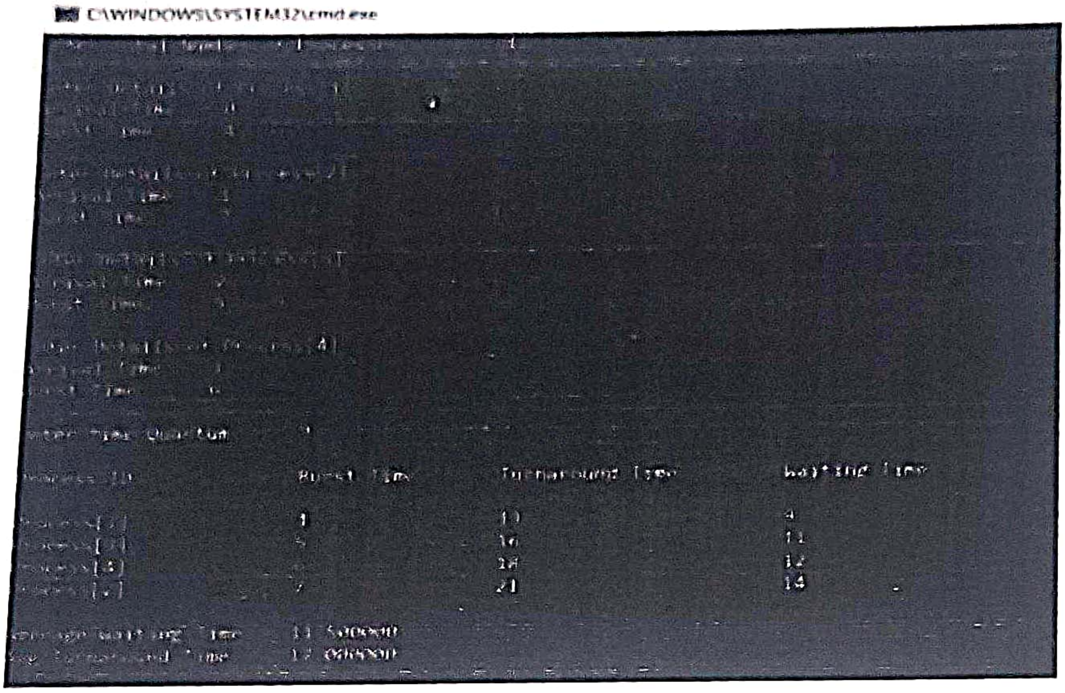
Ready queue :



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

```
[illegible]

[illegible]

[illegible]

[illegible]

Enter Time Quantum    [illegible]

Process ID          Burst Time       Turnaround Time        Waiting Time

Process[1]          [illegible]      13                     [illegible]
Process[2]          5                16                     11
Process[3]          [illegible]      18                     12
Process[4]          7                21                     14

Average Waiting Time    11.500000
Avg Turnaround Time     17.000000
```

Enter the no. of process : 3.
Enter The process 1 Burst time : 5
Enter The process 2 Burst time : 4
Enter the process 3 Burst time : 3.
Enter The process 1 Arrival time : 0
Enter the process 2 Arrival time : 1
Enter the 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 implentation of round robin cpu Scheduling has Been sucussfully executed.