Ex. No.: 6d)
Date 20/3/25

ROUND ROBIN SCHEDULING

Aim:

To implement the Round Robin (RR) scheduling technique

Algorithm:

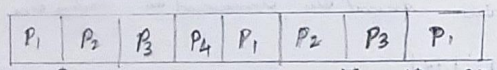
1. Declare the structure and its elements.
2. Get number of processes and Time quantum as input from the user.
3. Read the process name, arrival time and burst time
4. Create an array **rem_bt[]** to keep track of remaining burst time of processes which is initially copy of bt[] (burst times array)
5.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

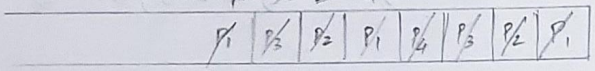
Program Code:

```
#include <stdio.h>
int main()
{
int q, n;
Printf("Enter the number of process: ");
scanf("%d", &n);
int bt[n], at[n], wt[n], tat[n], rat[n], et[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++;
}
```

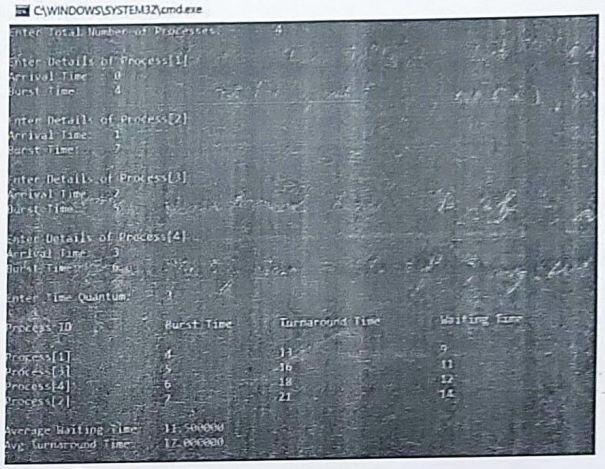
```
float avg_tat = total_tat / n;
float avg_wt = total_wt / n;
printf("\n Process  Burst Time  Arrival Time  Turnaround Time  Waiting Time \n");
for(int i = 0; i < n; i++)
{
  printf("\n %d  %d  %d  %d  %d", i+1,
                bt[i], at[i], tat[i], wt[i]);
}
printf("\n Average Turnaround time = %.2f", avg_tat);
printf("\n Average Waiting Time = %.2f", avg_wt);
return 0;
}
```





Process	AT (ms)	BT (ms)	CT (ms)	TAT = CT - AT (ms)	WT = TAT - BT (ms)
1	0	7	20	20	13
2	1	5	16	15	10
3	2	6	19	17	11
4	3	2	11	8	6

Sample Output:



Enter the no. of process: 4
Enter the burst time:
7
5
6
2

Enter the time quantum: 3
Enter the arrival time:
0
1
2
3

Process	Arrival time	Burst time	Waiting time	Turn Around time
4	3	2	6	8
2	1	5	10	15
3	2	6	11	17
1	0	7	13	20

Avg. waiting time is : 10 ms

Avg. turn around time is : 15 ms

Result:

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