Ex. No.: 6d)
Date

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>
void RR (int bt[] , int at[] , int n , int q)
{   int ct[n], tat[n], wt[n] ;
    int rem_bt[n];
    int to_tat=0 , tot_wt =0;
    for (int i =0; i<n ;i++)
        rem_bt[i] =bt[i];
int t=0;
int pr_left =n;
while (pr_left >0)
{   int done_this_round =0;
    for (int i=0; i <n;i++)
        { if (rem_bt[i]>0 && at[i]<=t)
            {   done_this_round =1;
```

```
            if (rem_bt[i] > q)
            {
                t += q;
                rem_bt[i] -= q;
            }
            else {
                t += rem_bt[i];
                ct[i] = t;
                rem_bt[i] = 0;
                pr_left--;
            }
        }
    }
    if (!done_this_round)
        t++;
}
for (int i=0; i<n; i++)
{   tat[i] = ct[i] - at[i];
    wt[i] = tat[i] - bt[i];
    tot_tat += tat[i];
    tot_wt += wt[i];
}
printf("\nProcess \tBT\t AT\t CT \t TAT \t WT \n");
for (int i=0; i<n; i++)
    printf(i+1, bt[i], at[i], ct[i], tat[i], wt[i]);
printf((float) tot_tat/n, (float) tot_wt/n);
}
```

```
int main ()
{
    int n, q;
    scanf ("%d", &n);
    int bt[n], at[n];
    for (int i =0; i <n; i++)
      { scanf ("%d", &bt[i]);
      }
    for (int i =0; i<n; i++)
      { scanf ("%d", & at[i]);
      }
    scanf ("%d", &q);
    ar (bt, at, n, q);
    return 0;
}
```

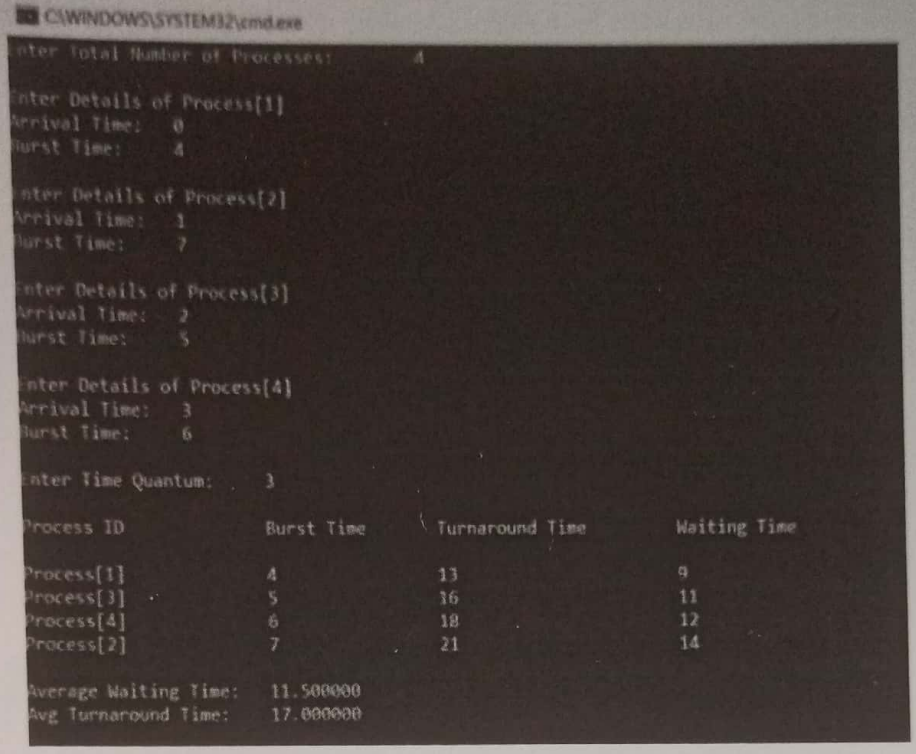
OUTPUT

Total no. of process : 3
Details of process : 1
Arrival Time : 0
Burst Time : 4
Enter Details of process : 2
Arrival Time : 1
Burst Time : 7
Details of process : 3
Arrival Time : 2
Burst Time : 5
Quantum : 2

Process ID	BT	TAT	WT
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.

Sample Output:



Result:

Thus the given code for Round Robin algorithm is executed successfully.