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>
int main () {
 int n;
 printf ("Enter no. of processes: ");
 scanf ("%d", &n);
 int wait =0, tat =0, arr[n], burst[n], temp[n];
 int x = n;
 for (int i=0 ; i<n ; i++)
   printf (" Enter ~~burst~~ time %d: process[i])
   scanf (" %d ", &burst[i]);
   printf (" %d", & arr[i];
 } rem_burst [i] = burst[i];
```

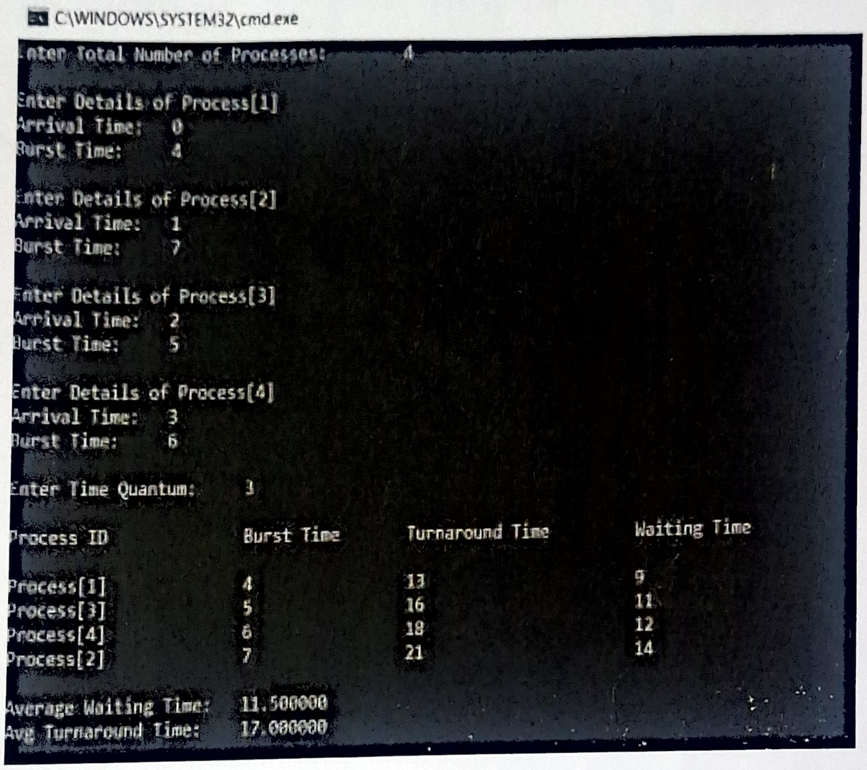
```
printf ("Enter time quantum: ");
int time = 0;
int done = 0;
int queue [n];
for (int i = 0; i<n ; i ++){
    queue [i] = i; }
while (done <n) {
    done = 0;
    for (int i=0; i<n ; i++){
      int   idx = queue [i];
      if (rem_burst [idx] >0 && arr[idx]
                                <= time) {
        if (rem_burst[idx] > time_quant){
            time + = time_quantum;
            rem_burst [idx] - = time_quant;
        } else {
          time + = rem_burst [idx];
            wait[idx] = (time_arr) - (burst [idx]);
            ~~rem~~-burst [idx] = 0; }  }
      if (rem_burst [idx] = =0) {
            done ++;
        }
    }
for (int i=0; i<n; i++){
      tat[i] = burst [i]+  wait[i];
}
int avg-wait =0 , avg-tat =0
 for (int i=0; i<n; i++){
      avg-wait + = wait[i];
      avg-tat += tat [i]; }
  printf(" \n Average waiting time : %..2f",
                                    (float) avg-wait/n);

   printf(" \n Average Turn around : %.2f \n",
                         (float) avg-turn/n);

     return 0;

 }
```
✓

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 processes : 4

Enter process [1]

Arrival time : 0 burst time : 6

Enter process [2]

Arrival time : 1 Burst time : 5

Enter process [3]

Arrival time : 2 Burst time : 4

Enter process [4]

Arrival time : 3 Burst time : 3

Proces	Burst time (ms)	Arrival time (ms)	Turn around time (ms)	Waiting time (ms)
1	6	0	17	11
2	5	1	17	12
3	4	2	10	6
4	3	3	12	9

Average waiting time : 9.5 ms

Average Turnaround time : 14 ms

Result:

Thus the program for Round Robin scheduling is executed succesfully.

[illegible]