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>
#define N 4
int at[N], bt[N], ct[N], wt[N], tat[N], rt[N], tq=4;
int queue[N], front=-1, rear=-1;

void enqueue(int a){
    if(front==-1){
        front=0;
    }
    rear=(rear+1)%N;
    queue[rear]=a;
}
```

```
int dequeue()
{
    int item = queue[front];
    if(front == rear){
        front =-1;
        rear=-1;
    }
    else{
        front =(front +1) % N;
    }
    return item;
}
void readyqueue()
{
    int t =0;
    for(int j =0; j<N; j++){
        if( at[j]<=t){
            enqueue(j);
            t++;
        }
        else {
            t++;
        }
    }
}
void gantchart(){
    int count =0,t =0;
    while (count<N){
        int p = dequeue();
        if( rt[p] > tq){
            rt[p] - =tq;
            t + = tq;
            enqueue(p);
        }
    else{
        t+= rt[p];
        rt[p] = 0;
        ct[p] = t;
        count++;
    }
  }
}

int main(){
    printf("Enter arrival time :\n");
    for( int i=0; i<N; i++){
        scanf("%d", &at[i]);
    }
    printf("\n Enter burst time :\n");
    for(int i =0; i <N; i++) {
        scanf("%d", &bt[i]);
        rt[i] = bt[i];
    }
    readyqueue();
    ganntchart();
    printf("\n Process\t AT \t BT \t CT\t TAT \t WT \n");
    for( int i=0; i<N; i++) {
        tat [i] = ct[i]- at[i];
        wt[i] = tat[i]- bt[i];
        atat+= tat[i];
        awt += wt[i];
        printf("%d\t %d\t %d\t %d\t %d\t %d\t", i+1,
                              at[i], bt[i], ct[i],
                          tat[i], wt[i];
    }
```

```
printf("%.2f\n %.2f\n"), (float)atat/4, (float)
                                    awt/4);

   return 0;
}
```

OUTPUT:

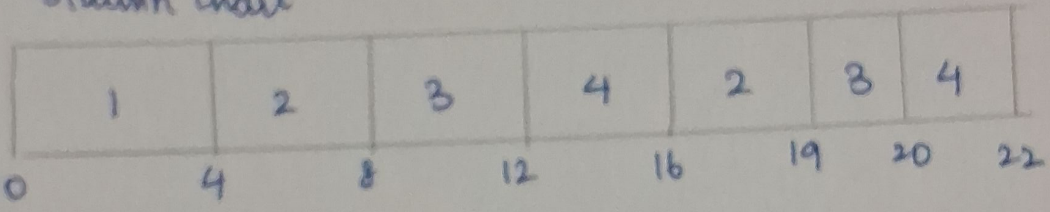
```
 Enter the arrival time:
  0
  1
  2
  3
Enter burst time:
  4
  7
  5
  6
Process   AT   BT   CT   TAT   WT
  1       0    4    4    4     0
  2       1    7    19   18    11
  3       2    5    20   18    13
  4       3    6    22   19    13

Average TAT = 14.75 ms
Average WT = 9.25 ms.
```

Process	AT (ms)	BT (ms)	CT (ms)	TAT (ms)	WT (ms)
1	0	4	4	4	0
2	1	7	19	18	11
3	2	5	20	18	13
4	3	6	22	19	13

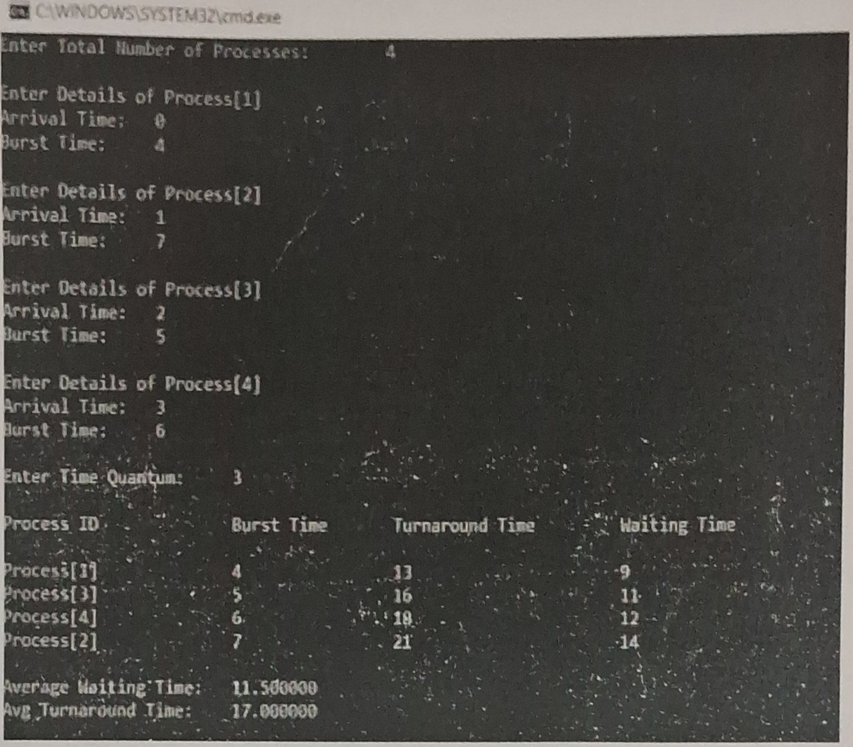
Average TAT = 14.75 ms

Average WT = 9.25ms.

Gantt Chart



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                19                      12
Process[2]              7                21                      14

Average Waiting Time:   11.500000
Avg Turnaround Time:    17.000000
```

Result:

Round robin CPU scheduling algorithm is implemented in C.

[illegible]