Ex. No.: 6d)
Date 27-02-25

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>
struct process {
        int pid, at, bt, ct, rem, wt, tat;
};
void sort(struct process p[], int n)
{ for(int i=0; i<n-1; i++){
        for(int j=0; j<n; j++){
                if(p[j].at > p[j+1].at) {
                        struct process t = p[j];
                        p[j] = p[j+1];
                        p[j+1] = t;
}}}}
```

```
int main(){
  int n;
   printf ("Enter the number of processes: ");
   scanf ("%d",&n);
   struct process p[n];
   for(int i=0;i<n ;i++){
        printf("Enter details of process [%d]\n", i+1);
        p[i] . pid = i+1;
        printf ("Arrival time: ");
        scanf ("%d", & p[i].at);
        printf ("Burst time");
        scanf("%d", & p[i].bt);
        p[i] . rem = p[i]. bt ;
        p[i]. ct =0;
   }
  int q;
  printf ("Enter the quantum: ");
  scanf ("%d", &q);
   int t=0, c=0;
   while (c<n)
     {for(int i=0; i<n ;i++)
      { if (p[i]. rem >q)
        { t+=q;
          p[i].rem =q;
        }
       elseif (p[i].ct = 0)
         { t += p[i].rem;
           p[i].rem = 0;
           p[i] . ct = t;
           p[i]. tat = p[i] . ct - p[i] . at;
           p[i].wt = p[i]. tat - p[i].bt;
           c++;
   } } }
```

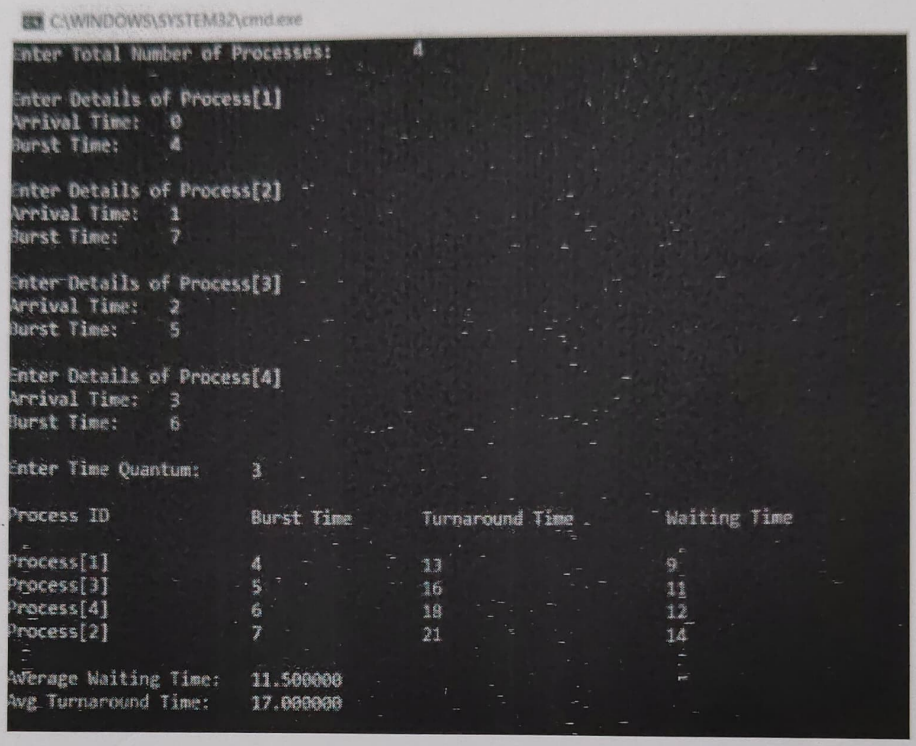
```
printf ("Process ID    Burst time    Turnaround time    Waiting time \n");
float twt =0, ttat =0;
for(int i = 0; i<n; i++)
  {
     twt = twt + p[i].wt;
     ttat = ttat + p[i].tat;
     printf ("%d    %d    %d    %d\n", p[i].pid,
                                      p[i].bt,
                                  p[i].tat, p[i].wt);
  }
printf ("Average Waiting Time = %.1f ", (twt/n));
 printf ("\n");
printf ("Average Turnaround Time = %.1f",
                                  (total/n));
}
```

P	AT (ms)	BT (ms)	CT (ms)	TAT (ms)	WT (ms)
1	0	4	8	8	4
2	0	5	12	12	7
3	0	3	11	11	8

Average Turn around time = 10.33

Average waiting time = 6.33

Sample Output:



Result:

Hence the round robin scheduling has been executed successfully.