Ex. No.: 6d)
Date 28/2/2025

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:

```
import array
num = int(input("Enter number process : "))
p = array.array('i', range(1, num+1))
bt = array.array('i', map(int, input("Enter
    the burst time of process : ").split()))
at = array.array('i', map(int, input
    ("Enter the arrival time of process:")
    .split()))
```

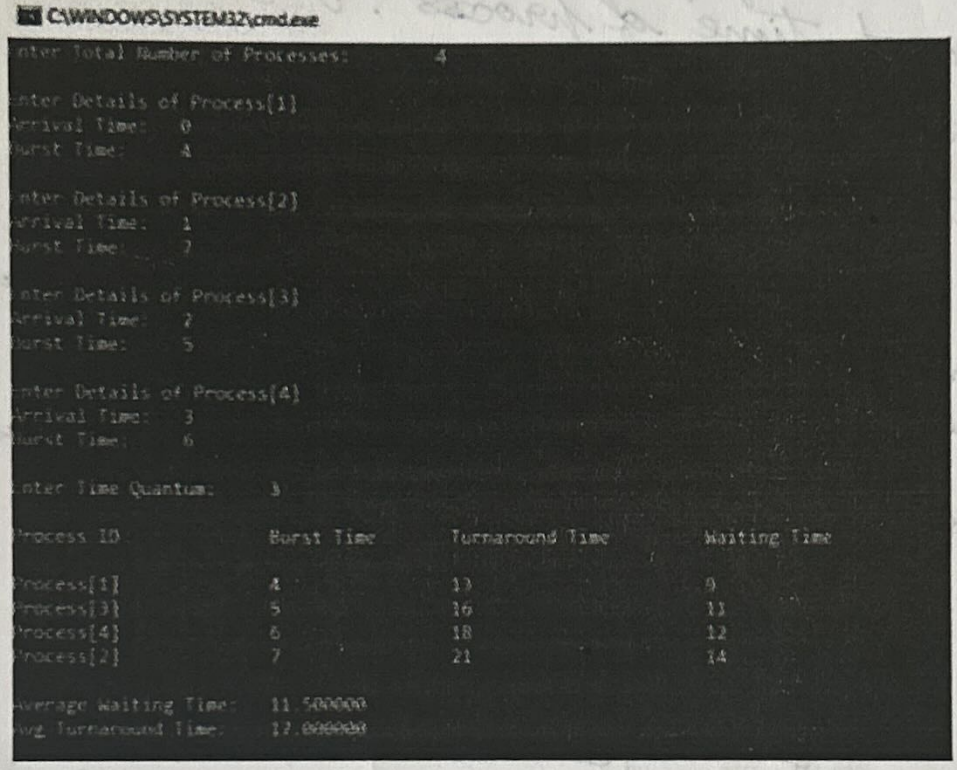
```
n = len(p)
wt = array.array('i', [0]*n)
tat = array.array('i', [0]*n)
rem_bt = array.array('i', bt[:])
tq = int(input("Enter time quantum :"))
time = 0
for i in range(n-1):
    for j in range(i+1, n):
        if at[i] > at[j]:
            at[i], at[j] = at[j], at[i]
            bt[i], bt[j] = bt[j], bt[i]
            rem_bt[i], rem_bt[j] = rem_bt[i]
                                   , rem_bt[i]

While true:
    complete = true
    for i in range(n):
        if rem_bt[i] > 0 and at[i] <= time:
            complete = False
            if rem_bt[i] > tq:
                time += tq
                rem_bt[i] -= tq

            else
                time += rem_bt[i]
```

```
        wt[i] = time - bt[i] - at[i]
        rem - bt[i] = 0
    if complete :
        break
for i in range(n):
    tat[i] = bt[i] + wt[i]
avg - wt = sum(wt) / n
avg - tat = sum(tat) / n
print("Process \t burst time \t Arrival time \t
      Waiting time \t turnaround time")
for i in range(n):
    print(f"{p[i]} \t {bt[i]} \t \t {at[i]}
      \t \t {wt[i]} \t \t {tat - [i]}")
Print(f"\n average waiting time : {avg - wt : 2f}")
Print(f"Average turnaround time = {avg - tat : 2f}")
```

Sample Output:



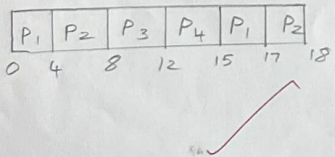
Output

Enter the no of processes : 4
Enter burst time of process : 6 5 4 3
Enter the arrival time of process : 0 1 2 3
Enter the time quantum : 4

Process	Arrival time (ms)	Burst time (ms)	Completion time (ms)	Waiting time (ms)	Turnaround time (ms)
1	0	6	17	17	11
2	1	5	18	17	12
3	2	4	12	10	6
4	3	3	15	12	9

Average waiting time = 9.5 ms
Average turnaround time = 14 ms

The gantt chart for the schedule is



Result:

Thus the program for round robin scheduling have been studied successfully