Name : Masuk Mia **ID : IT-18049**

Lab report no. : 10

Lab report name : Lab report on implementation of Round robin scheduling algorithm.

Aim and objectives: To learn about Round robin scheduling algorithm , To know how much time spends on context switching by Round robin method **,** implement it with a c program, to learn how to use this algorithm.

Explanation:

i) **Round robin Scheduling algorithm:** Round Robin Scheduling is a CPU scheduling algorithm that assigns CPU on basis of FCFS for fixed time called as time quantum. This fixed amount of time is called as time quantum or time slice. After the time quantum expires, the running process is preempted and sent to the ready queue. It is simple, easy to implement, and starvation-free as all processes get fair share of CPU. Context switching is used to save states of preempted processes

ii) **Implementation of Round robin scheduling algorithm in C Program:**

Code:

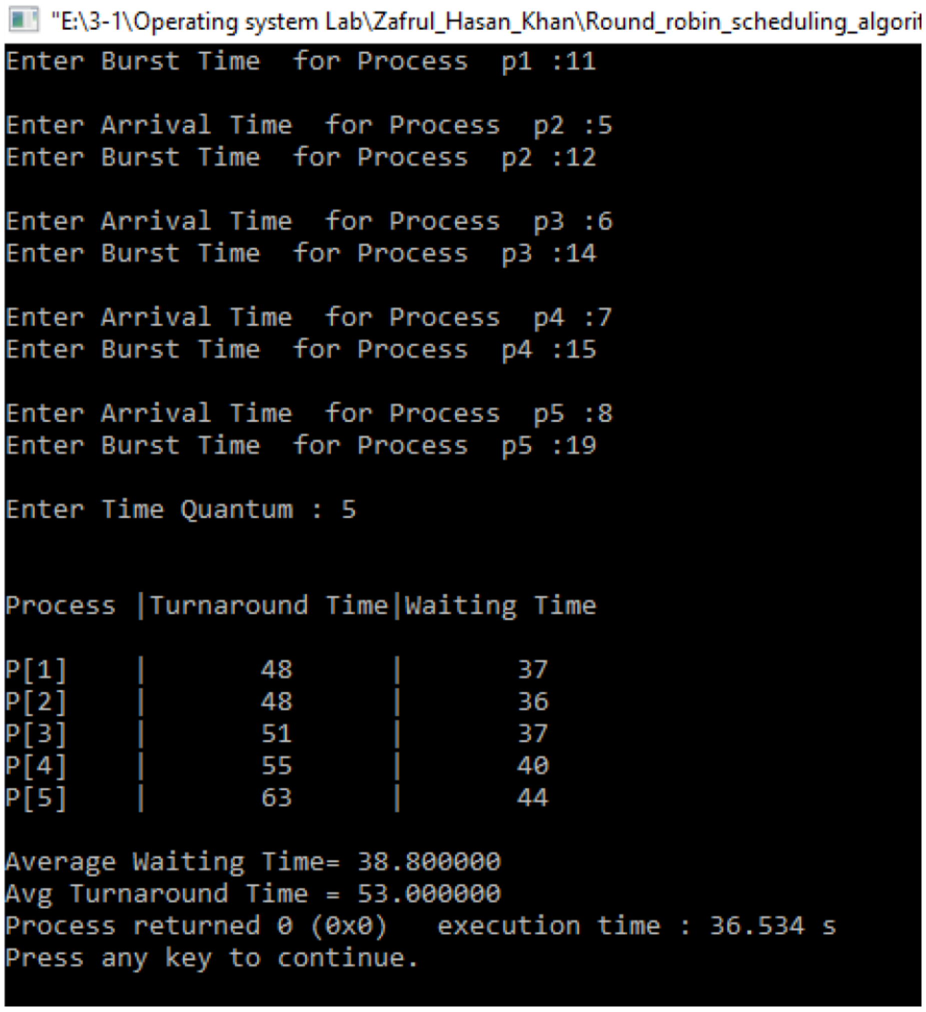
```
#include<stdio.h> int
main()
{

  int count,j,n,time,remain,flag=0,time_quantum;   int
wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
  printf("Enter Total Process: ");
scanf("%d",&n);
```

```
  printf("\n");
remain=n;
  for(count=0;count<n;count++)
  {
    printf("Enter Arrival Time  for Process  p%d :",count+1);
scanf("%d",&at[count]);
    printf("Enter Burst Time  for Process  p%d :",count+1);
scanf("%d",&bt[count]);     rt[count]=bt[count];
printf("\n");
  }
  printf("Enter Time Quantum : ");
scanf("%d",&time_quantum);
  printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
for(time=0,count=0;remain!=0;)
  {
    if(rt[count]<=time_quantum && rt[count]>0)
    {
      time+=rt[count];
rt[count]=0;       flag=1;
    }
    else if(rt[count]>0)
    {
      rt[count]-=time_quantum;
      time+=time_quantum;
    }
    if(rt[count]==0 && flag==1)
    {
remain--;
      printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-
at[count],timeat[count]-bt[count]);       wait_time+=time-
at[count]-bt[count];       turnaround_time+=time-at[count];
      flag=0;
    }
    if(count==n-1)
count=0;     else
if(at[count+1]<=time)
count++;
    else
count=0;
  }
  printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

  return 0;
}
```

Output:



```
"E:\3-1\Operating system Lab\Zafrul_Hasan_Khan\Round_robin_scheduling_algorit
Enter Burst Time  for Process  p1 :11

Enter Arrival Time  for Process  p2 :5
Enter Burst Time  for Process  p2 :12

Enter Arrival Time  for Process  p3 :6
Enter Burst Time  for Process  p3 :14

Enter Arrival Time  for Process  p4 :7
Enter Burst Time  for Process  p4 :15

Enter Arrival Time  for Process  p5 :8
Enter Burst Time  for Process  p5 :19

Enter Time Quantum : 5


Process |Turnaround Time|Waiting Time

P[1]    |       48      |       37
P[2]    |       48      |       36
P[3]    |       51      |       37
P[4]    |       55      |       40
P[5]    |       63      |       44

Average Waiting Time= 38.800000
Avg Turnaround Time = 53.000000
Process returned 0 (0x0)   execution time : 36.534 s
Press any key to continue.
```

Conclusion: From this lab , I have known that Round robin scheduling algorithm offers starvation free execution of processes and each ready task runs turn by turn only in a cyclic queue for a limited time slice. It is the oldest, simplest scheduling algorithm, which is mostly used for multitasking.