Saurabh Kumar Singh(16100051)

Round Robin Scheduling

ALGORITHM::
Mode:- Preemptive
Criteria:- (Time Quantum+Arrival Time)

1. The queue structure in ready queue is of First In First Out (FIFO) type.

2. A fixed time is allotted to every process that arrives in the queue. This fixed time is known as time slice or time quantum.

3. The first process that arrives is selected and sent to the processor for execution. If it is not able to complete its execution within the time quantum provided, then an interrupt is generated using an automated timer.

4. The process is then stopped and is sent back at the end of the queue. However, the state is saved and context is thereby stored in memory. This helps the process to resume from the point where it was interrupted.

5. The scheduler selects another process from the ready queue and dispatches it to the processor for its execution. It is executed until the time Quantum does not exceed.

6. The same steps are repeated until all the process are finished

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];
  int flag1[10],temp[10];
  for(count = 0;count<10;count++)
          flag1[count] = 0;
// enter the total number of process
  printf("Enter Total Process:\t ");
  scanf("%d",&n);

  remain=n;

// enter the details of all the process (arrival time and burst time)
  for(count=0;count<n;count++)
```

```
  {
        printf("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);
        scanf("%d",&at[count]);
        scanf("%d",&bt[count]);
        rt[count]=bt[count];
  }

// Define the time for which one process will run in a single loop
  printf("Enter Time Quantum:\t");
  scanf("%d",&time_quantum);

  printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
  for(time=0,count=0;remain!=0;)
  {

// The burst time is less than quantum and hence the process will be completely execute
// in its first run
        if(rt[count]<=time_quantum && rt[count]>0)
        {
        time+=rt[count];
        rt[count]=0;
        flag=1;
        if(flag1[count]==0 ){
        //printf("%d was here \t",count);
        flag1[count] = 1;
        }
        }

// if not then the process will run till quantum times in a single run
        else if(rt[count]>0)
        {
        if(flag1[count]==0 ){
        //printf("%d was here \t",count);
        flag1[count] = 1;
        }
        rt[count]-=time_quantum;
        time+=time_quantum;
        }

// if process is completed printing the wait and turn around time
        if(rt[count]==0 && flag==1)
        {
        remain--;
        printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);
        wait_time+=time-at[count]-bt[count];
        turnaround_time+=time-at[count];
        flag=0;
        }
```

```
            if(flag1[count]==1){
            temp[count] = time;
            flag1[count] = 999;
            }

// if all the processes execution for one time is completed or not
            if(count==n-1)
            count=0;
            else if(at[count+1]<=time)
            count++;
            else
            count=0;
  }

  for(count = 0;count<n;count++)
            printf("%d is %d \n",count, temp[count]-at[count]-2);

// printing the average waiting and tat
printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
  printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

  return 0;
}
```

I/O::

```
Enter Total Process:     4
Enter Arrival Time and Burst Time for Process Process Number 1 :0 5
Enter Arrival Time and Burst Time for Process Process Number 2 :0 4
Enter Arrival Time and Burst Time for Process Process Number 3 :0 2
Enter Time Quantum:     2


Process |Turnaround Time|Waiting Time

P[3]    |      6       |      4
P[2]    |      12      |      8
P[4]    |      13      |      10
P[1]    |      14      |      9
0 is 0
1 is 2
2 is 4
3 is 6
```

Average Waiting Time= 7.750000
Avg Turnaround Time = 11.250000