Lab Report No. **10**

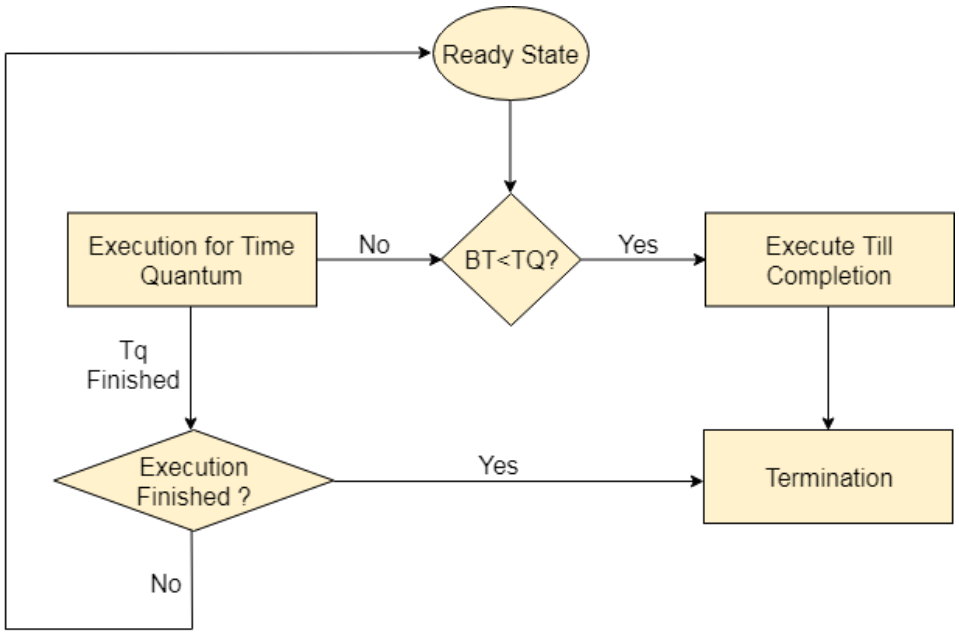
Lab Report Name: **Implementation of Round Robin Scheduling algorithm .**

Objectives:

i. What is Round Robin Scheduling algorithm.
ii. How to implementation

Theory : Round robin is the most widely used process scheduling algorithm .The basic strategy for round robin scheduling is that if there are n process,each of the process will receive 1/n CPU Execution Time.Each process is allotted a time quanta, for which its is executed.The incoming processes are kept in a ready list while another one is executing.If the time quanta allotted for a process is over,then that process is moved to ready and the next process in the ready list is executed for the allotted time quanta.

The Complete Example Implementation Source Code in C of Round Robin Algorithm to schedule N Processes and to calculate the Execution,wait time and turn around time is given below.



Characteristics of Round-Robin Scheduling

Here are the important characteristics of Round-Robin Scheduling:

1. Round robin is a pre-emptive algorithm.
2. The CPU is shifted to the next process after fixed interval time, which is called time quantum/time slice.
3. The process that is preempted is added to the end of the queue.
4. Round robin is a hybrid model which is clock-driven
5. Time slice should be minimum, which is assigned for a specific task that needs to be processed. However, it may differ OS to OS.
6. It is a real time algorithm which responds to the event within a specific time limit.
7. Round robin is one of the oldest, fairest, and easiest algorithm.
8. Widely used scheduling method in traditional OS.

Advantages

1.It can be actually implementable in the system because it is not depending on the burst time.

2.It doesn't suffer from the problem of starvation or convoy effect.

3.All the jobs get a fare allocation of CPU.

Disadvantages

1.The higher the time quantum, the higher the response time in the system.

2.The lower the time quantum, the higher the context switching overhead in the system.

3.Deciding a perfect time quantum is really a very difficult task in the system.

Corresponding Code:

```
#include<stdio.h>

int main()

{

   int n,i,k,x=0,s=0,r=0,q=0,a[30],e[30],t[30];

   float m,p=0;
```

```
printf("Enter the number of process: ");
scanf("%d",&n);
printf("Enter the execution time: ");
for(i=0; i<n; i++)
{
   scanf("%d",&a[i]);
   e[i]=a[i];
}
printf("Enter the quanta: ");
scanf("%d",&q);
printf("After Round Robin sheduling: ");
for(i=0; i<n; i++)
{
   if(x<a[i])
   {
      x=a[i];
   }
}
k=x/q;
while(s<=k)
{
   for(i=0; i<n; i++)
   {
      if(a[i]>0)
      {
         if(a[i]>q)
```

```
        {
            r=r+q;
            a[i]=a[i]-q;
            printf("P%d\t",i+1);
        }else
        {
            r=r+a[i];
            a[i]=a[i]-q;
            printf("P%d ",i+1);
            t[i]=r;
        }
      }
   }
   s++;
}
printf("\n\nProcess   BurstTime    WaitingTime    TurnAroundTime\n");
for(i=0; i<n; i++)
{
   printf(" %d \t\t %d\t\t %d\t\t %d\t\t \n",i,e[i],x,t[i]);
   x=x+q;
}
m=x/n;
printf("\nAverage waiting time=%f= ",m);
printf("\nAverage turn around time= ");
for(i=0; i<n; i++)
   p=p+t[i];
```

```
    p=p/n;
    printf("%f",p);
    printf("\n");
    return 0;
}
```

Output: