Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 10

Course code: ICT-3110

Course title: Operating Systems Lab

Date of Performance:

Date of Submission: 11/09/2020

Submitted by

Name: Golam Kibria Tuhin

ID:IT-18015

3th year 1ndsemester

Session: 2017-2018

Dept. of ICT

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Name of the lab report : Implementation of Round Robin Scheduling Algorithm.

Objective: Round Robin Scheduling algorithm Definition & executable code in c are followed.

Q.1 What is Round Robin Scheduling algorithm?

Ans: Round robin scheduling is the preemptive scheduling in which every process get executed in a cyclic way, i.e. in this a particular time slice is allotted to each process which is known as time quantum. Every process, which is present in the queue for processing, CPU is assigned to that process for that time quantum. Now, if the execution of the process gets completed in that time quantum, then the process will get terminate otherwise the process will again go to the ready queue, and the previous process will wait for the turn to complete its execution.

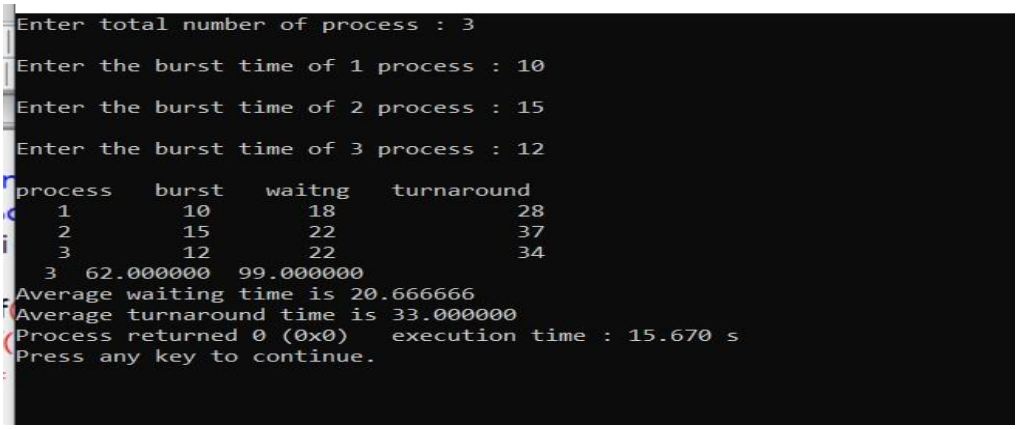
Q.2 How to implemented in C?

Ans:

```
#include<stdio.h>

int main() {

   int burst [100],waiting [100],turnaround [100],b[100];

int i,n,time,count=0;

   float total=0,totalTT=0,avgwt,avgtt;


   printf("Enter total number of process : ");

scanf("%d",&n);     for(i=0; i<n; i++)

   {

      printf("\nEnter the burst time of %d process : ",i+1);

scanf("%d",&burst [i]);         b[i] = burst [i];

   }

i=0;

   for(time=0; count!=n; time++)

   {

      while(burst [i] == 0)
```

```
        {
i=(i+1)%n;
        }         burst
[i]--;         if(burst
[i]==0)
        {
            turnaround [i] = time+1;
count++;
        }         i =
(i+1)%n;
    }
    printf("\nprocess   burst   waitng   turnaround   ");
for(i=0; i<n; i++)
    {
        waiting [i] = turnaround [i] - b[i];
        printf("\n   %d \t    %d \t     %d \t    %d",i+1,b[i],waiting [i],turnaround [i]);
total = total + waiting [i];          totalTT = totalTT + turnaround [i];
    }
    printf("\n  %d  %f  %f",n,totalwt,total);
avgwt = totalwt / n;      avgtt = totalTT / n;
    printf("\nAverage waiting time is %f",avgwt);
printf("\nAverage turnaround time is %f ",avgtt);
return 0;
}
```

Output:



Conclusion:

A comparative study of Dynamic Quantum with Re-Adjusted Round Robin Scheduling algorithm, Optimized RR algorithm and proposed one is made. It is concluded that the proposed algorithm is superior to the other two algorithms as it has less waiting time, less turnaround time and usually less context switching thereby reducing the overhead and saving of memory space. Future work can be based on this algorithm modified and implemented for hard real time system where deadlines of the processes are to be taken into consideration.