**Lab Report No: 10**

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

**Name: Binodon**

**ID: IT-17046**

**Objective**: Round Robin Scheduling algorithm Definition & executable code in codeblocks

**Q.1 What is Round RobinScheduling 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.

The scheduling drives its name from the principle which is known as a round robin in which every person takes an equal share of anything they have in turn. We make use of round robin scheduling algorithm in a time-sharing system. It is generally used by those os which has multiple clients to make use of resources.

**Q.2 How to implemented in C?**

**Ans:**

#include<stdio.h>

int main()

{

intbursttime[100],waitingtime[100],turnaroundtime[100],b[100];

inti,n,time,count=0;

floattotalwt=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",&bursttime[i]);

b[i] = bursttime[i];

}

i=0;

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

{

while(bursttime[i] == 0)

{

i=(i+1)%n;

}

bursttime[i]--;

if(bursttime[i]==0)

{

turnaroundtime[i] = time+1;

count++;

}

i = (i+1)%n;

}

printf("\nprocess burst waitng turnaround ");

for(i=0;i<n;i++)

{ waitingtime[i] = turnaroundtime[i] - b[i];

printf("\n %d \t %d \t %d \t %d",i+1,b[i],waitingtime[i],turnaroundtime[i]);

totalwt = totalwt + waitingtime[i];

totalTT = totalTT + turnaroundtime[i];

}

printf("\n %d %f %f",n,totalwt,totalTT);

avgwt = totalwt / n;

avgtt = totalTT / n;

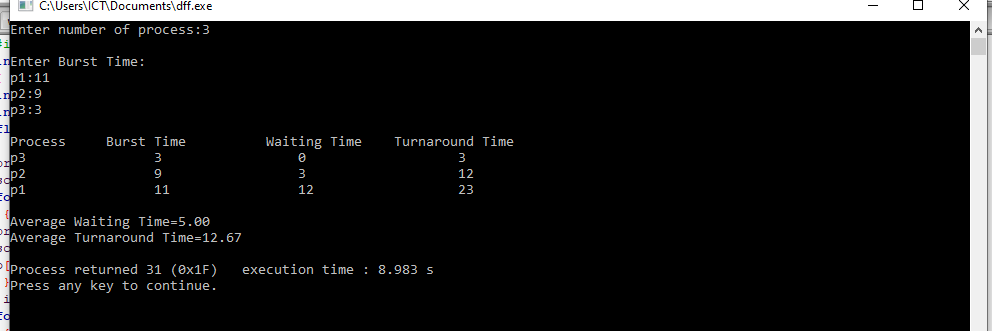
printf("\nAverage waiting time is %f",avgwt);

printf("\nAverage turnaround time is %f ",avgtt);

return 0;

}

**Output:**

****