**Ex.4D** **Round Robin Scheduling**
**Date:**

**Aim:**

To write a program to implement the Round Robin CPU scheduling Algorithm.

**Algorithm:**

1. Start the program
2. Get the number of processors
3. Get the Burst time(BT) of each processors
4. Get the Quantum time(QT) or time slice.
5. Execute each processor until reach the QT or BT
6. Time of reaching processor's BT is it's Turn Around Time(TAT)
7. Time waits to start the execution, is the waiting time(WT) of each processor
8. Calculation of Turn Around Time and Waiting

   Time 8.1.tot_TAT = tot_TAT + cur_TAT

   8.2.avg_TAT = tot_TAT/num_of_proc

   8.3.tot_WT = tot_WT + cur_WT

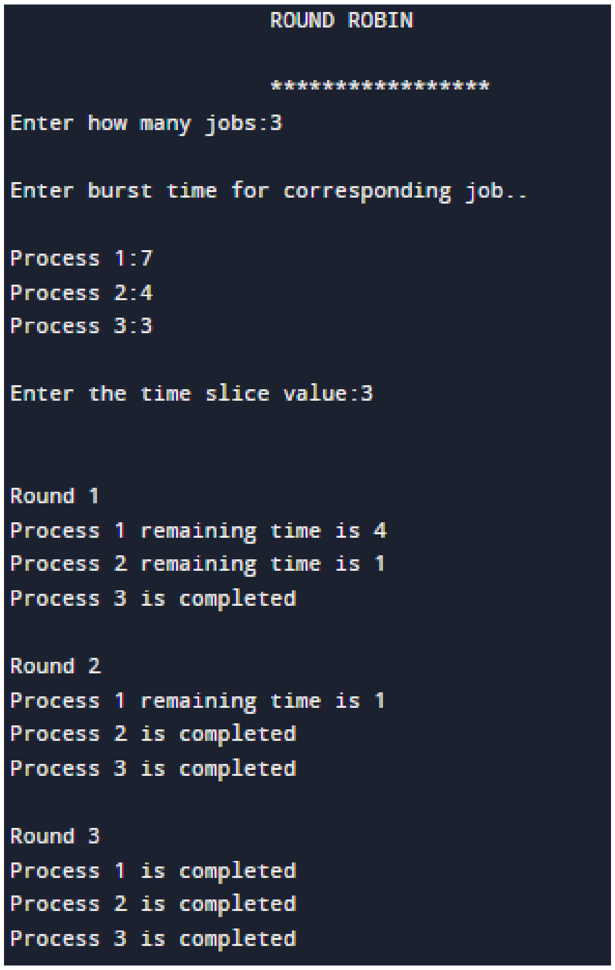
8. 4.avg_WT = tot_WT/num_of_proc
9. Display the result
10. STOP the program

**Program:**

```
#include<stdio.h>
int n,b[10],z[10],q,i,j,r,m[50],e=0,avg=0;
float f;
int rr();
int main()
{
 printf("\n\n\t\t\t\t\tROUND ROBIN\n\n");
printf("\t\t\t\t\t*****************\n");
printf("Enter how many jobs:");
scanf("%d",&n);
printf("\nEnter burst time for corresponding job..\n");
printf("\n");
for(i=1;i<=n;i++)
{
printf("Process %d:",i);
scanf("%d",&b[i]); z[i]=b[i];
}
printf("\nEnter the time slice value:");
scanf("%d",&q);
rr();//no return type with no argument function average();
return 0; }
int rr()
{
   int max=0; max=b[1]; for(j=1;j<=n;j++) if(max<=b[j]) max=b[j];
if((max%q)==0)
r=(max/q);
else
```

```
    r=(max/q)+1;
    for(i=1;i<=r;i++)
    {
    printf("\n\nRound %d",i);
    for(j=1;j<=n;j++)
    {
       if(b[j]>0) {
          b[j]=b[j]-q;
       }
    if(b[j]<=0)
    {
       b[j]=0;
    printf("\nProcess %d is completed",j);
    }
    else
    {
    printf("\nProcess %d remaining time is %d",j,b[j]);
    }
    }
    }
    return 0;
    }
    int average()
    {
       for(i=1;i<=n;i++)
    { e=0;
    for(j=1;j<=r;j++)
    {
       if(z[i]!=0) {
    if(z[i]>=q)
    {
    m[i+e]=q; z[i]-=q; }
    else
    { m[i+e]=z[i];
    z[i]=0; } }
    else
    m[i+e]=0; e=e+n;
    } }
    for(i=2;i<=n;i++)
    for(j=1;j<=i-1;j++) avg=avg+m[j];
for(i=n+1;i<=r*n;i++)
{
   if(m[i]!=0) {
      for(j=i-(n-1);j<=i-1;j++)
avg=m[j]+avg;
} }
f=avg/n;
printf("\n\nTotal Waiting:%d",avg);
printf("\n\nAverage Waiting Time:%f\n",f);
printf("\n\t\t\t\tGANTT CHART");
printf("\n\t\t\t\t***********\n\n");
for(i=1;i<=r*n;i++) {
   if(m[i]!=0) {
        if(i%n==0) {
            printf("P%d",(i%n)+(n));
}
else
{
    printf("P%d",(i%n)); for(j=1;j<=m[i];j++)
    printf("_",n);
}
} }
printf("\n\n\n");
return 0; }
```

**Output:**



```
                    ROUND ROBIN

                    *****************
Enter how many jobs:3

Enter burst time for corresponding job..

Process 1:7
Process 2:4
Process 3:3

Enter the time slice value:3


Round 1
Process 1 remaining time is 4
Process 2 remaining time is 1
Process 3 is completed

Round 2
Process 1 remaining time is 1
Process 2 is completed
Process 3 is completed

Round 3
Process 1 is completed
Process 2 is completed
Process 3 is completed
```

**Result:**