Design, develop and implement a C/C++/Java program to simulate the working of Shortest remaining time and Round Robin (RR) scheduling algorithms. Experiment with different quantum sizes for RR algorithm.

PROGRAM 7

```
import java.io.IOException;

import java.util.Scanner;

public class srt {



        /**

         * @param args

         */

public static void main(String[] args) throws IOException{

                // TODO Auto-generated method stub

                int choice;

                  Scanner scanner=new Scanner(System.in);



                  boolean l=true;

                  while(l){

                          System.out.println("1:SRTF\n2:ROUND ROBIN\n3:EXIT");

                          System.out.println("Enter your choice:");

                          choice=scanner.nextInt();

                  switch(choice)

                  {

                  case 1:

                  int n,serviceTime=0,tot=0;

                float avgwt=0,avgta=0;

    System.out.println("Enter the number of processes:");
```

```
n=scanner.nextInt();
int bt[]=new int[n];
int at[]=new int[n];
int ct[]=new int[n];
int ta[]=new int[n];
int wt[]=new int[n];
int pid[]=new int[n];
int k[]=new int[n];
int flag[]=new int[n];
System.out.println("Enter the burst time of processes:");
for(int i=0;i<n;i++)
        bt[i]=scanner.nextInt();
System.out.println("Enter the arrival time of processes:");
for(int i=0;i<n;i++)
        at[i]=scanner.nextInt();
for(int i=0;i<n;i++)
{
        pid[i]=i+1;
        k[i]=bt[i];
        flag[i]=0;
}



        while(true)
        {
                int min=99,c=n;
                if(tot==n)
                        break;
```

```
            for(int i=0;i<n;i++)
            {
                        if((at[i]<=serviceTime) && (flag[i]==0) && bt[i]<min)
                        {
                                    min=bt[i];
                                    c=i;
                        }
            }

            if(c==n)
                        serviceTime++;
            else
            {
            bt[c]--;
            serviceTime++;
            if(bt[c]==0)
            {
                        ct[c]=serviceTime;
                        flag[c]=1;
                        tot++;
            }
  }
}
for(int i=0;i<n;i++)
{
           ta[i]=ct[i]-at[i];
           wt[i]=ta[i]-k[i];
           avgwt+=wt[i];
```

```
                    avgta+=ta[i];
            }
            System.out.println("Process\t Arival Time\t Burst Time\t Waiting Time\t Turn
around time\t Completion time");
            for(int i=0;i<n;i++)
              System.out.println((i+1)+"\t\t"+
at[i]+"\t\t"+k[i]+"\t\t"+wt[i]+"\t\t"+ta[i]+"\t\t\t"+ct[i]);
            System.out.println("\nAverage waiting time:"+avgwt/n);
            System.out.println("Average turn around time:"+avgta/n+"\n");
               break;
                    case 2:
                    {
                              int n1,i,j,tq;
                              float avgwt1=0,avgta1=0;
                System.out.println("Enter the number of processes:");
                      n1=scanner.nextInt();
                      int bt1[]=new int[n1];
                        //int at[]=new int[n];
                        //int ct[]=new int[n];
                        int ta1[]=new int[n1];
                        int wt1[]=new int[n1];
                        int copy[]=new int[n1];
                      System.out.println("Enter the time quantum:");
                            tq=scanner.nextInt();

                            System.out.println("Enter the burst time for processes:");
                            for(i=0;i<n1;i++)
                               bt1[i]=scanner.nextInt();
```

```
/*System.out.println("Enter the arrival time of processes:");
for(int j1=0;j1<n;j1++)
 at[j1]=scanner.nextInt();*/

for(i=0;i<n1;i++)
   copy[i]=bt1[i];

 for(i=0;i<n1;i++)
 {
  if(bt1[i]>tq)
    {
     bt1[i]=bt1[i]-tq;
     for(j=0;j<n1;j++)
            if(i!=j && bt1[j]!=0)
                        wt1[j]=wt1[j]+tq;
     }
  else
    {
     for(j=0;j<n1;j++)
            if(i!=j && bt1[j]!=0)
                        wt1[j]=wt1[j]+bt1[i];
     bt1[i]=0;
   }
}
for(i=0;i<n1;i++)
{
  ta1[i]=copy[i]+wt1[i];
```

```
                    }



                    System.out.println("Process\t Turn around time\t Burst Time\t");
                     for(int i1=0;i1<n1;i1++)
                System.out.println((i1+1)+"\t\t"+ ta1[i1]+"\t\t"+copy[i1]);


              for(int i1=0;i1<n1;i1++)
              {
                     ta1[i1]=wt1[i1]+copy[i1];
                     //wt[i1]=ta[i1]-k[i1];
                     avgwt1+=wt1[i1];
                     avgta1+=ta1[i1];
              }
              System.out.println("\nAverage waiting time:"+avgwt1/n1);
              System.out.println("Average turn around time:"+avgta1/n1+"\n");
            }break;
             case 3:l=false;break;
             default:System.out.println("invalid input");
                   break;
}//--------------------class
             }
}
}
```