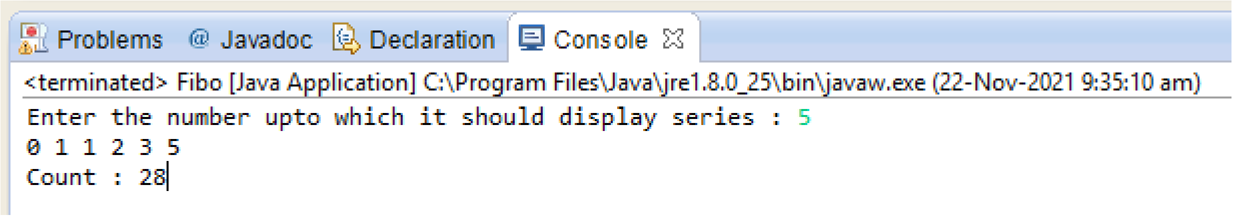
LAB CYCLE 1

Aim : To write a java to analyse time complexity of Fibonacci series

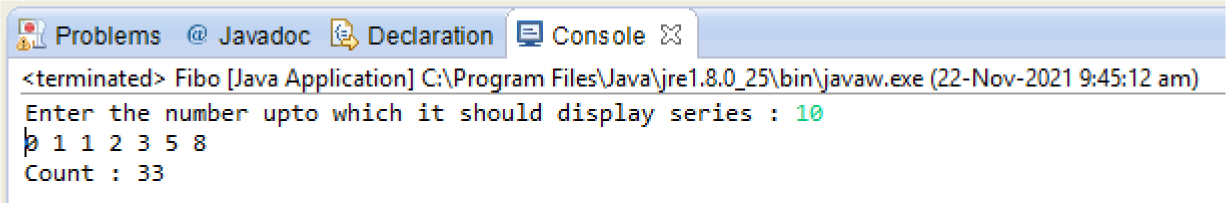
Program :

```
import java.util.Scanner;
public class Main
{
	public static void main(String[] args)
	{
		Scanner sc = new Scanner(System.in);
		 int n,f0=0,f1=1,f2,count = 0;
		 count += 3;

		 System.out.print("Enter the number upto which it should display series : ");
		 count += 1;

		n = sc.nextInt();
		count += 1;

		System.out.print(f0+" "+f1+" ");
		count += 1;

		f2=f0+f1;
		count +=1;

		while(f2<=n)
		{
		    System.out.print(f2+" ");

		    f0=f1;
		    f1=f2;
		    f2=f0+f1;

		    count += 5;
		}
		count += 1;

		System.out.println("\nCount : "+count);
		count += 1;

	}
}
```

OUTPUT :



```
<terminated> Fibo [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (22-Nov-2021 9:35:10 am)
Enter the number upto which it should display series : 5
0 1 1 2 3 5
Count : 28
```



```
<terminated> Fibo [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (22-Nov-2021 9:45:12 am)
Enter the number upto which it should display series : 10
0 1 1 2 3 5 8
Count : 33
```

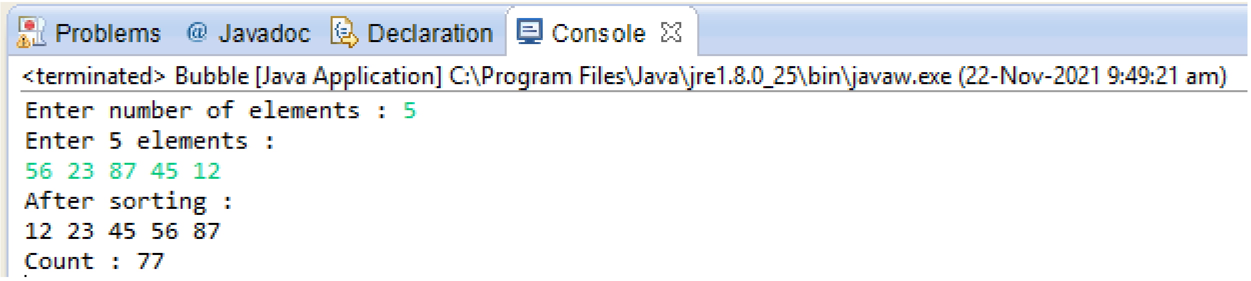
Aim : To write a java program to analyse time complexity of Bubble sort

Program :

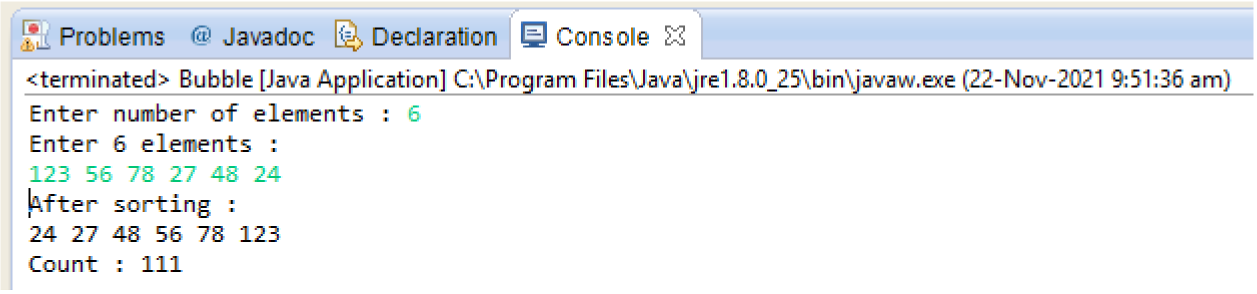
```
import java.util.Scanner;
public class Main
{
        public static void main(String[] args)
        {
                Scanner sc = new Scanner(System.in);
                int n,tmp,i,j,count=0;

                System.out.print("Enter number of elements : ");
                count += 1;
                n = sc.nextInt();
                count++;
                int arr[] = new int[n];
                count++;

                System.out.println("Enter "+n+" elements : ");
                count++;

                for(i=0;i<n;i++)
                {
                        arr[i] = sc.nextInt();
                        count += 2;
                }
                count++;
                for(i=0;i<n-1;i++)
                {
                   count++;
                   for(j=0;j<n-i-1;j++)
                   {
                      count+= 2;
                      if(arr[j] > arr[j+1])
                      {
                         tmp = arr[j];
                         arr[j] = arr[j+1];
                         arr[j+1] = tmp;
                         count += 3;
                      }
                   }
```

```
            count += 1;
        }
        count += 1;
        System.out.println("After sorting : ");
        count += 1;
        for(i=0;i<n;i++)
        {
            System.out.print(arr[i]+" ");
            count += 2;
        }
        count += 1;
        System.out.println("\nCount : "+count);
    }
}
```

OUTPUT :



```
<terminated> Bubble [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (22-Nov-2021 9:49:21 am)
Enter number of elements : 5
Enter 5 elements :
56 23 87 45 12
After sorting :
12 23 45 56 87
Count : 77
```



```
<terminated> Bubble [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (22-Nov-2021 9:51:36 am)
Enter number of elements : 6
Enter 6 elements :
123 56 78 27 48 24
After sorting :
24 27 48 56 78 123
Count : 111
```

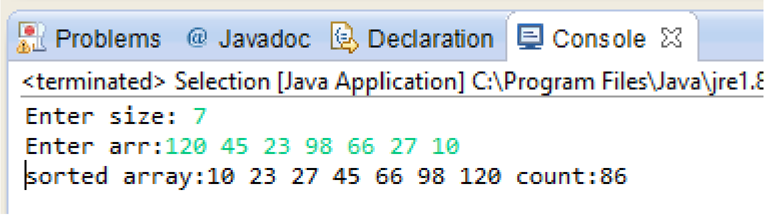
Aim: To write a java program to analyse time complexity of selection sort Program.

Program:

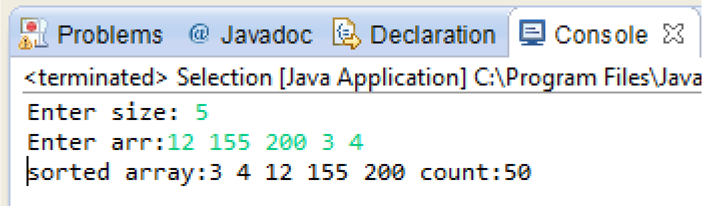
```
import java.util.Scanner;
public class Selection{
static int c=0;
public static void main(String[] args){
    Scanner sc=new Scanner(System.in);
    int n,m;
    System.out.print("Enter size: ");
    n=sc.nextInt();
    int a[]=new int[n];
    System.out.print("Enter arr:");
    for(int i=0;i<n;i++)
    {
        a[i]=sc.nextInt();
        }
    for(int i=0;i<n-1;i++)
    {
        c++;
        m=i;
        c++;
        for(int j=i+1;j<n;j++)
        {
            c++;
            if(a[j]<a[m])
            {
                m=j;
                c++;
                }
            c++;
            }
        c++;
        int temp=a[m];
        c++;
        a[m]=a[i];
        c++;
        a[i] = temp;
        c++;
       }
    c++;

    System.out.print("sorted array:");
    for(int i=0;i<n;i++)
    {
        System.out.print(a[i]+" ");
    }
```

```
        System.out.print("count:"+c);
        }
}
```

OUTPUT:



```
<terminated> Selection [Java Application] C:\Program Files\Java\jre1.8
Enter size: 7
Enter arr:120 45 23 98 66 27 10
sorted array:10 23 27 45 66 98 120 count:86
```



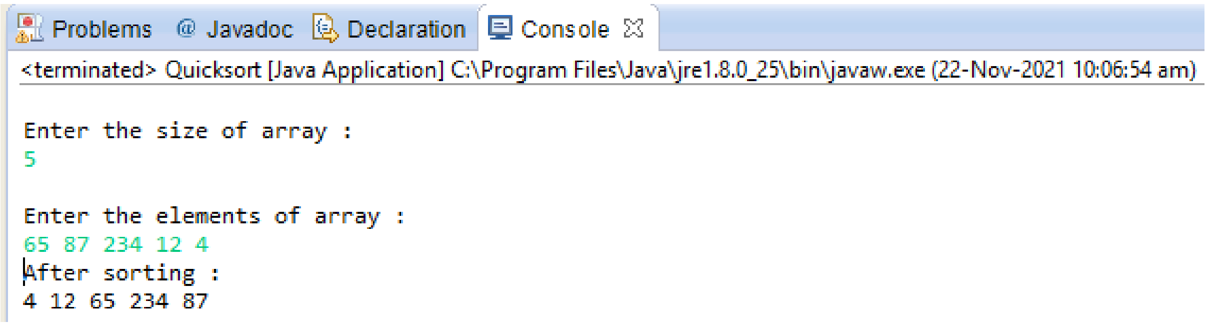
```
<terminated> Selection [Java Application] C:\Program Files\Java
Enter size: 5
Enter arr:12 155 200 3 4
sorted array:3 4 12 155 200 count:50
```

Aim : To write a java program to sort a list of integers in ascending order using Quicksort algorithm

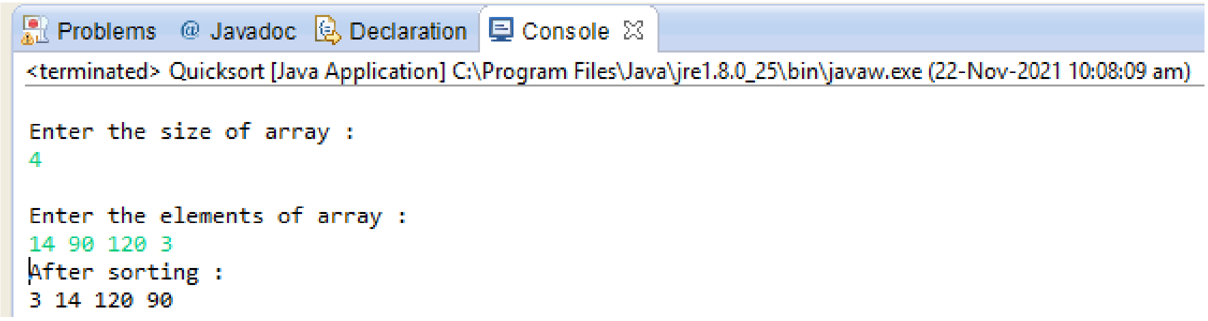
Program :

```
import java.util.Scanner;
public class Quicksort {
    static int Partition(int arr[],int m,int p)
    {
        int pivot =arr[m];
        int i=m+1,j=p;

        while(i <= j)
        {
            while(arr[i] < pivot && i <= p)
            {
                i++;
            }

            while(arr[j] > pivot && j > m)
            {
                j--;
            }
            if(i < j)
            {
                int temp =arr[i];
                arr[i] =arr[j];
                arr[j] = temp;
            }
        }
        arr[m] = arr[j];
        arr[j] = pivot;
        return j;
    }
    static void QuickSortAlgo(int arr[],int p,int q)
    {
        if(p < q)
        {

            int j = Partition(arr,p,q);
            QuickSortAlgo(arr,p,j-1);
            QuickSortAlgo(arr,j+1,p);
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int i,size;
        System.out.println("\nEnter the size of array : ");
        size = sc.nextInt();
        int arr[] = new int[size];
        System.out.println("\nEnter the elements of array : ");
        for(i=0;i<size;i++)
            arr[i] = sc.nextInt();
        QuickSortAlgo(arr,0,size-1);

        System.out.println("After sorting : ");
        for(i=0;i<size;i++)
            System.out.print(arr[i]+" ");

    }

}
```

OUTPUT :



```
<terminated> Quicksort [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (22-Nov-2021 10:06:54 am)

Enter the size of array :
5

Enter the elements of array :
65 87 234 12 4
After sorting :
4 12 65 234 87
```



```
<terminated> Quicksort [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (22-Nov-2021 10:08:09 am)

Enter the size of array :
4

Enter the elements of array :
14 90 120 3
After sorting :
3 14 120 90
```

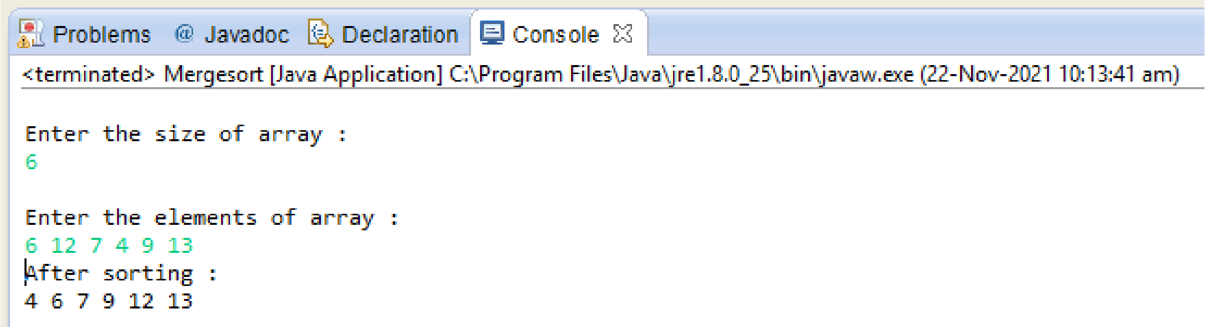
Aim : To write a java program to sort a list of integers in ascending order using Mergesort algorithm

Program :

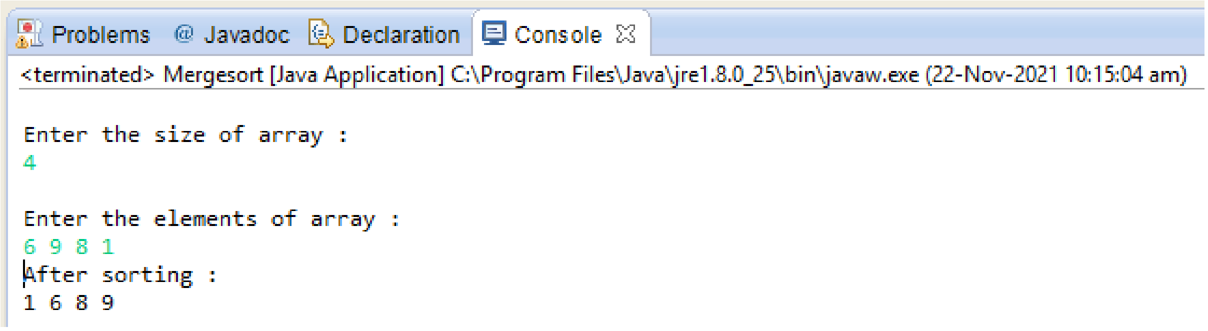
```
import java.util.Scanner;

public class MergeSort {

        static void MergeSortAlgo(int arr[],int low,int high)
        {
                if(low < high)
                {
                        int mid = (low+high)/2;
                        MergeSortAlgo(arr,low,mid);
                        MergeSortAlgo(arr,mid+1,high);
                        Merge(arr,low,mid,high);
                }
        }
        static void Merge(int arr[],int low,int mid,int high)
        {
                int l = low,index=low,h = mid+1;
                int b[] = new int[high+1];

                while(l <= mid && h <= high)
                {
                        if(arr[l] <= arr[h])
                                b[index] = arr[l++];
                        else
                                b[index] = arr[h++];
                        index++;

                }

                if(l > mid)
                {
                        for(int k=h;k<=high;k++)
                                b[index++] = arr[k];
                }
                else
                {
                        for(int k=l;k<=mid;k++)
                                b[index++] = arr[k];
                }
                for(int k=low;k<= high;k++)
                        arr[k] = b[k];
        }
        public static void main(String[] args)
```

```
    {
            Scanner sc = new Scanner(System.in);
            int i,size;
            System.out.println("\nEnter the size of array : ");
            size = sc.nextInt();
            int arr[] = new int[size];
            System.out.println("\nEnter the elements of array : ");
            for(i=0;i<size;i++)
                    arr[i] = sc.nextInt();
            MergeSortAlgo(arr,0,size-1);
            System.out.println("After sorting : ");
            for(i=0;i<size;i++)
                    System.out.print(arr[i]+" ");
     }
}
```

OUTPUT :



```
<terminated> Mergesort [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (22-Nov-2021 10:13:41 am)

Enter the size of array : 
6

Enter the elements of array : 
6 12 7 4 9 13
After sorting : 
4 6 7 9 12 13
```



```
<terminated> Mergesort [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (22-Nov-2021 10:15:04 am)

Enter the size of array : 
4

Enter the elements of array : 
6 9 8 1
After sorting : 
1 6 8 9
```

LAB CYCLE 2

Aim : To write a java program to implement greedy algorithm for job sequencing with deadlines.

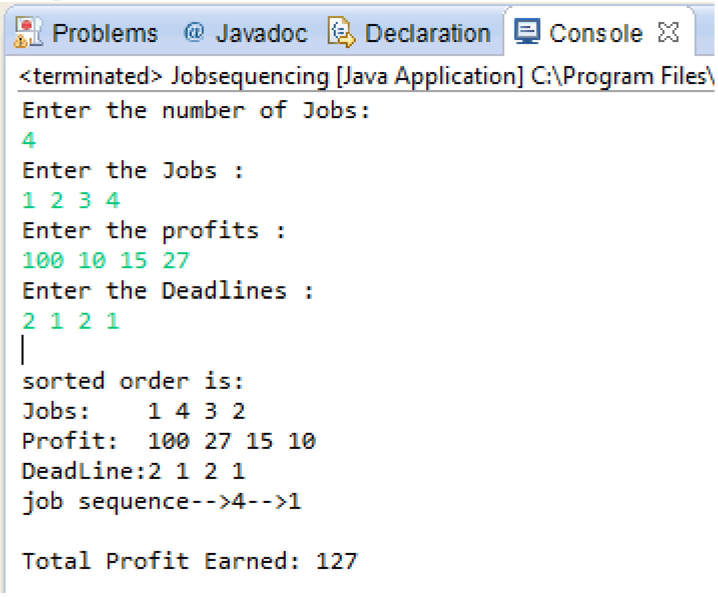
Program :

```
import java.util.Scanner;
public class Jobsequencing{
        public static void main(String[] args)
                          {
                           Scanner sc=new Scanner(System.in);
                           System.out.println("Enter the number of Jobs:");
                           int n=sc.nextInt();
                           String a[]=new String[n];
                           int b[]=new int[n];
                           int c[]=new int[n];
                           System.out.println("Enter the Jobs : ");
                           for(int i=0;i<n;i++)  {
                             a[i]=sc.next();
                           }
                           System.out.println("Enter the profits : ");
                           for(int i=0;i<n;i++)  {
                             b[i]=sc.nextInt();
                           }
                           System.out.println("Enter the Deadlines : ");
                           for(int i=0;i<n;i++)  {
                             c[i]=sc.nextInt();
                           }
                           for(int i=0;i<n-1;i++)
                           {
                             for(int j=i+1;j<n;j++)
                             {
                                 if(b[i]<b[j])
                                 {
                                     int temp=b[i];
                                     b[i]=b[j];
                                      b[j]=temp;
                                     temp=c[i];
                                     c[i]=c[j];
                                      c[j]=temp;
                                     String temp1=a[i];
                                     a[i]=a[j];
                                      a[j]=temp1;
                                 }
                             }
                           }
                           System.out.println();
                           System.out.println("sorted order is:");
                           System.out.print("Jobs:    ");
```

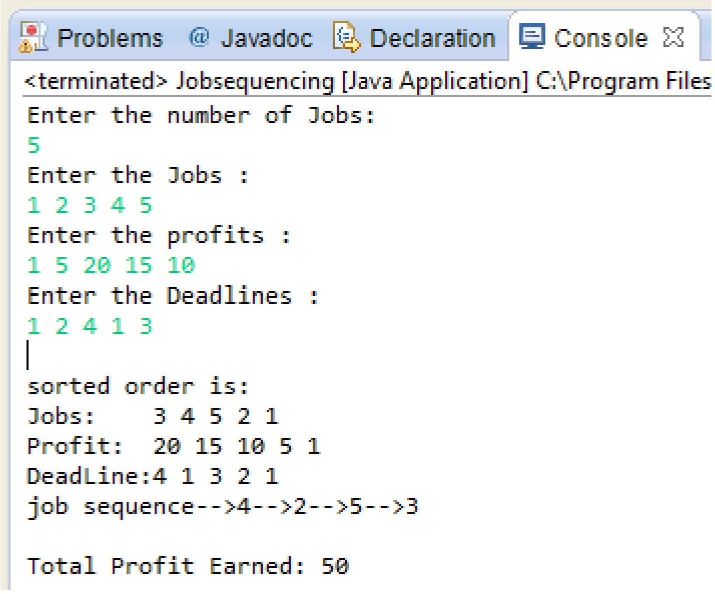
```
for(int i=0;i<n;i++)
{
  System.out.print(a[i]+" ");
}
System.out.println();
System.out.print("Profit:  ");
for(int i=0;i<n;i++)
{
  System.out.print(b[i]+" ");
}
System.out.println();
System.out.print("DeadLine:");
for(int i=0;i<n;i++)
{
  System.out.print(c[i]+" ");
}
System.out.println();
int max=c[0];
for(int i=0;i<n;i++)
{
  if(c[i]>max)
  {
    max=c[i];
  }
}
String x[]=new String[max];
int x1[]=new int[max];
int profit=0;
for(int i=0;i<n;i++)
{
  int p1=c[i];
  p1=p1-1;
  if(x[p1]==null )
  {
    x[p1]=a[i];
    profit+=b[i];
  }
  else
  {
    while(p1!=-1)
    {
      if(x[p1]==null)
      {
        x[p1]=a[i];
        profit+=b[i];
        break;
```

```
                }
                 p1=p1-1;
              }
            }
          }
          System.out.print("job sequence");
          for(int i=0;i<max;i++)
          {
             System.out.print("-->"+x[i]);
          }
          System.out.println();
          System.out.print("\nTotal Profit Earned: "+profit);
        }
}
```

Output :



```
<terminated> Jobsequencing [Java Application] C:\Program Files\
Enter the number of Jobs:
4
Enter the Jobs :
1 2 3 4
Enter the profits :
100 10 15 27
Enter the Deadlines :
2 1 2 1

sorted order is:
Jobs:    1 4 3 2
Profit:  100 27 15 10
DeadLine:2 1 2 1
job sequence-->4-->1

Total Profit Earned: 127
```



```
<terminated> Jobsequencing [Java Application] C:\Program Files
Enter the number of Jobs:
5
Enter the Jobs :
1 2 3 4 5
Enter the profits :
1 5 20 15 10
Enter the Deadlines :
1 2 4 1 3

sorted order is:
Jobs:    3 4 5 2 1
Profit:  20 15 10 5 1
DeadLine:4 1 3 2 1
job sequence-->4-->2-->5-->3

Total Profit Earned: 50
```

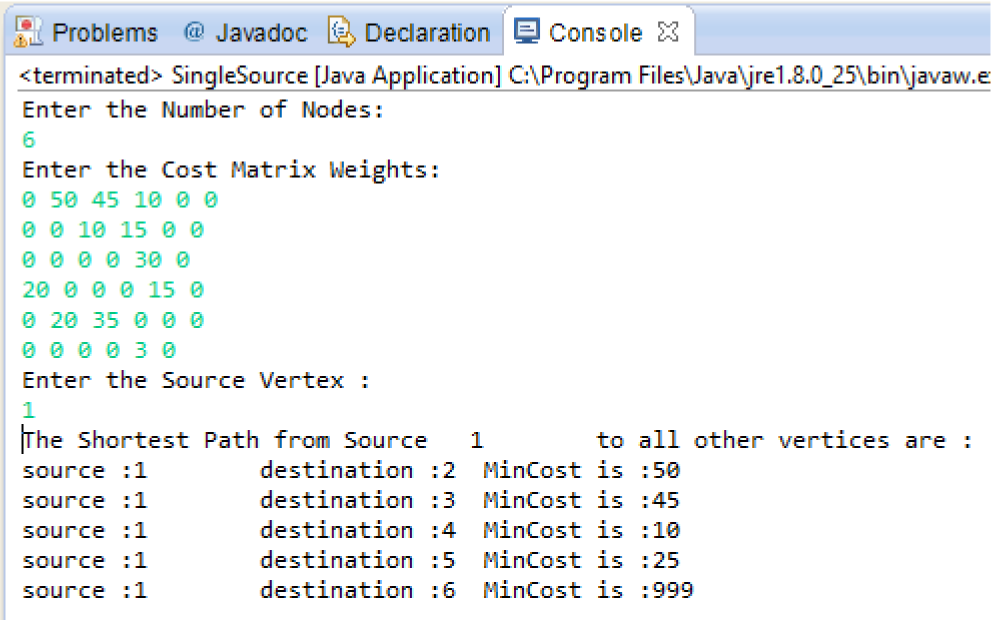
Aim : To write a java program to implement Dijkstra’salgorithm for the single source shortest path problem.

Program :

```
import java.util.*;
public class singlessp
{
    public int distance[] = new int[10];
    public int cost[][] = new int[10][10];
    public void calc(int n,int s)
    {
        int flag[] = new int[n+1];
        int i,minpos=1,k,c,minimum;
        for(i=1;i<=n;i++)
        {
            flag[i]=0;
            this.distance[i]=this.cost[s][i];
        }
        c=2;
        while(c<=n)
        {
            minimum=99;
            for(k=1;k<=n;k++)
            {
                if(this.distance[k]<minimum && flag[k]!=1)
                {
                    minimum=this.distance[i];
                    minpos=k;
                }
        }
        flag[minpos]=1;
        c++;
        for(k=1;k<=n;k++){
        if(this.distance[minpos]+this.cost[minpos][k] < this.distance[k] &&
        flag[k]!=1 )
        this.distance[k]=this.distance[minpos]+this.cost[minpos][k];
        }
    }
}
public static void main(String args[])
{
    int nodes,source,i,j;
    Scanner in = new Scanner(System.in);
    System.out.println("Enter the Number of Nodes \n");
```

```
        nodes = in.nextInt();
        singlessp d = new singlessp();
        System.out.println("Enter the Cost Matrix Weights: \n");
        for(i=1;i<=nodes;i++)
                for(j=1;j<=nodes;j++)
                {
                        d.cost[i][j]=in.nextInt();
                        if(d.cost[i][j]==0)
                                d.cost[i][j]=999;
                }
                System.out.println("Enter the Source Vertex :\n");
                source=in.nextInt();
                d.calc(nodes,source);
                System.out.println("The Shortest Path from Source \t"+source+"\t to all other
                vertices are : \n");
                for(i=1;i<=nodes;i++)
                        if(i!=source)
                System.out.println("source :"+source+"\t destination :"+i+"\t MinCost is
:"+d.distance[i]+"\t");
        }
}
```

Output :



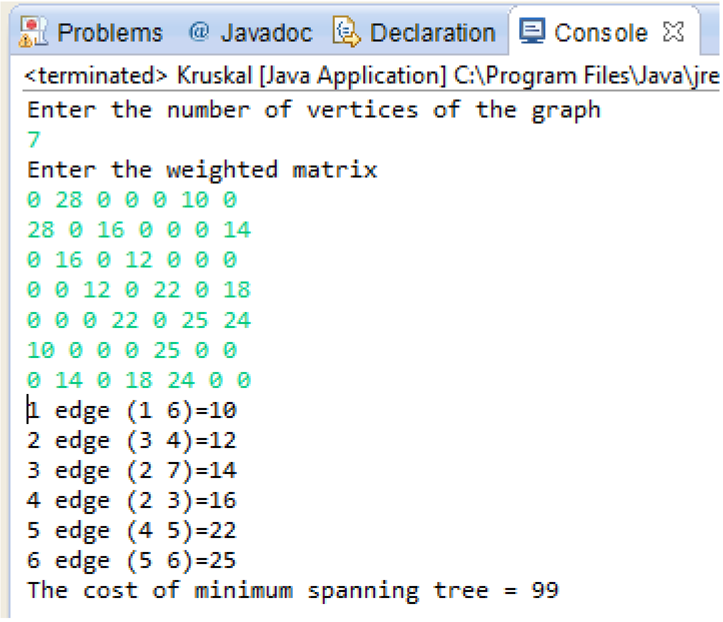
```
<terminated> SingleSource [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.e
Enter the Number of Nodes:
6
Enter the Cost Matrix Weights:
0 50 45 10 0 0
0 0 10 15 0 0
0 0 0 0 30 0
20 0 0 0 15 0
0 20 35 0 0 0
0 0 0 0 3 0
Enter the Source Vertex :
1
The Shortest Path from Source   1        to all other vertices are :
source :1        destination :2  MinCost is :50
source :1        destination :3  MinCost is :45
source :1        destination :4  MinCost is :10
source :1        destination :5  MinCost is :25
source :1        destination :6  MinCost is :999
```

Aim : To write a java program to implement kruskal’s algorithm to generate minimum spanning tree.

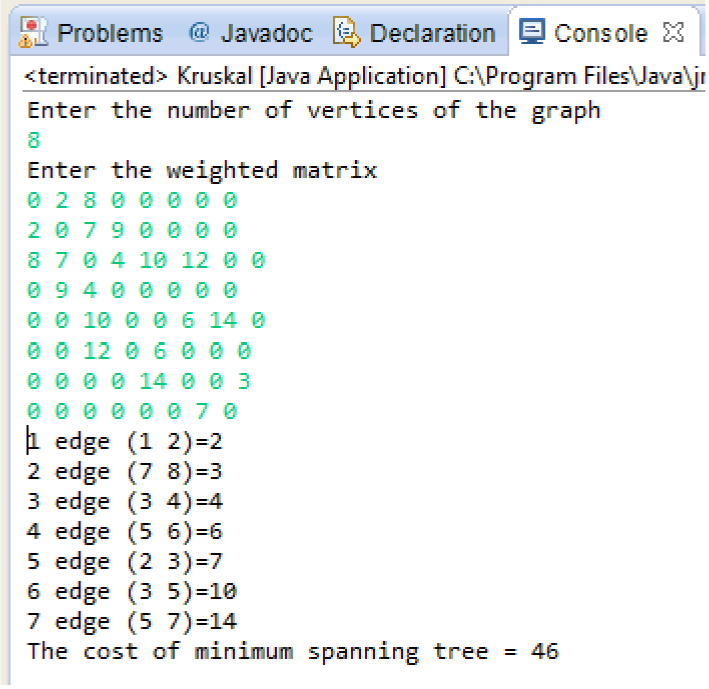
Program :

```
import java.util.Scanner;
public class krusk {
     int parent[]=new int[10];
     int find(int m)
     {
         int p=m;
         while(parent[p]!=0)
         p=parent[p];
         return p;
     }
     int union(int i,int j)
    {
      if(i!=j)
      {
         parent[j]=i;
      return 1;
      }
      return 0;
    }
     void krkl(int[][]A, int n)
    {
      int a=0,b=0,u=0,v=0,min,k=0,i,j,sum=0,ne=1,mincost=0;
       for(i=1;i<=n;i++)
         for(j=1;j<=n;j++)
            if(A[i][j]==0)
              {
                  A[i][j]=999;
              }
       while(ne<n)
       {
      for(i=1,min=999;i<=n;i++)
      {
      for(j=1;j<=n;j++)
      {
      if(A[i][j]<min)
      {
      min=A[i][j];
      a=u=i;
      b=v=j;
      }
      }
      }
      u=find(u);
           v=find(v);
           if(union(u,v)==1)
            {
          System.out.println(ne+++ “ edge ( “ +a+” “+b+”) = “+min);
          mincost+=min;
            }
           A[a][b]=A[b][a]=999;
          }
      System.out.println("The cost of minimum spanning tree = "+mincost);
}
public static void main(String[] args) {
       int a[][]=new int[10][10];
          int i,j;
        System.out.println("Enter the number of vertices of the graph");
        Scanner sc=new Scanner(System.in);
        int n;
        n=sc.nextInt();
        System.out.println("Enter the weighted matrix");
       for(i=1;i<=n;i++)
         for(j=1;j<=n;j++)
            a[i][j]=sc.nextInt();
         krusk k=new krusk();
         k.krkl(a,n);
       sc.close();
 }
}
```

Output :



```
<terminated> Kruskal [Java Application] C:\Program Files\Java\jre
Enter the number of vertices of the graph
7
Enter the weighted matrix
0 28 0 0 0 10 0
28 0 16 0 0 0 14
0 16 0 12 0 0 0
0 0 12 0 22 0 18
0 0 0 22 0 25 24
10 0 0 0 25 0 0
0 14 0 18 24 0 0
1 edge (1 6)=10
2 edge (3 4)=12
3 edge (2 7)=14
4 edge (2 3)=16
5 edge (4 5)=22
6 edge (5 6)=25
The cost of minimum spanning tree = 99
```

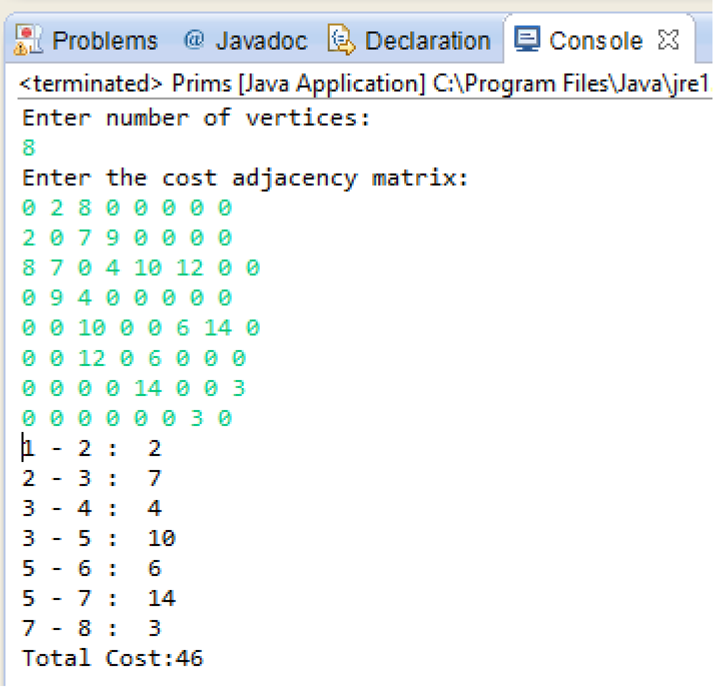


```
<terminated> Kruskal [Java Application] C:\Program Files\Java\jr
Enter the number of vertices of the graph
8
Enter the weighted matrix
0 2 8 0 0 0 0 0
2 0 7 9 0 0 0 0
8 7 0 4 10 12 0 0
0 9 4 0 0 0 0 0
0 0 10 0 0 6 14 0
0 0 12 0 6 0 0 0
0 0 0 0 14 0 0 3
0 0 0 0 0 0 7 0
1 edge (1 2)=2
2 edge (7 8)=3
3 edge (3 4)=4
4 edge (5 6)=6
5 edge (2 3)=7
6 edge (3 5)=10
7 edge (5 7)=14
The cost of minimum spanning tree = 46
```

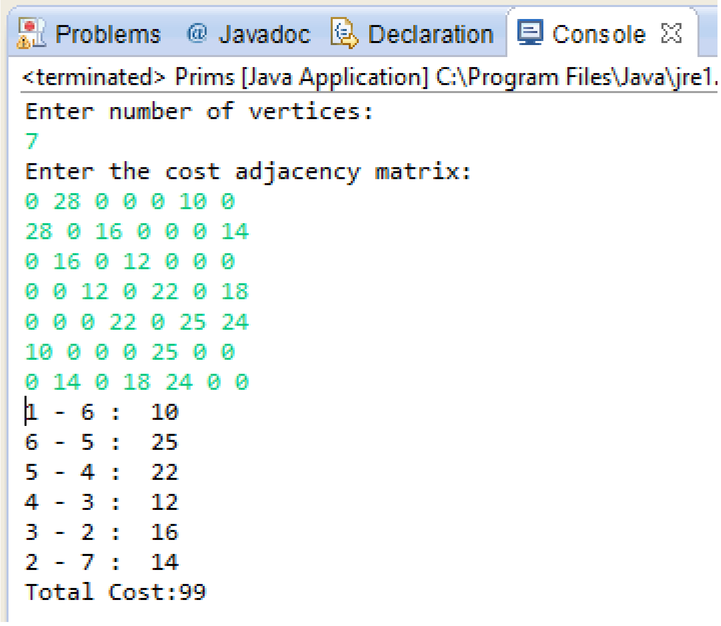
Aim : To write a java program to implement prim’s algorithm to generate minimum spanning tree.
Program:

```
import java.util.*;
public class prims {
        public void PA(int G[][],int n){
                int INF = 9999999;
                int v,sum=0;
                boolean[] selected = new boolean[n];
                for(int i=0;i<n;i++) {
                        selected[i] = false;
        }
                v = 0;
                selected[0] = true;
                while(v<(n-1))
                {
                        int x=0,y=0,min = INF;
                        for(int i=0;i<n;i++)
                        {
                                if(selected[i]== true)
                                {
                                        for(int j=0;j<n;j++)
                                        {
                                                if(!selected[j] && G[i][j]!=0)
                                                {
                                                        if(min>G[i][j])
                                                        {
                                                                min = G[i][j];
                                                                x=i;
                                                                y=j;
                                                        }
                                                }
                                        }
                                }
                        }
                        sum += G[x][y];
                        System.out.println((x+1) + " - " + (y+1) + " :  " + G[x][y]);
                        selected[y] = true;
                        v++;
                }
                System.out.println("Total Cost:"+sum);
        }
        public static void main(String[] args) {
                int V;
                prims p = new prims();
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number of vertices:");
        V = sc.nextInt();
        int arr[][] = new int[V][V];
        System.out.println("Enter the cost adjacency matrix:");
        for(int i=0;i<V;i++) {
            for(int j=0;j<V;j++)
            {
                arr[i][j] = sc.nextInt();
            }
        }
        p.PA(arr,V);
    }
}
```

Output :



```
<terminated> Prims [Java Application] C:\Program Files\Java\jre1
Enter number of vertices:
8
Enter the cost adjacency matrix:
0 2 8 0 0 0 0 0
2 0 7 9 0 0 0 0
8 7 0 4 10 12 0 0
0 9 4 0 0 0 0 0
0 0 10 0 0 6 14 0
0 0 12 0 6 0 0 0
0 0 0 0 14 0 0 3
0 0 0 0 0 0 3 0
1 - 2 :  2
2 - 3 :  7
3 - 4 :  4
3 - 5 :  10
5 - 6 :  6
5 - 7 :  14
7 - 8 :  3
Total Cost:46
```



```
<terminated> Prims [Java Application] C:\Program Files\Java\jre1.
Enter number of vertices:
7
Enter the cost adjacency matrix:
0 28 0 0 0 10 0
28 0 16 0 0 0 14
0 16 0 12 0 0 0
0 0 12 0 22 0 18
0 0 0 22 0 25 24
10 0 0 0 25 0 0
0 14 0 18 24 0 0
1 - 6 :  10
6 - 5 :  25
5 - 4 :  22
4 - 3 :  12
3 - 2 :  16
2 - 7 :  14
Total Cost:99
```