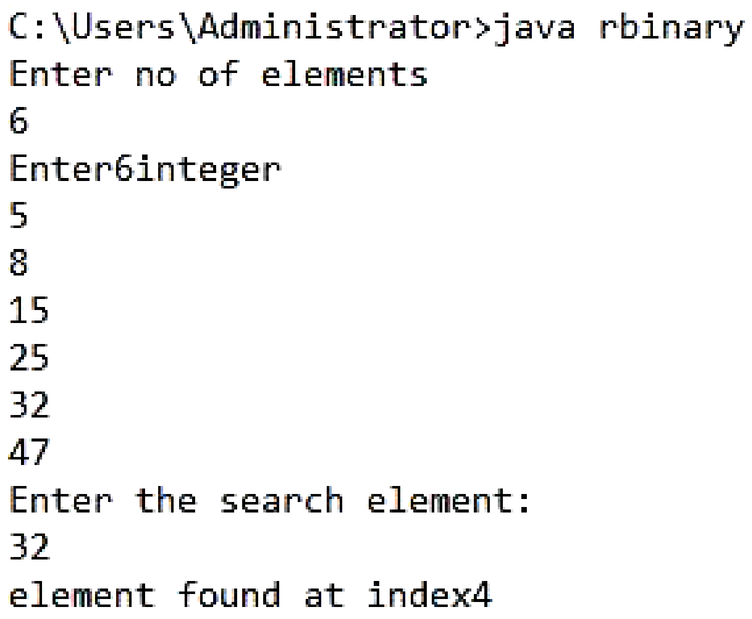
```
//Binary Search

import java.util.*;
public class rbinary
{
private static Object ob;
int binarysearch(int a[],int low,int high,int x)
{
int mid;
if(low<=high)
{
mid=(low+high)/2;
if(a[mid]==x)
return mid;
if(x>a[mid])
return binarysearch(a,mid+1,high,x);
else
return binarysearch(a,low,mid-1,x);
}
return-1; }
public static void main(String[] args)
{
rbinary ob=new rbinary();
Scanner in=new Scanner(System.in);
int i,a[],n,x,result;
System.out.println("Enter no of elements");
n=in.nextInt();
a=new int[n];
System.out.println("Enter"+n+"integer");
for(i=0;i<n;i++)
a[i]=in.nextInt();
System.out.println("Enter the search element:");
x=in.nextInt();
result=ob.binarysearch(a,0,n-1,x);
if(result==-1)
System.out.println("element not found");
else
System.out.println("element found at index"+result);
}
}
```

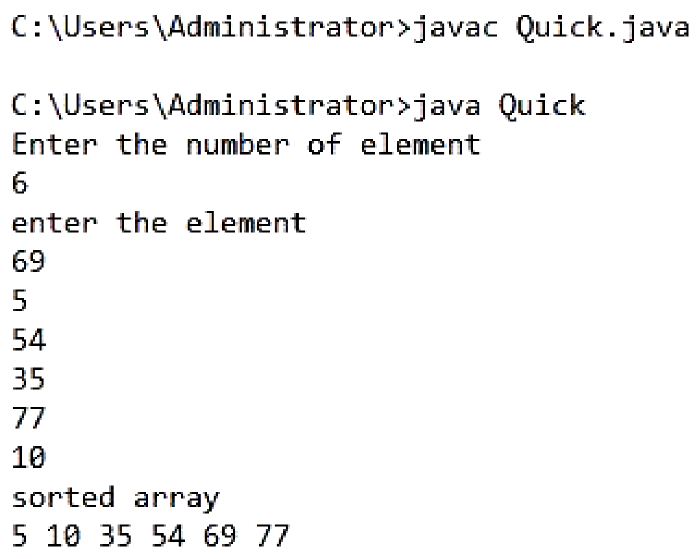
OUTPUT



```
C:\Users\Administrator>java rbinary
Enter no of elements
6
Enter6integer
5
8
15
25
32
47
Enter the search element:
32
element found at index4
```

```
//QuickSort

import java .util.*;
class Quick
{
int partition(int a[],int low,int high)
{
int i,j,pivot;
pivot=a[low];
i=low+1;j=high;
while(i<=j)
{
while((a[i]<pivot)&&(i<=high))
i++;
while(a[j]>pivot&&(j>low))
j--;
if(i<j)
{
int p=a[i];
a[i]=a[j];
a[j]=p;
}}
a[low]=a[j];
a[j]=pivot;
return j;
}
void sort (int a[],int low,int high)
{
if(low<high)
{
int j=partition(a,low,high);
sort(a,low,j-1);
sort(a,j+1,high);
}}
public static void main(String args[])
{
int i,n;
Quick ob=new Quick();
Scanner in=new Scanner(System.in);
System.out.println("Enter the number of element");
n=in.nextInt();
int a[]=new int [n+1];
System.out.println("enter the element");
for(i=0;i<n;i++)
a[i]=in.nextInt();
ob.sort(a,0,n-1);
System.out.println ("sorted array");
for(i=0;i<n;i++)
System.out.print(a[i]+ " ");
}}
```

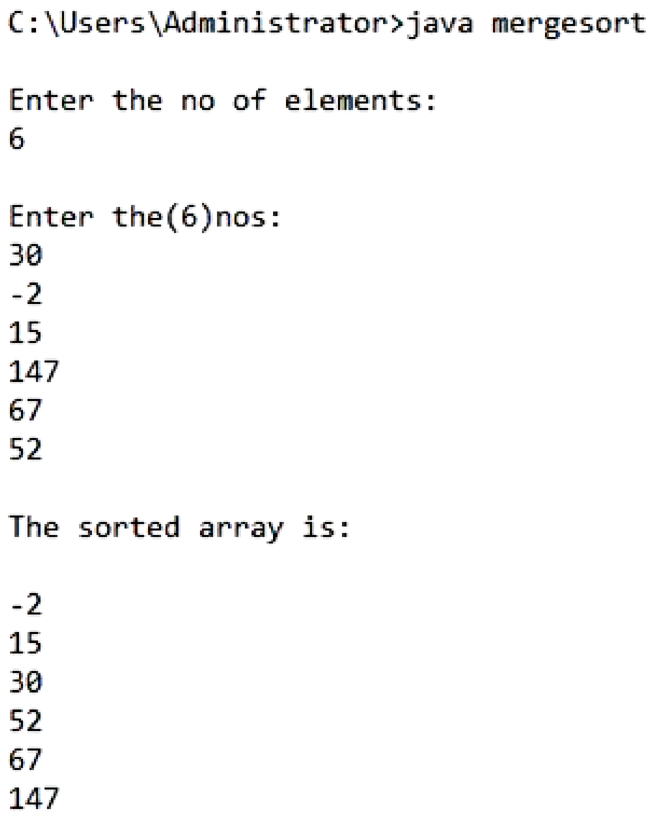
OUTPUT



```
C:\Users\Administrator>javac Quick.java

C:\Users\Administrator>java Quick
Enter the number of element
6
enter the element
69
5
54
35
77
10
sorted array
5 10 35 54 69 77
```

```
//Mergesort

import java.util.*;
public class mergesort
{
public static int a[]=new int[50];
public static void mergesort(int low,int high)
{
int mid;
if(low<high)
{
mid=(low+high)/2;
mergesort(low,mid);
mergesort(mid+1,high);
merge(low,mid,high);
}
}
public static void merge(int low,int mid,int high)
{
int h,i,j,k;
int b[]=new int[50];
h=low; i=low; j=mid+1;
while((h<=mid)&&(j<=high))
{
if(a[h]<=a[j])
{
b[i]=a[h];
h++;
}
else
{
b[i]=a[j];
j++;
}
i++;
}
if(h>mid)
{
for(k=j;k<=high;k++)
{
b[i]=a[k];
i++;
}
}
else
{
for(k=h;k<=mid;k++)
{
b[i]=a[k];
i++;
}
}
for(k=low;k<=high;k++)
a[k]=b[k];
}
public static void main(String []args)
{
int num,i;
System.out.println();
System.out.println("Enter the no of elements:");
num=new Scanner(System.in).nextInt();
System.out.println();
System.out.println("Enter the("+num+")nos:");
for(i=1;i<=num;i++)
{
a[i]=new Scanner(System.in).nextInt();
}
mergesort(1,num);
System.out.println();
System.out.println("The sorted array is:");
System.out.println();
for(i=1;i<=num;i++)
System.out.println(a[i]+" ");
}
}
```

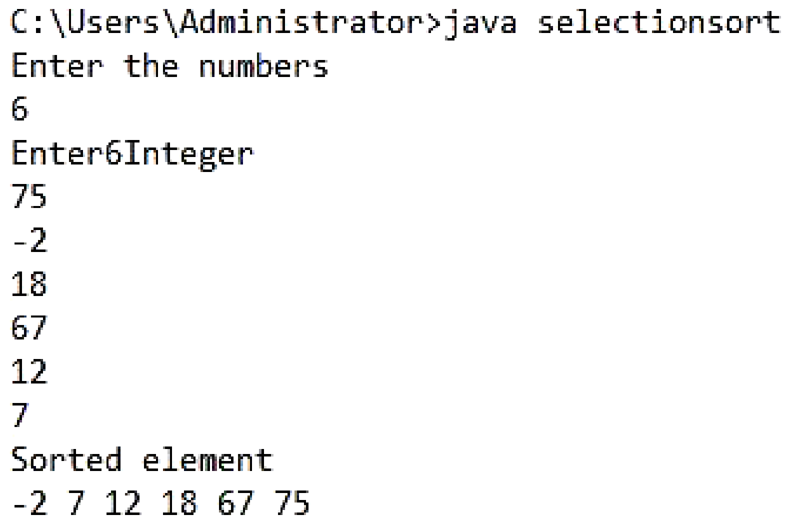
OUTPUT



```
C:\Users\Administrator>java mergesort

Enter the no of elements:
6

Enter the(6)nos:
30
-2
15
147
67
52

The sorted array is:

-2
15
30
52
67
147
```

```
//Selection sort

import java.io.*;
import java.util.*;
public class selectionsort
{
public static void main(String[] args)
{
int a[],i,j,n,min,pos,t;
Scanner s=new Scanner(System.in);
System.out.println("Enter the numbers");
n=s.nextInt();
a=new int[n+1];
System.out.println("Enter" +n+ "Integer");
for(i=1;i<=n;i++)
{
a[i]=s.nextInt();
}
for(i=1;i<=n;i++)
{
min=a[i];
pos=i;
for(j=i+1;j<=n;j++)
{
if(min>a[j]){
min=a[j];
pos=j;
}}
t=a[i];
a[i]=min;
a[pos]=t;
}
System.out.println("Sorted element");
for(i=1;i<=n;i++)
{
System.out.print(a[i]+" ");
}
}
}
```

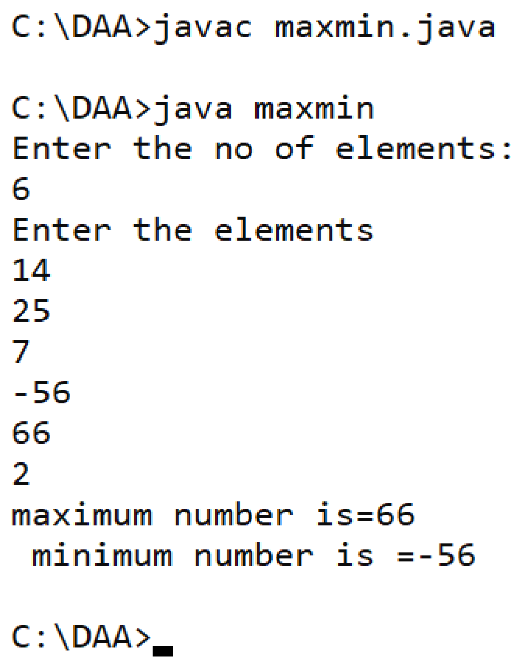
OUTPUT



```
C:\Users\Administrator>java selectionsort
Enter the numbers
6
Enter6Integer
75
-2
18
67
12
7
Sorted element
-2 7 12 18 67 75
```

```
//Maximum and Minimum

import java.util.*;
public class maxmin
{
static int a[];
static int max = Integer.MIN_VALUE,min = Integer.MAX_VALUE;
void maxmin (int i,int j)
{
int min1,max1,mid;
if(i==j)
min=max=a[i];
else
{
if(i==(j-1))
{
if(a[i]<a[j])
{
max=a[j];
min=a[i];
}
else
{
min=a[i];
max=a[j];
}
}
else
{
mid=(i+j)/2;
maxmin(i,mid);
max1=max;
min1=min;
maxmin(mid+1,j);
if(max<max1)
max=max1;
if(min>min1)
min=min1;
}
}
}
public static void main(String[] args)
{
maxmin ob=new maxmin();
int n,i;
Scanner in=new Scanner(System.in);
System.out.println("Enter the no of elements:");
n=in.nextInt();
a=new int[n];
System.out.println("Enter the elements");
for(i=0;i<n;i++)
a[i]=in.nextInt();
ob.maxmin(0,n-1);
System.out.println("maximum number is="+max+"\n minimum number is ="+min);
}
}
```

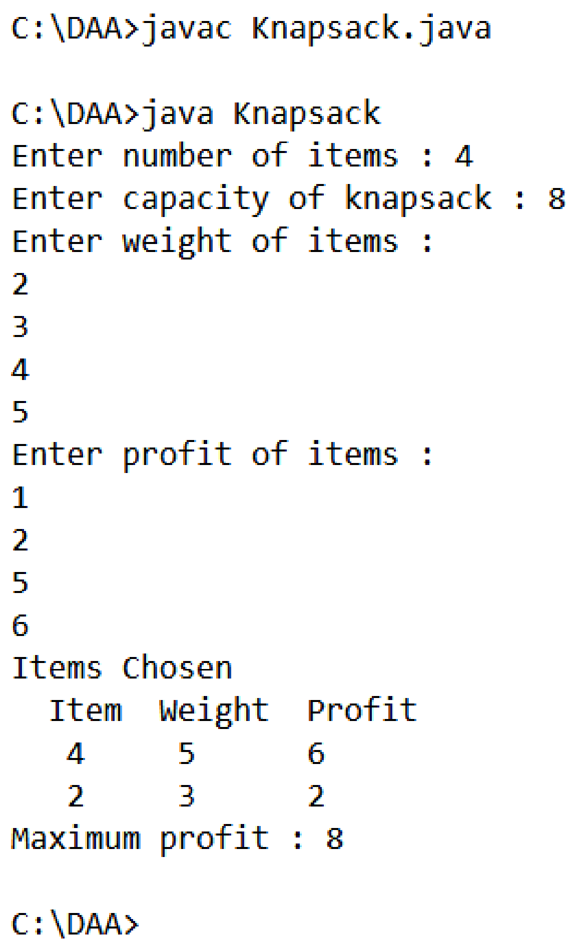
OUTPUT



```
C:\DAA>javac maxmin.java

C:\DAA>java maxmin
Enter the no of elements:
6
Enter the elements
14
25
7
-56
66
2
maximum number is=66
 minimum number is =-56

C:\DAA>
```

```
//Knapsack

import java.util.Scanner;
public class Knapsack
{
 private int n,W;
 private int w[],v[];
 private int V[][];
 private void initialize()
 {
  Scanner sc = new Scanner(System.in);
  System.out.print("Enter number of items : ");
  n = sc.nextInt();
  System.out.print("Enter capacity of knapsack : ");
  W = sc.nextInt();
  w = new int[n];
  v = new int[n];
  System.out.println("Enter weight of items : ");
  for(int i = 0; i < n; i++)
  {
   w[i] = sc.nextInt();
  }
  System.out.println("Enter profit of items : ");
  for(int i = 0; i < n; i++)
  {
   v[i] = sc.nextInt();
  }
  V = new int[n+1][W+1];
  for(int i = 0; i <= W; i++)
      V[0][i] = 0;
 }
 public void knapsack()
 {
  int x[][] = new int[n+1][W+1];
  for(int i = 1; i <= n; i++)
  {
   for(int j = 0; j <= W; j++)
   {
    if((w[i-1] <= j) && (v[i-1]+V[i-1][j-w[i-1]] > V[i-1][j]))
    {
     V[i][j] = v[i-1] + V[i-1][j-w[i-1]];
     x[i][j] = 1;
    }
    else
    {
     V[i][j] = V[i-1][j];
     x[i][j] = 0;
    }
   }
  }
  System.out.println("Items Chosen");
  System.out.println("  Item  Weight  Profit ");
  int K = W;
  for(int i = n; i >= 1; i--)
  {
   if(x[i][K] == 1)
   {
    System.out.println("    "+i+"      "+w[i-1]+"       "+v[i-1]);
    K -= w[i-1];
   }
  }
  System.out.println("Maximum profit : "+V[n][W]);
 }
 public static void main(String[] args)
 {
  Knapsack obj = new Knapsack();
  obj.initialize();
  obj.knapsack();
 }
}
```

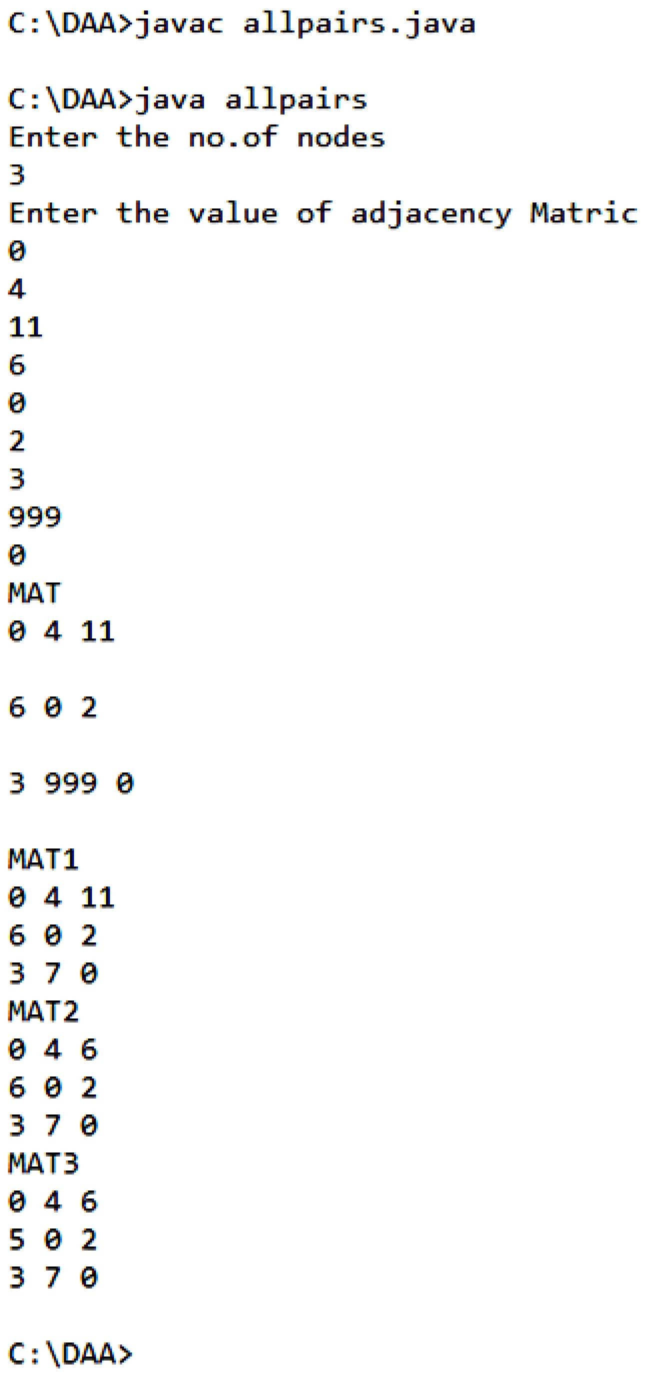
OUTPUT



```
C:\DAA>javac Knapsack.java

C:\DAA>java Knapsack
Enter number of items : 4
Enter capacity of knapsack : 8
Enter weight of items :
2
3
4
5
Enter profit of items :
1
2
5
6
Items Chosen
  Item  Weight  Profit
   4     5       6
   2     3       2
Maximum profit : 8

C:\DAA>
```

```
//All pair shortest path

import java.util.*;
import java.lang.*;
class allpairs
{
public static void main(String[] args)
{
int n,i,j,k;
Scanner s=new Scanner(System.in);
System.out.println("Enter the no.of nodes");
n=s.nextInt();
int mat[][]=new int[n+1][n+1];
System.out.print("Enter the value of adjacency Matric");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
mat[i][j]=s.nextInt();
}
}
System.out.println("MAT");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
System.out.print(mat[i][j]+" ");
}
System.out.println("\n");
}
for(k=1;k<=n;k++)
{
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
mat[i][j]=Math.min(mat[i][j],mat[i][k]+mat[k][j]);
}
}
System.out.println("MAT"+k);
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
System.out.print(mat[i][j]+" ");
}
System.out.print("\n");
}
}
}
}
```

OUTPUT

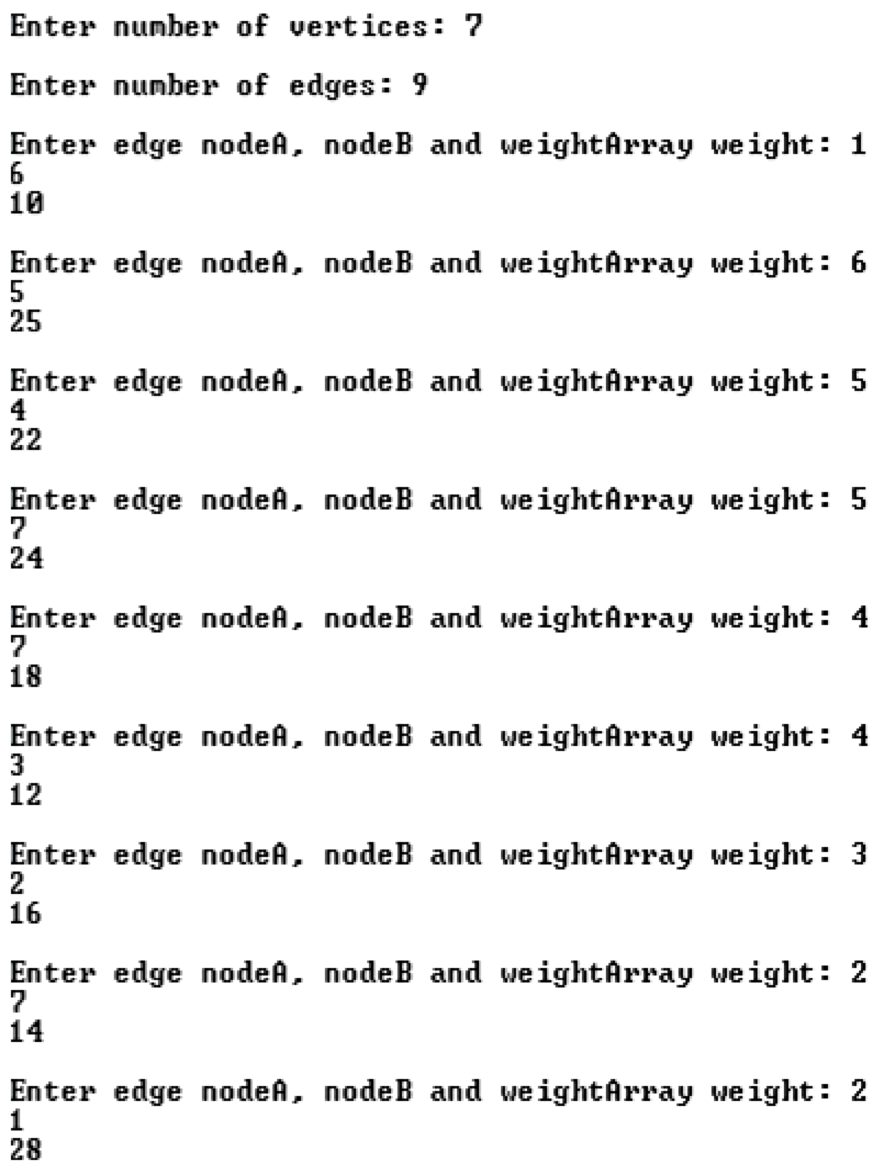


```
C:\DAA>javac allpairs.java

C:\DAA>java allpairs
Enter the no.of nodes
3
Enter the value of adjacency Matric
0
4
11
6
0
2
3
999
0
MAT
0 4 11

6 0 2

3 999 0

MAT1
0 4 11
6 0 2
3 7 0
MAT2
0 4 6
6 0 2
3 7 0
MAT3
0 4 6
5 0 2
3 7 0

C:\DAA>
```

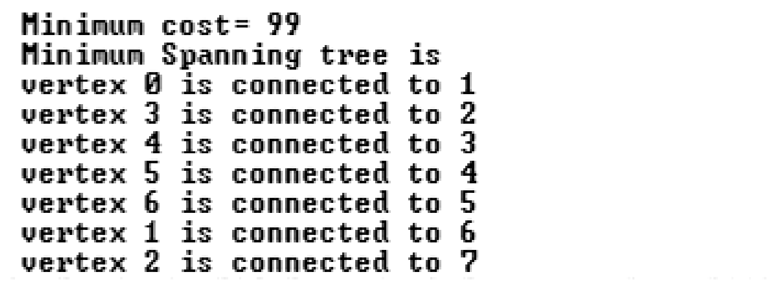
```
/* Prim's Algorithm */

import java.util.*;
import java.io.*;
public class Prim {
   static int w[][] = new int[20][20];
   static int v[] = new int [20];
   static int d[] = new int[20];
   static int p[] = new int[20];
   static int verticeCount, edgeCount;
   static int nodeA, nodeB, weight;
   static int current, total, mincost;

public static void main(String args[]) throws IOException
{
      BufferedReader buf = new BufferedReader(new InputStreamReader(System.in));
      System.out.print("\nEnter number of vertices: ");
      verticeCount = Integer.parseInt(buf.readLine());
      System.out.print("\nEnter number of edges: ");
      edgeCount = Integer.parseInt(buf.readLine());
      for (int i = 1; i <= verticeCount; i++)
      {
         for(int j = 1; j <= verticeCount; j++)
         {
            w[i][j] = 0;
         }
      }
     for (int i = 1; i <= verticeCount; i++)
     {
        p[i] = v[i] = 0;
        d[i] = 32767;
      }
      for (int i = 1; i <= edgeCount; i++)
      {
         System.out.print("\nEnter edge nodeA, nodeB and weightArray weight: ");
         nodeA=Integer.parseInt(buf.readLine());
         nodeB=Integer.parseInt(buf.readLine());
         weight=Integer.parseInt(buf.readLine());
         w[nodeA][nodeB] = w[nodeB][nodeA] = weight;
      }
         current = 1;
      d[current] = 0;
      total = 1;
      v[current] = 1;
      while( total != verticeCount)
```

```
        {
           for (int i = 1; i <= verticeCount; i++)
           {
              if ( w[current][i] != 0)
              {
                 if( v[i] == 0)
                 {
                    if (d[i] > w[current][i])
                    {
                       d[i] = w[current][i];
                       p[i] = current;
                    }
                 }
             } }
           mincost=32767;
           for (int i = 1 ; i <= verticeCount; i++)
           {
              if (v[i] == 0)
              {
                 if (d[i] < mincost)
                 {
                    mincost = d[i];
                    current = i;
                 }
              } }
           v[current]=1;
           total++;
        }
        mincost=0;
        for(int i=1;i<=verticeCount;i++)
        mincost=mincost+d[i];
        System.out.print("\n Minimum cost= "+mincost);
        System.out.print("\n Minimum Spanning tree is");
        for(int i=1;i<=verticeCount;i++)
        System.out.print("\n vertex "+p[i]+" is connected to "+i);
     }
  }
```

OUTPUT



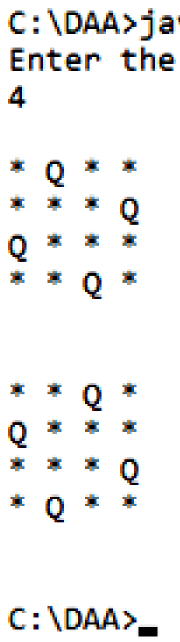
```
Enter number of vertices: 7

Enter number of edges: 9

Enter edge nodeA, nodeB and weightArray weight: 1
6
10

Enter edge nodeA, nodeB and weightArray weight: 6
5
25

Enter edge nodeA, nodeB and weightArray weight: 5
4
22

Enter edge nodeA, nodeB and weightArray weight: 5
7
24

Enter edge nodeA, nodeB and weightArray weight: 4
7
18

Enter edge nodeA, nodeB and weightArray weight: 4
3
12

Enter edge nodeA, nodeB and weightArray weight: 3
2
16

Enter edge nodeA, nodeB and weightArray weight: 2
7
14

Enter edge nodeA, nodeB and weightArray weight: 2
1
28
```



```
 Minimum cost= 99
 Minimum Spanning tree is
 vertex 0 is connected to 1
 vertex 3 is connected to 2
 vertex 4 is connected to 3
 vertex 5 is connected to 4
 vertex 6 is connected to 5
 vertex 1 is connected to 6
 vertex 2 is connected to 7
```

```
//NQueens

import java.util.*;
import java.lang.*;
public class queens1
{
public static int n, q[];
public static boolean Place(int k, int i)
{
   for (int j = 1; j <= k-1; j++)
   {
       if ((q[j] == i) || (Math.abs(q[j]-i) == Math.abs(j-k)))
             return false;
   }
   return true;
}

public static void printQueens(int[] q)
{
   System.out.println();
   for (int i = 1; i <= n; i++)
   {
       for (int j = 1; j <= n; j++)
       {
           if (q[i] == j) System.out.print("Q ");
           else System.out.print("* ");
       }
       System.out.println();
   }
   System.out.println();
}

public static void NQueens(int k, int n)
{
   for (int i = 1; i <= n; i++)
     {
       if (Place(k,i))
       {
         q[k] = i;
         if (k == n)
                     printQueens(q);
         else NQueens(k+1,n);
       }
     }
}
public static void main(String[] args)
{
   queens1 qu = new queens1();
   Scanner in = new Scanner(System.in);
   System.out.println("Enter the number of queens: ");
   n = in.nextInt();
   q = new int[n+1];
   qu.NQueens(1,n);
}
}
```

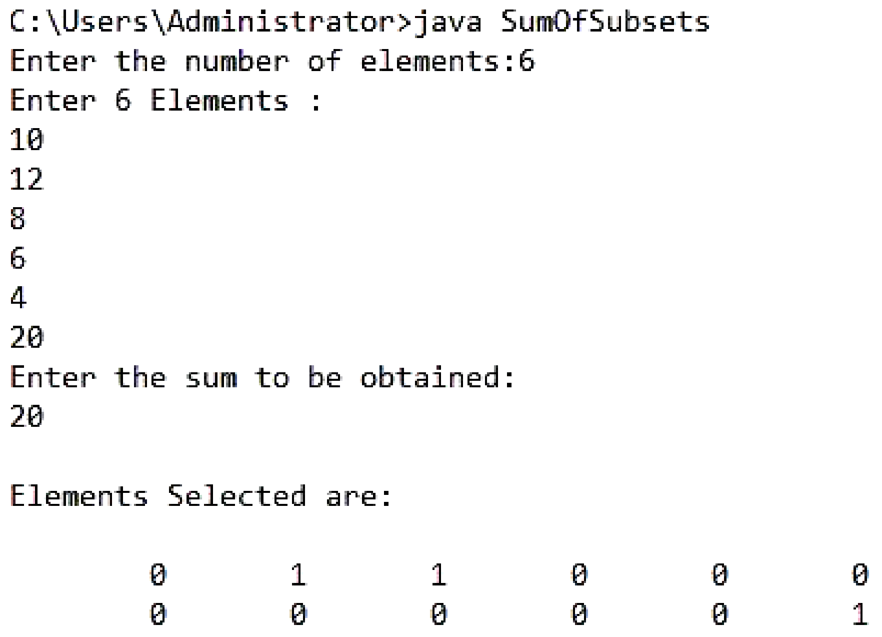
OUTPUT



```
C:\DAA>java queens1.java
Enter the number of queens:
4

* Q * *
* * * Q
Q * * *
* * Q *


* * Q *
Q * * *
* * * Q
* Q * *


C:\DAA>
```

```
/* Sum of Subsets */

import java.util.*;
public class SumOfSubsets
{
   int[] w;
   int[] x;
   int sum,n;

public void getData()
{
   Scanner sc = new Scanner(System.in);
   System.out.print("Enter the number of elements:");
   n = sc.nextInt();
   w = new int[n + 1];
   x = new int[n + 1];
   int total = 0;
   System.out.println("Enter " + n + " Elements :");
   for (int i = 1; i < n + 1; i++)
   {
       w[i] = sc.nextInt();
       total += w[i];
       x[i] = 0;
   }
   System.out.println("Enter the sum to be obtained: ");
   sum = sc.nextInt();
   if (total < sum)
   {
     System.out.println("Not possible to obtain the subset!!!");
     System.exit(1);
   }
   System.out.println("\nElements Selected are: ");
   subset(0, 1, total);
}

private void subset(int s, int k, int r)
{
   int i = 0;
   x[k] = 1;
   if (s + w[k] == sum)
   {
     System.out.println();
     for (i = 1; i <= n; i++)
     {
        System.out.print("\t" + x[i]);
     }
   }
   else if ((s + w[k] + w[k + 1]) <= sum)
   {
     subset(s + w[k], k + 1, r - w[k]);
   }
   if ((s + r - w[k]) >= sum && (s + w[k + 1]) <= sum)
   {
     x[k] = 0;
     subset(s, k + 1, r - w[k]);
   }
}
public static void main(String[] args)
{
   SumOfSubsets s = new SumOfSubsets();
   s.getData();
}
}
```

OUTPUT



```
C:\Users\Administrator>java SumOfSubsets
Enter the number of elements:6
Enter 6 Elements :
10
12
8
6
4
20
Enter the sum to be obtained:
20

Elements Selected are:

        0       1       1       0       0       0
        0       0       0       0       0       1
```