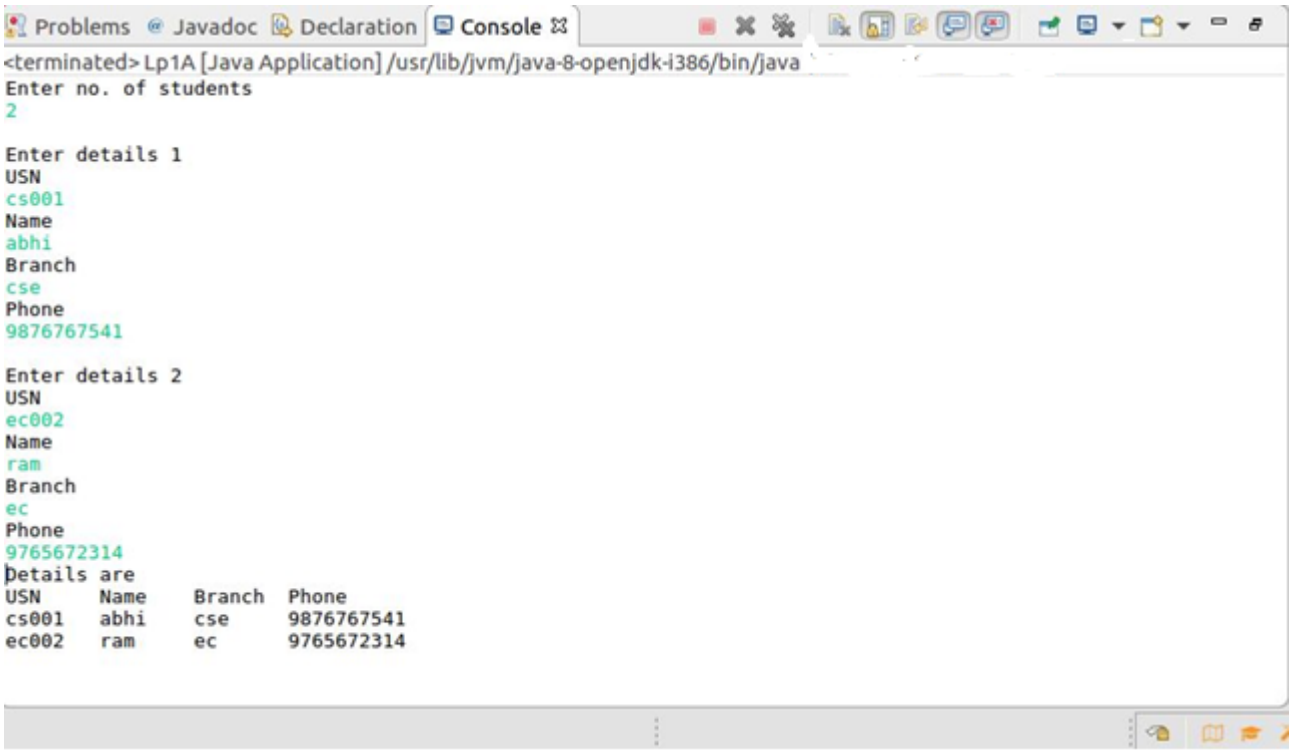
PROGRAM 1A

```
import java.io.*;
import java.util.*;
public class Student {
private String USN;
private String Name;
private String Branch;
private String Phone;
public String getUSN()
{
return USN;
}
public String getName()
{
	return Name;
}
public String getBranch()
{
	return Branch;
}
public String getPhone()
{
	return Phone;
}
public Student(String usn,String name,String branch,String phone)
{
	super();
	USN=usn;
	Name=name;
	Branch=branch;
	Phone=phone;
}
}
```

```
import java.io.*;
import java.util.*;
public class Lp1A {

	public static void main(String[] args) throws IOException {
		// TODO Auto-generated method stub
String usn,name,branch,phone;
Scanner in=new Scanner(System.in);
System.out.println("Enter no. of students");
int n=in.nextInt();
Student st[]=new Student[n];
for(int i=0;i<n;i++)
{
	System.out.println("\nEnter details "+(i+1));
	System.out.println("USN");
	usn=in.next();
```

```
        System.out.println("Name");
        name=in.next();
        System.out.println("Branch");
        branch=in.next();
        System.out.println("Phone");
        phone=in.next();
        st[i]=new Student(usn,name,branch,phone);
}
System.out.println("Details are");
System.out.println("USN\tName\tBranch\tPhone");
for(int i=0;i<n;i++)
{
        System.out.println(st[i].getUSN()+"\t"+st[i].getName()+"\t"+st[i].getBranch()+"\t"+st[i].get
Phone());
}

        }

}
```

OUTPUT:



PROGRAM 1B

```
import java.lang.*;
import java.util.*;
public class Lp1B {
Scanner in=new Scanner(System.in);
int top=-1;
int a[]=new int[10];
int SMAX=3;
public void push()
```

```
{
	int item;
	if(top==(SMAX-1))
	{
		System.out.println("Overflow");
	}
	else
	{
		System.out.println("Enter element to be inserted");
		item=in.nextInt();
		top++;
		a[top]=item;
	}
}
void pop()
{
	int item;
	if(top==-1)
	{
		System.out.println("Underflow");
	}
	else
	{
		item=a[top];
		top--;
		System.out.println("popped element is "+item);
	}
}
void display()
{
	int i;
	if(top==-1)
	{
		System.out.println("Empty");
	}
	else
	{
		System.out.println("Elements are");
		for(i=top;i>=0;i--)
		{
			System.out.print(a[i]+"\t");
		}
	}
	System.out.println();
}
	public static void main(String[] args) {
		// TODO Auto-generated method stub
Lp1B s1=new Lp1B();
int ch=0;
Scanner in=new Scanner(System.in);
for(;;)
{
```

```
        System.out.println("Stack");
        System.out.println("1.Push");
        System.out.println("2.Pop");
        System.out.println("3.Display");
        System.out.println("4.Exit");
        System.out.println("Enter choice");
        ch=in.nextInt();
        switch(ch)
        {
        case 1: s1.push();
        break;
        case 2: s1.pop();
        break;
        case 3: s1.display();
        break;
        case 4: System.exit(0);
        break;
        default: System.out.println("Invalid choice");
        break;
        }
}

        }

}
```

OUTPUT:

```
Problems  Javadoc  Declaration  Console
Lp1B [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
Stack
1.Push
2.Pop
3.Display
4.Exit
Enter choice
1
Enter element to be inserted
3
Stack
1.Push
2.Pop
3.Display
4.Exit
Enter choice
1
Enter element to be inserted
2
Stack
1.Push
2.Pop
3.Display
4.Exit
Enter choice
1
Enter element to be inserted
1
```

```
Stack
1.Push
2.Pop
3.Display
4.Exit
Enter choice
3
Elements are
1       2       3
Stack
1.Push
2.Pop
3.Display
4.Exit
Enter choice
2
popped element is 1
Stack
1.Push
2.Pop
3.Display
4.Exit
Enter choice
2
popped element is 2
```

```
Stack
1.Push
2.Pop
3.Display
4.Exit
Enter choice
2
popped element is 3
Stack
1.Push
2.Pop
3.Display
4.Exit
Enter choice
2
Underflow
Stack
1.Push
2.Pop
3.Display
4.Exit
Enter choice
1
Enter element to be inserted
1
```

```
Stack
1.Push
2.Pop
3.Display
4.Exit
Enter choice
1
Enter element to be inserted
2
Stack
1.Push
2.Pop
3.Display
4.Exit
Enter choice
1
Enter element to be inserted
3
Stack
1.Push
2.Pop
3.Display
4.Exit
Enter choice
1
4
Overflow
```

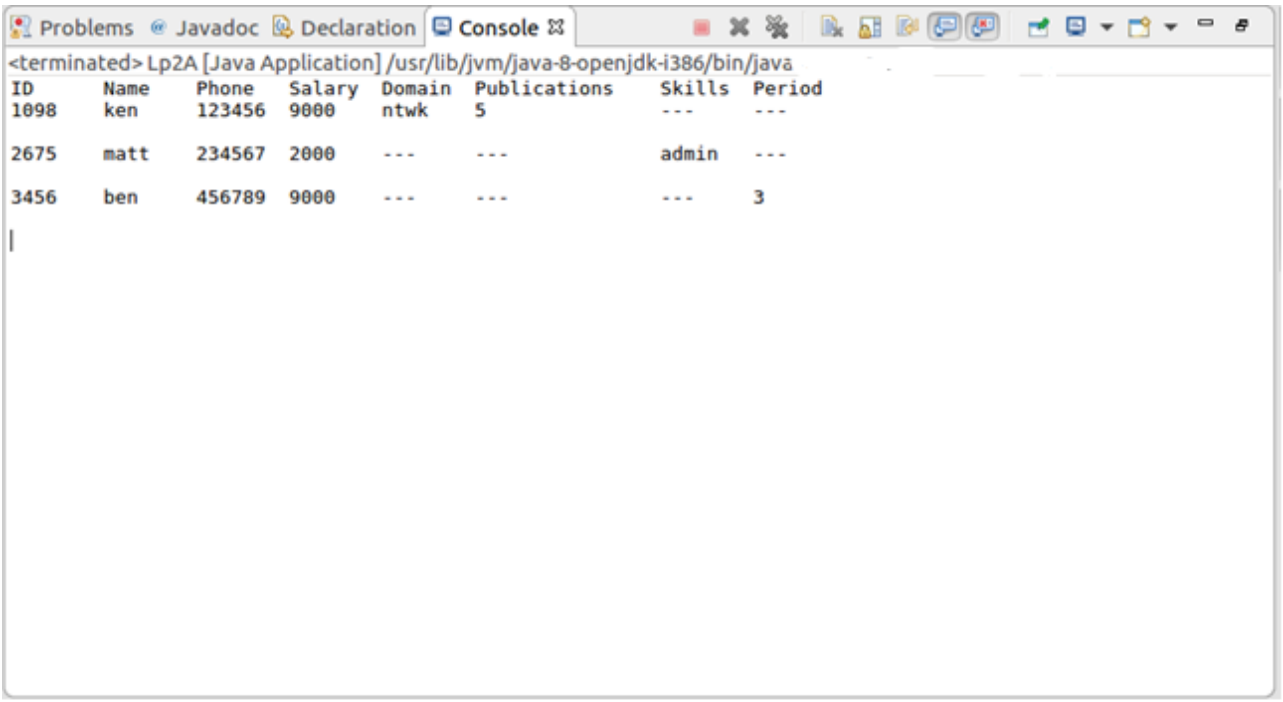
PROGRAM 2A

```
import java.lang.*;
import java.util.*;
public class Staff {
private int ID;
private String Name;
private String Phone;
private long Salary;
public Staff(int id,String name,String phone,long salary)
{
        ID=id;
        Name=name;
        Phone=phone;
        Salary=salary;
}
public void display()
{
        System.out.print(ID+"\t"+Name+"\t"+Phone+"\t"+Salary);
}
```

```
}
class Teaching extends Staff
{
	private String Domain;
	private int Publications;
	public Teaching(int id,String name,String phone,long salary,String domain,int publications)
	{
		super(id,name,phone,salary);
		Domain=domain;
		Publications=publications;
	}
	public void display()
	{
		super.display();
		System.out.println("\t"+Domain+"\t"+Publications+"\t"+"\t"+"---"+"\t"+"---");
	}
}
class Technical extends Staff
{
	private String Skills;
	public Technical(int id,String name,String phone,long salary,String skills)
	{
		super(id,name,phone,salary);
		Skills=skills;
	}
	public void display()
	{
		super.display();
		System.out.println("\t"+"---"+"\t"+"---"+"\t"+"\t"+Skills+"\t"+"---");
	}
}
class Contract extends Staff
{
	private int Period;
	public Contract(int id,String name,String phone,long salary,int period)
	{
		super(id,name,phone,salary);
		Period=period;
	}
	public void display()
	{
		super.display();
		System.out.println("\t"+"---"+"\t"+"---"+"\t"+"\t"+"---"+"\t"+Period);
	}
}
```

```
package maverick;
import java.lang.*;
import java.util.*;
public class Lp2A {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
Staff staff[]=new Staff[3];
staff[0]=new Teaching(1098,"ken","123456",9000,"ntwk",5);
staff[1]=new Technical(2675,"matt","234567",2000,"admin");
staff[2]=new Contract(3456,"ben","456789",9000,3);
System.out.println("ID\tName\tPhone\tSalary\tDomain\tPublications\tSkills\tPeriod");
for(int i=0;i<3;i++)
{
	staff[i].display();
	System.out.println();
}
	}

}
```

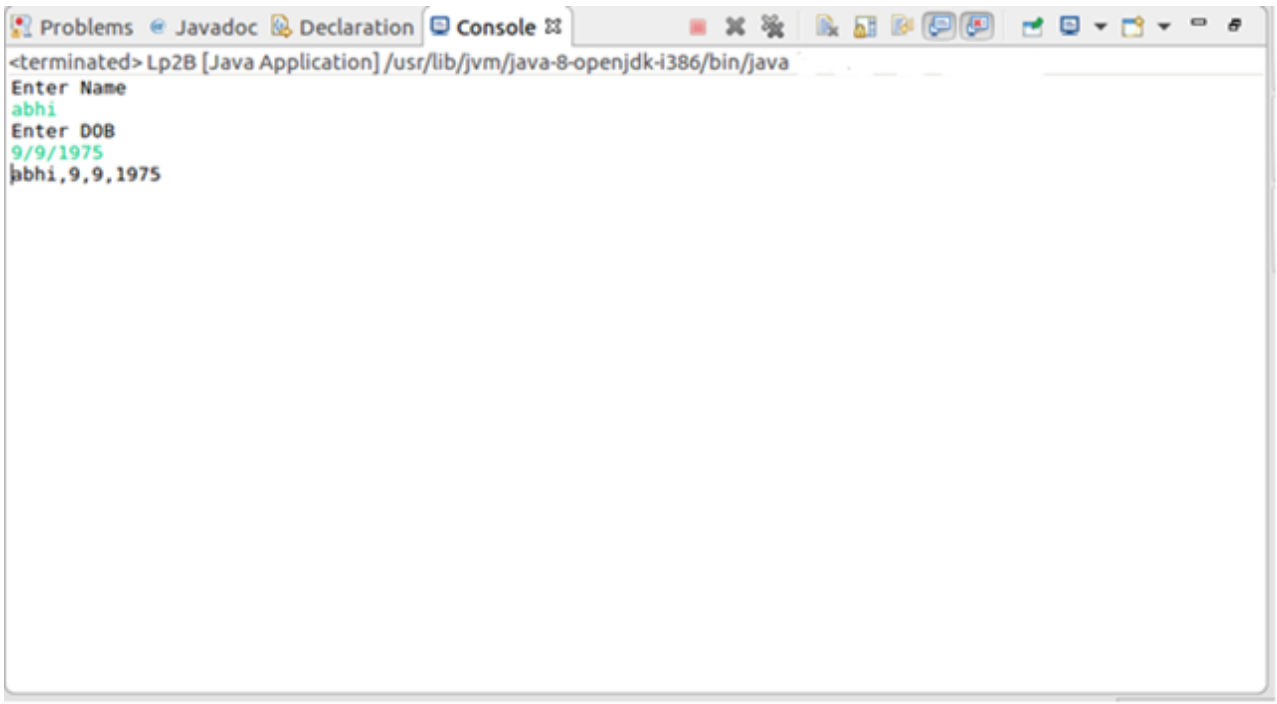
OUTPUT:



PROGRAM 2B

```
import java.util.*;
class Customer {
private String Name;
private String DOB;
public void readData(String name,String dob)
{
	this.Name=name;
	this.DOB=dob;
}
public void displayData()
{
	StringTokenizer st=new StringTokenizer(this.DOB,"/");
	System.out.print(this.Name);
	while(st.hasMoreTokens())
	{
		System.out.print(“,”+st.nextToken());
	}

}
}

public class Lp2B {

	public static void main(String[] args) {
Scanner in=new Scanner(System.in);
System.out.println("Enter Name");
String name=in.nextLine();
System.out.println("Enter DOB");
String date=in.next();
Customer customer=new Customer();
customer.readData(name, date);
customer.displayData();
	}

}
```

OUTPUT:



PROGRAM 3A

```
import java.util.*;
public class Lp3A {

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

int a,b;
float Q;
Scanner in=new Scanner(System.in);
```

```
System.out.println("Enter a");
a=in.nextInt();
System.out.println("Enter b");
b=in.nextInt();
try
{
	if(b!=0)
	{
		Q=(float)a/b;
		System.out.println("result="+Q);
	}
	else
		throw new ArithmeticException();
}
catch(ArithmeticException e)
{
	System.out.println("Divide by zero error");
	System.out.println(e);
}
	}
}
```

OUTPUT:

```
Problems  Javadoc  Declaration  Console
<terminated> Lp3A [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
Enter a
6
Enter b
2
result=3.0
```

```
Problems  Javadoc  Declaration  Console
<terminated> Lp3A [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
Enter a
5
Enter b
0
Divide by zero error
java.lang.ArithmeticException
```

PROGRAM 3B

```
import java.util.*;
public class Square implements Runnable {
public int x;
public Square(int x)
{
        this.x=x;
}
public void run()
{
        System.out.println("2nd Thread-Square of "+x+" is "+(x*x));;
```

```
}
}


import java.util.*;
public class Cube implements Runnable {
public int x;
public Cube(int x)
{
	this.x=x;
}
public void run()
{
	System.out.println("3rd Thread-Cube of "+x+" is "+(x*x*x));;
}
}


import java.util.*;
public class ThreadRandom extends Thread {
public void run()
{
	int n=0;
	Random r=new Random();
	try
	{
		for(int i=0;i<2;i++)
		{
			n=r.nextInt(100);
			System.out.println("Main Thread Started and Generated "+n);
			Thread t2=new Thread(new Square(n));
			t2.start();
			Thread t3=new Thread(new Cube(n));
			t3.start();
			Thread.sleep(1000);
			System.out.println("----------");
		}
	}
	catch(Exception e)
	{
		System.out.println(e.getMessage());
	}
}
}


import java.util.*;
public class MultiThread {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
ThreadRandom tr=new ThreadRandom();
```

```
Thread t1=new Thread(tr);
t1.start();
        }

}
```

OUTPUT:

```
Problems  Javadoc  Declaration  Console
<terminated> Lp3B [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
Main Thread Started and Generated 81
2nd Thread-Square of 81 is 6561
3rd Thread-Cube of 81 is 531441
----------
Main Thread Started and Generated 49
2nd Thread-Square of 49 is 2401
3rd Thread-Cube of 49 is 117649
----------
```

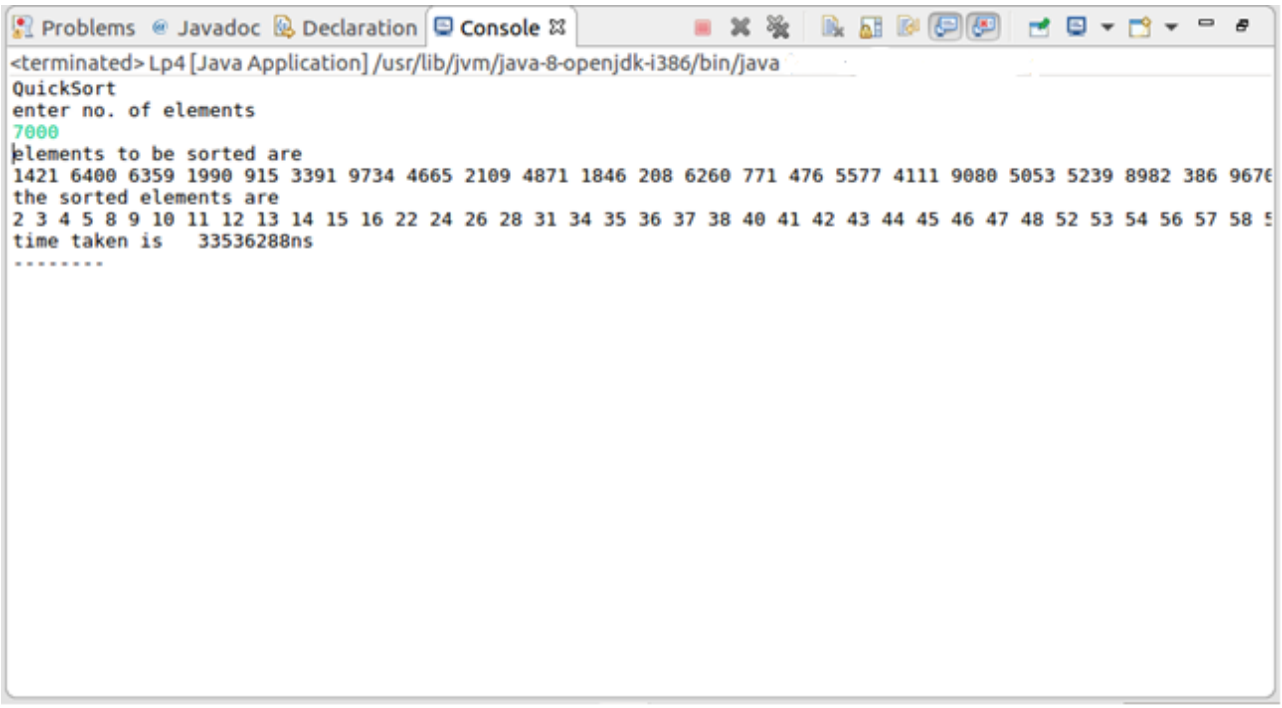
PROGRAM 4

```
import java.util.*;
public class Lp4 {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
Lp4 Lab=new Lp4();
int a[]=new int[100000];
Scanner in=new Scanner(System.in);
long start,end;
System.out.println("QuickSort");
System.out.println("enter no. of elements");
int n=in.nextInt();
for(int i=0;i<n;i++)
{
        a[i]=Lab.generateRandom(a,10000);

}
System.out.println("elements to be sorted are");
for(int i=0;i<n;i++)
{
        System.out.print(a[i]+" ");
}
System.out.println();
start=System.nanoTime();
quicksort(a,0,n-1);
```

```
end=System.nanoTime();
System.out.println("the sorted elements are");
for(int i=0;i<n;i++)
{
	System.out.print(a[i]+" ");
}
System.out.println();
System.out.println("time taken is"+"\t"+(end-start)+"ns");
System.out.println("--------");
	}
static void quicksort(int a[],int p,int q)
{
	int j;
	if(p<q)
	{
		j=partition(a,p,q);
		quicksort(a,p,j-1);
		quicksort(a,j+1,q);

	}
}
static int partition(int a[],int m,int p)
{
	int v,i,j;
	v=a[m];
	i=m;
	j=p;
	while(i<j)
	{
		while(a[i]<=v)
			i++;
		while(a[j]>v)
			j--;
		if(i<j)
			interchange(a,i,j);

	}
	a[m]=a[j];
	a[j]=v;
	return j;
}
static void interchange(int a[],int i,int j)
{
	int p;
	p=a[i];
	a[i]=a[j];
	a[j]=p;
}
public int generateRandom(int a[],int bound)
{
	Random r=new Random();
	int offset=r.nextInt(bound);
```

```
        while(alreadyThere(a,offset))
                offset=r.nextInt(bound);
        return offset;
}
private boolean alreadyThere(int arr[],int e)
{
        for(int i=0;i<arr.length;i++)
        {
                if(e==arr[i])
                        return true;
        }
        return false;
}
}
```

OUTPUT:



PROGRAM 5

```
package maverick;
import java.util.*;
public class Lp5 {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
Lp5 Lab=new Lp5();
int a[]=new int[100000];
Scanner in=new Scanner(System.in);
long start,end;
System.out.println("MergeSort");
System.out.println("enter no. of elements");
int n=in.nextInt();
for(int i=0;i<n;i++)
{
```

```
        a[i]=Lab.generateRandom(a,10000);

}
System.out.println("elements to be sorted are");
for(int i=0;i<n;i++)
{
        System.out.print(a[i]+" ");
}
System.out.println();
start=System.nanoTime();
mergesort(a,0,n-1);
end=System.nanoTime();
System.out.println("the sorted elements are");
for(int i=0;i<n;i++)
{
        System.out.print(a[i]+" ");
}
System.out.println();
System.out.println("time taken is"+"\t"+(end-start)+"ns");
System.out.println("--------");
        }
static void mergesort(int a[],int l,int h)
{
        int m;
        if(l<h)
        {
                m=(l+h)/2;
                mergesort(a,l,m);
                mergesort(a,m+1,h);
                merge(a,l,m,h);
        }
}
static void merge(int a[],int low,int mid,int high)
{
        int j,i,h,k;
        int b[]=new int[100000];
        h=low;
        i=low;
        j=mid+1;
        while((h<=mid)&&(j<=high))
        {
                if(a[h]<a[j])
                {
                        b[i]=a[h];
                        h=h+1;
                }
                else
                {
                        b[i]=a[j];
                        j=j+1;
                }
                i=i+1;
```

```
        }
        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 int generateRandom(int a[],int bound)
{
        Random r=new Random();
        int offset=r.nextInt(bound);
        while(alreadyThere(a,offset))
                offset=r.nextInt(bound);
        return offset;
}
private boolean alreadyThere(int arr[],int e)
{
        for(int i=0;i<arr.length;i++)
        {
                if(e==arr[i])
                        return true;
        }
        return false;
}
}
```

OUTPUT:

```
Problems  Javadoc  Declaration  Console
<terminated> Lp5 [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
MergeSort
enter no. of elements
8000
elements to be sorted are
1305 9525 9081 7334 7528 93 4533 7301 4649 7425 8356 1818 2344 8430 5622 2553 4929 3249 5460 6754 4977 5097 7
the sorted elements are
1 2 3 6 7 8 9 10 11 13 14 16 19 21 23 24 25 26 27 29 31 32 33 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 56
time taken is   3352436356ns
--------
```

PROGRAM 6A

```
import java.util.*;
public class Lp6A {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
int v[][]=new int[10][10];
int w[]=new int[10];
int p[]=new int[10];
Scanner in=new Scanner(System.in);
int i,j;
System.out.println("0/1 Knapsack by Dynamic Programming");
System.out.println("enter total no. of items");
int n=in.nextInt();
System.out.println("enter weight of each item");
for(i=1;i<=n;i++)
{
	w[i]=in.nextInt();
}
System.out.println("enter profit of each item");
for(i=1;i<=n;i++)
{
	p[i]=in.nextInt();
}
System.out.println("enter Knapsack capacity");
int m=in.nextInt();
DisplayInfo(m,n,w,p);
Knapsack(m,n,w,p,v);
System.out.println("contents of Knapsack table are");
for(i=1;i<=n;i++)
{
```

```
		for(j=1;j<=m;j++)
		{
			System.out.print(v[i][j]+" ");
		}
		System.out.println();
	}
	Optimal(m,n,w,v);
		}
	static void DisplayInfo(int m,int n,int w[],int p[])
	{
		System.out.println("ITEM\tWEIGHT\tPROFIT");
		for(int i=1;i<=n;i++)
		{
			System.out.println(i+"\t"+w[i]+"\t"+p[i]);
		}
		System.out.println("capacity="+"\t"+m);
	}
	static void Knapsack(int m,int n,int w[],int p[],int v[][])
	{
		for(int i=0;i<=n;i++)
		{
			for(int j=0;j<=m;j++)
	{
		if(i==0||j==0)
			v[i][j]=0;
		else if(j<w[i])
			v[i][j]=v[i-1][j];
		else
			v[i][j]=max(v[i-1][j],v[i-1][j-w[i]]+p[i]);
	}
		}
	}
	private static int max(int i,int j)
	{
		if(i>j)
			return i;
		else
			return j;
	}
	static void Optimal(int m,int n,int w[],int v[][])
	{
		int i=n,j=m,item=0;
		int x[]=new int[10];
		while(i!=0&&j!=0)
		{
			if(v[i][j]!=v[i-1][j])
			{
				x[i]=1;
				j=j-w[i];
			}
			i=i-1;
		}
```

```
        System.out.println("Optimal solution is"+"\t"+v[n][m]);
        System.out.println("selected items are");
        for(i=1;i<=n;i++)
        {
                if(x[i]==1)
{
        System.out.println(i+" ");
        item=i;
}
if(item==0)
        System.out.println("Knapsack is full");
        }
}
}
```

OUTPUT:

```
Problems  Javadoc  Declaration  Console
<terminated> Lp6A [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
enter total no. of items
4
enter weight of each item
2
1
3
2
enter profit of each item
12
10
20
15
enter Knapsack capacity
5
ITEM    WEIGHT  PROFIT
1       2       12
2       1       10
3       3       20
4       2       15
capacity=       5
contents of Knapsack table are
0 12 12 12 12
10 12 22 22 22
10 12 22 30 32
10 15 25 30 37
Optimal solution is     37
selected items are
1
2
4
```

PROGRAM 6B

```
import java.util.*;
public class Lp6B {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
float w[]=new float[10],p[]=new float[10];
float ratio[]=new float[10];
Scanner in=new Scanner(System.in);
int i;
System.out.println("0/1 Knapsack problem by Greedy Method");
System.out.println("enter total no. of items");
int n=in.nextInt();System.out.println("enter weight of each item");
for(i=1;i<=n;i++)
{
        w[i]=in.nextFloat();
```

```
}
System.out.println("enter profit of each item");
for(i=1;i<=n;i++)
{
	p[i]=in.nextFloat();
}
System.out.println("enter Knapsack capacity");
int m=in.nextInt();
for(i=1;i<=n;i++)
{
	ratio[i]=p[i]/w[i];
}
System.out.println("Information about the problem is");
DisplayInfo(n,w,p,ratio);
System.out.println("capacity is"+"\t"+m);
SortArray(n,ratio,w,p);
System.out.println("Details after sorting items on p/w ratio in descending order");
DisplayInfo(n,w,p,ratio);
GreKnapsack(m,n,w,p);
}
	static void SortArray(int n,float ratio[],float w[],float p[])
	{
		int i,j;
		for(i=1;i<=n;i++)
		{
			for(j=1;j<=n-i;j++)
			{
				if(ratio[j]<ratio[j+1])
				{
					float temp=ratio[j];
					ratio[j]=ratio[j+1];
					ratio[j+1]=temp;
					temp=w[j];
					w[j]=w[j+1];
					w[j+1]=temp;
					temp=p[j];
					p[j]=p[j+1];
					p[j+1]=temp;


				}
			}
		}
	}
	static void DisplayInfo(int n,float w[],float p[],float ratio[])
	{
		System.out.println("ITEM\tWEIGHT\tPROFIT\tRATIO");
		for(int i=1;i<=n;i++)
		{
			System.out.println(i+"\t"+w[i]+"\t"+p[i]+"\t"+ratio[i]);
		}
	}
```

```
        static void GreKnapsack(int u,int n,float w[],float p[])
        {
                float x[]=new float[10],tp=0;
                int i;
                for(i=1;i<=n;i++)
                        x[i]=0;
                for(i=1;i<=n;i++)
                {
                        if(w[i]>u)
                                break;
                        else
                        {
                                x[i]=1;
                                tp=tp+p[i];
                                u=(int)(u-w[i]);
                        }
                }
                if(i<n)
                        x[i]=u/w[i];
                tp=tp+(x[i]*p[i]);
                System.out.println("Result is");
                for(i=1;i<=n;i++)
                        System.out.println("\t"+x[i]);
                System.out.println("max profit="+"\t"+tp);
        }
}
```

OUTPUT:

```
Problems  Javadoc  Declaration  Console
<terminated> Lp6B [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
0/1 Knapsack problem by Greedy Method
enter total no. of items
4
enter weight of each item
2
1
3
2
enter profit of each item
12
10
20
15
enter Knapsack capacity
5
Information about the problem is
ITEM    WEIGHT  PROFIT  RATIO
1       2.0     12.0    6.0
2       1.0     10.0    10.0
3       3.0     20.0    6.6666665
4       2.0     15.0    7.5
capacity is     5
```

```
Information about the problem is
ITEM    WEIGHT  PROFIT  RATIO
1       2.0     12.0    6.0
2       1.0     10.0    10.0
3       3.0     20.0    6.6666665
4       2.0     15.0    7.5
capacity is     5
Details after sorting items on p/w ratio in descending order
ITEM    WEIGHT  PROFIT  RATIO
1       1.0     10.0    10.0
2       2.0     15.0    7.5
3       3.0     20.0    6.6666665
4       2.0     12.0    6.0
Result is
        1.0
        1.0
        0.6666667
        0.0
max profit=     38.333336
```

PROGRAM 7

```
import java.util.*;
public class Lp7 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
int i,j;
int dist[]=new int[10],visited[]=new int[10];
int cost[][]=new int[10][10],path[]=new int[10];
Scanner in=new Scanner(System.in);
System.out.println("DIJKSTRA");
System.out.println("enter no. of nodes");
int n=in.nextInt();
System.out.println("enter cost matrix");
for(i=1;i<=n;i++)
{
	for(j=1;j<=n;j++)
		{
			cost[i][j]=in.nextInt();
		}
}
System.out.println("the entered cost matrix is");
for(i=1;i<=n;i++)
{
		for(j=1;j<=n;j++)
		{
			System.out.print(cost[i][j]+"\t");
		}
		System.out.println();
}
System.out.println("enter source ");
int sv=in.nextInt();
Dij(cost,dist,sv,n,path,visited);
PrintPath(sv,n,dist,path,visited);
System.out.println("-----------");
}
static void Dij(int cost[][],int dist[],int sv,int n,int path[],int visited[])
{
```

```
        int count=2,min,v=0;
        for(int i=1;i<=n;i++)
        {
                visited[i]=0;
                dist[i]=cost[sv][i];
                if(cost[sv][i]==999)
                        path[i]=0;
                else
                        path[i]=sv;

        }
        visited[sv]=1;
        while(count<=n)
        {
                min=999;
                for(int w=1;w<=n;w++)

                        if((dist[w]<min)&&(visited[w]==0))
                        {
                                min=dist[w];
                                v=w;
                        }
                visited[v]=1;
                count++;
                for(int w=1;w<=n;w++)
                {
                        if((dist[w]>dist[v]+cost[v][w]))
                        {
                                dist[w]=dist[v]+cost[v][w];
                                path[w]=v;
                        }
                }
        }
}
static void PrintPath(int sv,int n,int dist[],int path[],int visited[])
{
        for(int w=1;w<=n;w++)
        {
                if(visited[w]==1&&w!=sv)
                {
                        System.out.print("shortest distance between ");
                        System.out.println(sv+"->"+w+" is "+dist[w]);
                        int t=path[w];
                        System.out.print("the path is");
                        System.out.print(" "+w);
                        while(t!=sv)
                        {
                                System.out.print("<->"+t);
                                t=path[t];
                        }
                        System.out.println("<->"+sv);
                        }
```

```
                }
}
}
```

OUTPUT:

```
Problems  Javadoc  Declaration  Console
<terminated> Lp7 [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
DIJKSTRA
enter no. of nodes
5
enter cost matrix
0 3 999 7 999
3 0 4 2 999
999 4 0 5 6
7 2 5 0 4
999 999 6 4 0
the entered cost matrix is
0       3       999     7       999
3       0       4       2       999
999     4       0       5       6
7       2       5       0       4
999     999     6       4       0
enter source
1
shortest distance between 1->2 is 3
the path is 2<->1
shortest distance between 1->3 is 7
the path is 3<->2<->1
shortest distance between 1->4 is 5
the path is 4<->2<->1
shortest distance between 1->5 is 9
the path is 5<->4<->2<->1
-----------
```

PROGRAM 8

```
package maverick;
import java.util.*;
public class Lp8 {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
int cost[][]=new int[10][10];
int i,j,mincost=0;
Scanner in=new Scanner(System.in);
System.out.println("kruskal's");
System.out.println("enter no. of nodes");
int n=in.nextInt();
System.out.println("enter cost matrix");
for(i=1;i<=n;i++)
{
        for(j=1;j<=n;j++)
        {
                cost[i][j]=in.nextInt();
        }
}
System.out.println("The entered cost matrix is");
for(i=1;i<=n;i++)
{
        for(j=1;j<=n;j++)
        {
                System.out.print(cost[i][j]+"\t");
```

```
        }
        System.out.println();
}
System.out.println("The MST edges and costs are");
mincost=Kruskal(cost,n,mincost);
System.out.println("The MST cost is");
System.out.println(mincost);

        }
        static int Kruskal(int cost[][],int n,int mincost)
        {
                int ne=1,min;
                int a=0,b=0,u=0,v=0;
                int parent[]=new int[10];
                while(ne<n)
                {
                        min=999;
                        for(int i=1;i<=n;i++)
                        {
                                for(int j=1;j<=n;j++)
                                {
                                        if(cost[i][j]<min)
                                        {
                                                min=cost[i][j];
                                                a=u=i;
                                                b=v=j;
                                        }
                                }
                        }
                        u=find(u,parent);
                        v=find(v,parent);
                        if(union(u,v,parent)!=0)
                        {
                                System.out.print((ne++)+") min edge is ");
                                System.out.println("("+a+","+b+") and cost is "+min);
                                mincost+=min;
                                parent[v]=u;
                        }
                        cost[a][b]=cost[b][a]=999;
                }
        return mincost;

                }
static int find(int i,int parent[])
{
        while(parent[i]!=0)
                i=parent[i];
        return i;
}
static int union(int i,int j,int parent[])
{
        if(i!=j)
```

```
        {
                parent[j]=i;
                return 1;
        }
        else
                return 0;
        }

        }
```

OUTPUT:

```
Problems  Javadoc  Declaration  Console
<terminated> Lp8 [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
kruskal's
enter no. of nodes
5
enter cost matrix
0 3 999 7 999
3 0 4 2 999
999 4 0 5 6
7 2 5 0 4
999 999 6 4 0
The entered cost matrix is
0       3       999     7       999
3       0       4       2       999
999     4       0       5       6
7       2       5       0       4
999     999     6       4       0
The MST edges and costs are
1) min edge is (2,4) and cost is 2
2) min edge is (1,2) and cost is 3
3) min edge is (2,3) and cost is 4
4) min edge is (4,5) and cost is 4
The MST cost is
13
```

PROGRAM 9

```
import java.util.*;
public class Lp9 {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
int cost[][]=new int[10][10];
int i,j,mincost=0;
Scanner in=new Scanner(System.in);
System.out.println("PRIMS");
System.out.println("enter no. of nodes");
int n=in.nextInt();
System.out.println("enter cost matrix");
for(i=1;i<=n;i++)
{
        for(j=1;j<=n;j++)
        {
                cost[i][j]=in.nextInt();
        }
```

```
}
System.out.println("the entered cost matrix is");
for(i=1;i<=n;i++)
{
	for(j=1;j<=n;j++)
	{
		System.out.print(cost[i][j]+"\t");
	}
	System.out.println();
}
System.out.println("MST edges and cost are ");
mincost=Prims(cost,n,mincost);
System.out.println("MST cost is");
System.out.println(+mincost);
System.out.println("----------");
}
static int Prims(int cost[][],int n,int mincost)
{
	int nearV[]=new int[10],t[][]=new int[10][3],u=0,i,j,k;
	for(i=2;i<=n;i++)
		nearV[i]=1;
	nearV[1]=0;
	for(i=1;i<n;i++)
	{
		int min=999;
		for(j=1;j<=n;j++)
		{
			if(nearV[j]!=0&&cost[j][nearV[j]]<min)
			{
				min=cost[j][nearV[j]];
				u=j;

			}
		}
		t[i][1]=u;
		t[i][2]=nearV[u];
		mincost+=min;
		nearV[u]=0;
		for(k=1;k<=n;k++)
		{
			if(nearV[k]!=0&&cost[k][nearV[k]]>cost[k][u])
				nearV[k]=u;
		}
		System.out.print(i+")min edge is ("+t[i][1]);
		System.out.println(","+t[i][2]+") and cost is "+min);
	}
	return mincost;
}
}
```

OUTPUT:

```
Problems  Javadoc  Declaration  Console
<terminated> Lp9 [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
PRIMS
enter no. of nodes
5
enter cost matrix
0 3 999 7 999
3 0 4 2 999
999 4 0 5 6
7 2 5 0 4
999 999 6 4 0
the entered cost matrix is
0       3       999     7       999
3       0       4       2       999
999     4       0       5       6
7       2       5       0       4
999     999     6       4       0
MST edges and cost are
1)min edge is (2,1) and cost is 3
2)min edge is (4,2) and cost is 2
3)min edge is (3,2) and cost is 4
4)min edge is (5,4) and cost is 4
MST cost is
13
----------
```

PROGRAM 10A

```
import java.util.*;
public class Lp10A {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
int a[][]=new int[10][10];
int i,j;
Scanner in=new Scanner(System.in);
System.out.println("Floyd's");
System.out.println("enter no. of nodes");
int n=in.nextInt();
System.out.println("Enter the adjacency matrix");
for(i=1;i<=n;i++)
{
	for(j=1;j<=n;j++)
	{
		a[i][j]=in.nextInt();
	}
}
System.out.println("The entered adjacency matrix is");
for(i=1;i<=n;i++)
{
	for(j=1;j<=n;j++)
	{
		System.out.print(a[i][j]+"\t");

	}
	System.out.println();
}
Floyd(a,n);
```

```
System.out.println("All pair shortest path matrix");
for(i=1;i<=n;i++)
{
	for(j=1;j<=n;j++)
	{
		System.out.print(a[i][j]+"\t");

	}
	System.out.println();
}
	}
static void Floyd(int a[][],int n)
{
	int i,j,k;
	for(k=1;k<=n;k++)
	{
		for(i=1;i<=n;i++)
		{
			for(j=1;j<=n;j++)
			{
				a[i][j]=min(a[i][j],a[i][k]+a[k][j]);
			}
		}
	}
}
static int min(int a,int b)
{
	if(a>b)
		return b;
	else
		return a;
}
}
```

OUTPUT:

```
Problems  Javadoc  Declaration  Console
<terminated> Lp10A [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
Floyd's
enter no. of nodes
5
Enter the adjacency matrix
0 3 999 7 999
3 0 4 2 999
999 4 0 5 6
7 2 5 0 4
999 999 6 4 0
The entered adjacency matrix is
0       3       999     7       999
3       0       4       2       999
999     4       0       5       6
7       2       5       0       4
999     999     6       4       0
All pair shortest path matrix
0       3       7       5       9
3       0       4       2       6
7       4       0       5       6
5       2       5       0       4
9       6       6       4       0
```

PROGRAM 10B

```
import java.util.*;
public class Lp10B {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
int c[][]=new int[10][10],tour[]=new int[10];
int i,j,cost;
Scanner in=new Scanner(System.in);
System.out.println("TSP by Dynamic Programming");
System.out.println("Enter no. of cities");
int n=in.nextInt();
if(n==1)
{
	System.out.println("Path not possible");
	System.exit(0);
}
System.out.println("Enter the cost matrix");
for(i=1;i<=n;i++)
{
	for(j=1;j<=n;j++)
	{
		c[i][j]=in.nextInt();
	}
}
System.out.println("The entered cost matrix is");
for(i=1;i<=n;i++)
{
	for(j=1;j<=n;j++)
	{
		System.out.print(c[i][j]+"\t");
```

```
        }
        System.out.println();
}
for(i=1;i<=n;i++)
        tour[i]=i;
cost=tsp(c,tour,1,n);
System.out.println("The accurate path is");
for(i=1;i<=n;i++)
        System.out.print(tour[i]+"->");
System.out.println("1");
System.out.println("The accurate cost is "+cost);

        }
static int tsp(int c[][],int tour[],int start,int n)
{
        int mintour[]=new int[10],temp[]=new int[10],mincost=999,ccost,i,j,k;
        if(start==n-1)
        {
                return(c[tour[n-1]][tour[n]]+c[tour[n]][1]);
        }
        for(i=start+1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                        temp[j]=tour[j];
        temp[start+1]=tour[i];
        temp[i]=tour[start+1];
        if((c[tour[start]][tour[i]]+(ccost=tsp(c,temp,start+1,n)))<mincost)
        {
                mincost=c[tour[start]][tour[i]]+ccost;
                for(k=1;k<=n;k++)
                        mintour[k]=temp[k];
}
}
for(i=1;i<=n;i++)
        tour[i]=mintour[i];
return mincost;
}
}
```

OUTPUT:

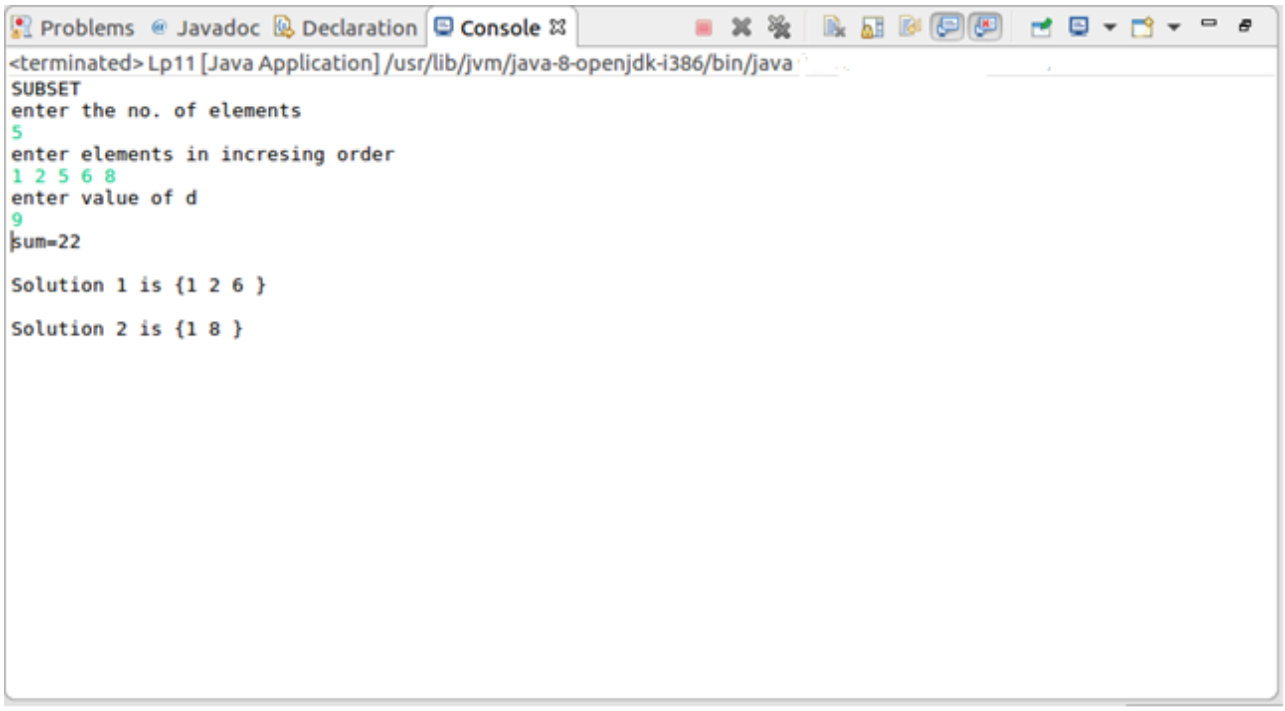
```
Problems  Javadoc  Declaration  Console
<terminated> Lp10B [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
TSP by Dynamic Programming
Enter no. of cities
5
Enter the cost matrix
0 3 999 7 999
3 0 4 2 999
999 4 0 5 6
7 2 5 0 4
999 999 6 4 0
The entered cost matrix is
0       3       999     7       999
3       0       4       2       999
999     4       0       5       6
7       2       5       0       4
999     999     6       4       0
The accurate path is
1->2->3->5->4->1
The accurate cost is 24
```

PROGRAM 11

```
import java.util.*;
public class Lp11 {
static int c=0;
        public static void main(String[] args) {
                // TODO Auto-generated method stub
int w[]=new int[10];
int n,d,i,sum=0;
int x[]=new int[10];
Scanner in=new Scanner(System.in);
System.out.println("SUBSET");
System.out.println("enter the no. of elements");
n=in.nextInt();
System.out.println("enter elements in incresing order");
for(i=0;i<n;i++)
        w[i]=in.nextInt();
System.out.println("enter value of d");
d=in.nextInt();
for(i=0;i<n;i++)
        sum=sum+w[i];
System.out.println("sum="+sum);
if(sum<d)
{
        System.out.println("subset not possible");
        System.exit(0);
}
Subset(0,0,sum,x,w,d);
if(c==0)
System.out.println("subset not posssible");
```

```
}
static void Subset(int cs,int k,int r,int x[],int w[],int d)
{
x[k]=1;
if(cs+w[k]==d)
{
        c++;
        System.out.print("\nSolution "+c+" is {");
        for(int i=0;i<=k;i++)
                if(x[i]==1)
                {
                        System.out.print(w[i]+" ");
                }
        System.out.println("}");
}
else if((cs+w[k]+w[k+1])<=d)
        Subset(cs+w[k],k+1,r-w[k],x,w,d);
        if((cs+r-w[k])>=d&&(cs+w[k+1])<=d)
                {
                        x[k]=0;
                        Subset(cs,k+1,r-w[k],x,w,d);
                }

}
}
```

OUTPUT:



```
Problems  Javadoc  Declaration  Console
<terminated> Lp11 [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
SUBSET
enter the no. of elements
5
enter elements in incresing order
1 2 5 6 8
enter value of d
4
sum=22
subset not posssible
```

PROGRAM 12

```
import java.util.*;
public class Lp12 {
int count;
int path=1;
	public static void main(String[] args) {
		// TODO Auto-generated method stub
int graph[][]=new int[10][10];
Scanner in=new Scanner(System.in);
System.out.println("Hamiltonian Cycles");
System.out.println("enter no. of nodes");
int n=in.nextInt();
System.out.println("enter the adjacency matrix");
for(int i=0;i<n;i++)
{
	for(int j=0;j<n;j++)
	{
		graph[i][j]=in.nextInt();
	}
}
int arr[]=new int[n];
for(int i=0;i<n;i++)
	arr[i]=i;
System.out.println("All possible Hamiltonian Cycles are");
new Lp12().Permute(arr,graph);
}
void Permute(int arr[],int graph[][])
{
Permute(arr,0,arr.length-1,graph);
```

```
}
void Permute(int arr[],int i,int n,int cost[][])
{
int j;
if(i==n)
{
	HamCycle(arr,cost);
}
else
{
	for(j=i;j<=n;j++)
	{
		Swap(arr,i,j);
		Permute(arr,i+1,n,cost);
		Swap(arr,i,j);
	}
}
}
void HamCycle(int a[],int graph[][])
{
	count=0;
	for(int i=0;i<a.length-1;i++)
	{
		if(graph[a[i]][a[i+1]]!=0)
			count++;
	}
	if(count==a.length-1&&graph[a[a.length-1]][a[0]]==1)
	{
		System.out.println("cycle no."+path+"->");
		for(int i=0;i<a.length;i++)
			System.out.print(a[i]+" ");
		System.out.println(a[0]);
		System.out.println();
		path++;
	}
}
void Swap(int a[],int i,int j)
{
	int temp=a[i];
	a[i]=a[j];
	a[j]=temp;
}
}
```

OUTPUT:

Problems Javadoc Declaration Console

```
<terminated> Lp12 [Java Application] /usr/lib/jvm/java-8-openjdk-i386/bin/java
Hamiltonian Cycles
enter no. of nodes
4
enter the adjacency matrix
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
All possible Hamiltonian Cycles are
cycle no.1->
0 1 3 2 0

cycle no.2->
0 2 3 1 0

cycle no.3->
1 0 2 3 1

cycle no.4->
1 3 2 0 1

cycle no.5->
2 0 1 3 2

cycle no.6->
2 3 1 0 2

cycle no.7->
3 1 0 2 3
```