// 1a) lab program(student details)

```java
package studentdetails;

import java.util.Scanner;

class Student //create class
{

	String USN, Name, Branch, Phone;

	Scanner input = new Scanner(System.in);

	void read() //to read student details
	{
		System.out.println("Enter Student Details");
		System.out.println("Enter USN");
		USN = input.nextLine();

		System.out.println("Enter Name");
		Name = input.nextLine();

		System.out.println("Enter Branch");
		Branch = input.nextLine();

		System.out.println("Enter Phone");
		Phone = input.nextLine();
	}

	void display()//to display student details
	{
		System.out.printf( USN+"   "+Name+"   "+Branch+"   "+Phone);
	}

}
public class studentdetails
{

	public static void main(String[] args)
	{

		Scanner input = new Scanner(System.in);
		System.out.println("Enter number of student details to be created");
		int number = input.nextInt();

		Student s[] = new Student[number];

		// Read student details into array of student objects

		for (int i = 0; i < number; i++)
		{
			s[i] = new Student();
			s[i].read();//reads the student details from class Student and st
		}

		// Display student information

		System.out.println("Student details :");
		System.out.println("  ");
		System.out.println("USN   NAME  BRANCH  PHONE");
		for (int i = 0; i < number; i++)
		{
			s[i].display();//calls the display function from class Student
		}

	}


}
```

// 1b lab program(stack)

```
package stacks;
import java.util.Scanner;
public class stacks
{
	public static void main(String[] args)
	{
		int top=-1;
		int ele,n,i;
		Scanner s = new Scanner(System.in);//Scanner method used to take input
		System.out.println("Enter Stack Size");
		n = s.nextInt();//java Scanner class method is used to scan no of elements
in stack.
		int a[]  = new int[n+1];//array declaration
		System.out.println("enter your choice");
		System.out.println("1.push\n 2.pop\n 3.display\n");
		for(;;)
		{
			System.out.println("enter your choice\n");
			int choice = s.nextInt();//java Scanner class method is used to
scan next token choice.
			switch (choice)
			{
				case 1:if(top==n)//maximum condition
				{
					System.out.println("stack overflow\n");
				}
				else
				{
					System.out.println("Enter element to be pushed");
					ele = s.nextInt();//java Scanner class method is
used to scan next token element pushed to stack.
					a[++top] = ele;
				}
				break;
				case 2: if(top == -1)//stack underflow condition
				{
					System.out.println("Stack Underflow");
				}
				else
				{
					System.out.println("Popped element " + a[top--]);
				}
				break;
				case 3: if(top== -1)//stack empty condition
				{
					System.out.println("Stack Empty");
				}
				else
				{
					System.out.println("Elements in stack :");
					for ( i = top; i >= 0; i--)
					{
						System.out.println(a[i]);
					}
				}
				break;
				case 4: System.exit(0);
				break;

			}
		}
	}
}
```

// 2a) lab program(staff details)

```
package staffinfo;
import java.util.Scanner;
class Staff {
	String StaffID, Name, Phone, Salary;

	Scanner input = new Scanner(System.in);

	void read() {
		System.out.println("Enter StaffID");
		StaffID = input.nextLine();

		System.out.println("Enter Name");
		Name = input.nextLine();

		System.out.println("Enter Phone");
		Phone = input.nextLine();

		System.out.println("Enter Salary");
		Salary = input.nextLine();
	}

	void display() {
		System.out.println("STAFFID: " +StaffID);
		System.out.println("NAME: "+Name);
		System.out.println("PHONE:"+Phone);
	 System.out.println("SALARY:"+Salary);
	}




}


class Teaching extends Staff {
	String Domain, Publication;

	void read_Teaching() {
		super.read(); // call super class read method
		System.out.println("Enter Domain");
		Domain = input.nextLine();
		System.out.println("Enter Publication");
		Publication = input.nextLine();
	}

	void display() {
		super.display(); // call super class display() method
		System.out.println("DOMAIN:"+Domain);
		System.out.println("PUBLICATION:"+Publication);

	}
}


class Technical extends Staff {
	String Skills;
	void read_Technical() {
		super.read(); // call super class read method
		//super.read_Teaching();
		System.out.println("Enter Skills");
		Skills = input.nextLine();
	}

	void display() {
		super.display(); // call super class display() method
		System.out.println("SKILLS:"+Skills);
```

```
	}
}


class Contract extends Staff {
	String Period;

	void read_Contract() {
		super.read(); // call super class read method
		System.out.println("Enter Period");
		Period = input.nextLine();
	}

	void display() {
		super.display(); // call super class display() method
		System.out.println("PERIOD:"+Period);

	}
}



public class staffinfo {
	public static void main(String[] args) {

		Scanner input = new Scanner(System.in);

		System.out.println("Enter number of staff details to be created");
		int n = input.nextInt();//here n is to store  staff data,

		Teaching steach[] = new Teaching[n];
		Technical stech[] = new Technical[n];
		Contract scon[] = new Contract[n];

		// Read Staff information under 3 categories

		for (int i = 0; i < n; i++) {
			System.out.println("Enter Teaching staff information");
			steach[i] = new Teaching();
			steach[i].read_Teaching();
		}

		for (int i = 0; i < n; i++) {
			System.out.println("Enter Technical staff information");
			stech[i] = new Technical();
			stech[i].read_Technical();
		}

		for (int i = 0; i < n; i++) {

			System.out.println("Enter Contract staff information");
			scon[i] = new Contract();
			scon[i].read_Contract();
		}

		// Display Staff Information
		System.out.println("\n STAFF DETAILS: \n");
		System.out.println("-----TEACHING STAFF DETAILS----- ");

		for (int i = 0; i < n; i++) {
			steach[i].display();
		}

		System.out.println();
		System.out.println("-----TECHNICAL STAFF DETAILS-----");
		for (int i = 0; i < n; i++) {
			stech[i].display();
		}

		System.out.println();
```

```
                System.out.println("-----CONTRACT STAFF DETAILS-----");
                for (int i = 0; i < n; i++) {
                        scon[i].display();
                }

                //input.close();
        }


}
```

2b) lab program(string tokenizer)

```
package customer;
import java.util.Scanner;
import java.util.StringTokenizer;
public class customer {
        public static void main(String[] args)
        {
                String name;
                Scanner scan = new Scanner(System.in);
                System.out.println("Enter Name and Date_of_Birth in the format
<Name,DD/MM/YYYY>");
                name = scan.next();
                StringTokenizer st = new StringTokenizer(name, ",/");
                // Count the number of tokens
                int count = st.countTokens();// Print one token at a time and induce new
delimiter ","

                for (int i = 1; i <= count; i++)
                {
                        System.out.print(st.nextToken());
                        if (i < count)
                        {                               System.out.print(",");
                        }
                }
        }


}
```

// 3a) lab program(exception handling)

```
package exception;
import java.util.Scanner;

public class exception
{
        public static void main(String args[])
        {
                int a, b, result;
                Scanner input = new Scanner(System.in);
                System.out.println("Input two integers");
                a = input.nextInt();
                b = input.nextInt();
                try
                {
                        result = a / b;
                        System.out.println("Result = " + result);
                }
                catch (ArithmeticException e)
         {
                        System.out.println(e);
```

```
            }

        }

}
```

// 3b) lab program(multi threading)

```
package mainthread;
import java.util.Random;

class SquareThread implements Runnable {
        int x;

        SquareThread(int x) {
                this.x = x;
        }

        public void run() {
        System.out.println("Thread Name:Square Thread and Square of " + x + " is: " + x *
x);
        }
}

class CubeThread implements Runnable {
        int x;

        CubeThread(int x) {
                this.x = x;
        }

        public void run() {
        System.out.println("Thread Name:Cube Thread and Cube of " + x + " is: " + x * x *
x);
        }
}

class RandomThread implements Runnable {
        Random r;
        Thread t2, t3;

        public void run() {
                int num;
                r = new Random();
                try {

                        while (true) {
                                num = r.nextInt(5);
                                System.out.println("Main Thread and Generated Number is " +
num);
                                t2 = new Thread(new SquareThread(num));
                                t2.start();

                                t3 = new Thread(new CubeThread(num));
                                t3.start();

                                Thread.sleep(1000);
                                System.out.println("-------------------------------------
");
                        }
                } catch (Exception ex) {
                        System.out.println("Interrupted Exception");
                }
        }
}
```

```
public class mainthread {
	public static void main(String[] args) {

		RandomThread thread_obj = new RandomThread();
		Thread t1 = new Thread(thread_obj);
		t1.start();
	}
}
```

==========
==========

//4) lab program (merge sort)

```
package mergesort;
import java.util.*;

public class mergesort {

	public static void main(String[] args) {
		Scanner input = new Scanner(System.in);
		System.out.println("enter the elements to be sort:");
		int n =input.nextInt();
		int [] inputArr = new int [n];
		Random rand = new Random();
		for(int i=0;i<n;i++)
		{
			inputArr[i]=rand.nextInt(10);
			System.out.println(inputArr[i]+" ");
		}
		System.out.println();
		long startTime = System.nanoTime();
		mergeSort(inputArr, n);
		long estimatedTime = System.nanoTime() - startTime;
		System.out.println();
		System.out.println("sorted elemts are:");
		for (int i=0;i<n;i++)
		{
			System.out.println(inputArr[i]+" ");
		}
		System.out.println();
		System.out.println("The time for sorting
is"+(estimatedTime/1000000000.0)+" secs ");

		}
		static void mergeSort(int a[],int n)
		{
			int b[] = new int [n/2];
			int c[] = new int [n-n/2];
			int i, j;
			if(n>1)
			{
				for(i=0;i<n/2;i++)
				{
					b[i]=a[i];
				}
				for(i=n/2,j=0;i<n;i++,j++)
				{
					c[j]=a[i];
				}
				mergeSort(b, n/2);
				mergeSort(c, n-n/2);
				merge(b, c, a, n/2 ,n-n/2, n);
			}
	}
	static void merge(int b[], int c[], int a[], int p, int q, int n)
	{
			int i,j,k;
```

```
                i=j=k=0;
                while(i<p && j<q)
                {
                        if(b[i]<=c[j])
                        {
                                a[k]=b[i];
                                i++;
                        }
                        else
                        {
                                a[k]=c[j];
                                j++;
                        }
                        k++;
                }
                if(i==p)
                {
                        while(j<q)
                        {
                                a[k++]=c[j++];

                        }
                }
                else
                {
                        while(i<p)
                        {
                                a[k++]=b[i++];
                        }
                }
        }


}
```

==========

// 5) lab program (quick sort)

```
package quicksort;
import java.util.*;
public class quicksort {

        public static void main(String[] args) {

                Scanner input = new Scanner(System.in);
                System.out.print("Enter the number of elements to sort:");


                int n = input.nextInt();
                int inputArr[]= new int[n+1];
                Random rand = new Random();
                for( int i=0; i<n ; i++)
                {
                inputArr[i]=rand.nextInt(10);
                System.out.print(inputArr[i] + " ");
                }
                System.out.println();
                long startTime = System.nanoTime();
                quicksort(inputArr, 0, n-1);
                long estimatedTime = System.nanoTime() - startTime;
                System.out.println();
                System.out.println("After Sorting");
                for( int i=0; i<n ; i++)
                {
                System.out.print(inputArr[i] +" ");
                }
                System.out.println();
                System.out.println("The time for sorting is " +
                (estimatedTime/1000000000.0)+ " secs");
```

```
		input.close();
		}
		static void quicksort(int a[],int low,int high)
		{
		int j;
		if(low<high)
		{
		j=partition(a,low,high);
		quicksort(a,low,j-1);
		quicksort(a,j+1,high);
		}
		}
		static int partition(int a[],int low,int high)
		{
		int i,j;
		int pivot;
		pivot=a[low];
		i=low;
		j=high+1;
		do
		{
		do i++; while(a[i]<pivot);
		do j--; while(a[j]>pivot);
		if(i<j)
		 swap(a,i,j);
		}while(i<j);
		swap(a,low,j);
		return j;
		}
		static void swap(int a[],int i,int j)
		{
		int temp;
		temp=a[i];
		a[i]=a[j];
		a[j]=temp;
		}


	}
```

==========

// 6a) lab program(greedy knapsack)

```
package knapsack;
import java.util.*;
public class knapsack
{

	public static void main(String[] args)
	{
		int i ;
		Scanner in = new Scanner(System.in);
		System.out.println("*****knapsack*****");
		System.out.println("enter the number of items:");
		int n = in.nextInt();
		float w[]=new float[n+1];
		float p[]=new float[n+1];
		float ratio[]=new float[n+1];
		System.out.println("enter the weight of each item");
		for(i=1;i<=n;i++)
			w[i]=in.nextFloat();
		System.out.println("enter the profit of each item");
		for(i=1;i<=n;i++)
			p[i]=in.nextFloat();
		System.out.println("enter the knapsack capacity");
		int m = in.nextInt();
		for(i=1;i<=n;i++)
			ratio[i]=p[i]/w[i];
```

```
                System.out.println("information about knapsack problemare:");
                displayinfo(n,w,p,ratio);
                System.out.println("capacity = "+m);
                SortArray(n,ratio,w,p);
                System.out.println("details for sorting items based on profit,weight,ratio
in descending order ");
                displayinfo(n,w,p,ratio);
                knapsack1(m,n,w,p);
                System.out.println("*********");
        }
        static void  displayinfo(int n,float w[],float p[],float ratio[])
        {
                System.out.println("ITEM \t WEIGHT \t PROFIT \t RATIO[p/w] \t ");
                for(int i=1;i<=n;i++)
                        System.out.println(i+"\t"+w[i]+"\t"+p[i]+"\t"+ ratio[i]);
        }

        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-1;j++)
                        {
                                if(ratio[j]<ratio[j+1])
                                {
                                        float temp;
                                        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 knapsack1(int n,int m,float w[],float p[])
        {
                        float x[]=new float[n+1];
                        float tp=0;
                        int i;
                        int u=m;
                        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+1;
                                        u=(int)(u-w[i]);
                                }
                        }
                        if(i<n)
                                x[i]=u/w[i];
                        tp=tp+(x[i]*p[i]);
                        System.out.println("\n the result is = ");
                                for(i=1;i<=n;i++)
                        System.out.println("\t"+x[i]);
                        System.out.println("\n max profit is = "+tp);
        }


}
```

{

// 6b) lab program (dynamic knapsack)

```
import java.util.Scanner;
public class Knapsack_DP
{
public static void main(String[] args)
{
int n,i,j,capacity;
int w;
Scanner sc = new Scanner(System.in);
System.out.println("Enter the number of items: \n");
n = sc.nextInt();
int weight[]=new int[n+1],value[] = new int[n+1];
int V[][]=new int[n+2][n+2];
System.out.println("\nWEIGHTS - VALUES");
for(i=1;i<=n;i++)
{
weight[i]=sc.nextInt();
value[i] = sc.nextInt();
}
System.out.println("Enter the capacity of kanpsack \n");
capacity = sc.nextInt();
for(i=0;i<=n;i++)
{
 for(j=0;j<=capacity;j++)
 {
 if(i==0 || j==0)
V[i][j]=0;
 else if ( j-weight[i]>=0 )
 V[i][j]=max(V[i-1][j] ,V[i-1][j-weight[i]]+value[i]);
 else
V[i][j]=V[i-1][j];
 System.out.print(" "+V[i][j]);
 }
System.out.print("\n");
}
w=capacity;
System.out.println("The item in the knapsack \n");
for(i=n;i>0;i--)
{
 if(V[i][w]==V[i-1][w])
continue;
 else
 {
w=w-weight[i];
System.out.println("item="+i+"and weight="+weight[i]);
 }
```

```
}
System.out.println("\n Total profit="+V[n][capacity]);
}
static int max(int a,int b)
{
if(a>b)
return a;
else
return b;
}
}
```

==========

// 7) lab program (dijkstras)

```
package dijkstras;

import java.util.Scanner;

public class dijkstras {

	public static void main(String[] args) {
		int n,source,i,j;
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the number of vertices\n");
		n=sc.nextInt();
		int cost[][]=new int [n+1][n+1];
		int dist[]=new int[n+1];
		System.out.println("Enter the cost adjacency matrix\n");
		for(i=1;i<=n;i++)
		for(j=1;j<=n;j++)
		{
		cost[i][j]=sc.nextInt();
		if(cost[i][j]==0)
		cost[i][j]=999;
		}
		System.out.println("source\n");
		source=sc.nextInt();
		dijikstras(cost,dist,n,source);
		for(i=1;i<=n;i++)
		if(source!=i)
		System.out.println(source+"->"+i+"::"+dist[i]);
		}
		static void dijikstras(int cost[][],int dist[],int n,int v)
		{
		int i,u=0,w,count,min;
		int flag[]=new int[n+1];
		for(i=1;i<=n;i++)
		{
		flag[i]=0;
		dist[i]=cost[v][i];
		}
		flag[v]=1;
		dist[v]=0;
```

```
                count=2;
                while(count<n)
                {
                for(i=1,min=999;i<=n;i++)
                {
                if((dist[i]<min) && (flag[i]==0))
                {
                min=dist[i];
                u=i;
                }
                }
                flag[u]=i;
                count++;
                for(w=1;w<=n;w++)
                {
                if((dist[u]+cost[u][w]<dist[w]) && (flag[w]==0))
                dist[w]=dist[u]+cost[u][w];
                }
                }
                }


        }
```

==========

// 8) lab program (kruskals)

```
package kruskals;
import java.util.Scanner;


public class kruskals {
        public static void main(String[] args)
        {
        Scanner sc=new Scanner(System.in);
        int mincost=0,n,i,j,ne,a = 0,b = 0,min,u = 0,v = 0;
        System.out.println("Enter the number of vertices\n");
        n=sc.nextInt();
        int cost[][]= new int [n+1][n+1];
        int parent[]=new int[n+1];
        System.out.println("Enter the cost matrix\n");
        for(i=1;i<=n;i++)
        {
        for(j=1;j<=n;j++)
        {
        cost[i][j]=sc.nextInt();
        if(cost[i][j]==0)
        cost[i][j]=999;
        }
        }
        ne=1;
        while(ne<n)
        {
        for(min=999,i=1;i<=n;i++)
        {
        for(j=1;j<=n;j++)
        if(cost[i][j]<min)
        {
        min=cost[i][j];
        a=u=i;
        b=v=j;
        }
        }
        while(parent[u]!=0)
        u=parent[u];
```

```
        while(parent[v]!=0)
        v=parent[v];
        if(v!=u)
        {
        System.out.println((ne++)+"edge("+a+","+b+")="+min);
        mincost+=min;
        parent[v]=u;
        }
        cost[a][b]=cost[b][a]=999;
        }
        System.out.println("The minimum cost of spanning tree is "+mincost);
        }

}
```

==========

// 9) lab program (prims)

```
package prims;
import java.util.Scanner;
public class prims {

        public static void main(String[] args) {


                Scanner sc= new Scanner(System.in);
                int mincost=0,n,i,j,ne,a=0,b=0,min,u = 0,v=0;
                System.out.println("Enter the number of vertices\n");
                n=sc.nextInt();
                int cost[][]= new int [n+1][n+1];
                int visited[]=new int[n+1];
                System.out.println("Enter the cost matrix\n");
                for(i=1;i<=n;i++)
                {
                for(j=1;j<=n;j++)
                {
                cost[i][j]=sc.nextInt();
                if(cost[i][j]==0)
                cost[i][j]=999;
                }
                }
                for(i=2;i<=n;i++)
                visited[i]=0;
                visited[1]=1;
                ne=1;
                while(ne<n)
                {
                for(min=999,i=1;i<=n;i++)
                {
                for(j=1;j<=n;j++)
                {
                if(cost[i][j]<min)
                {
                if(visited[i]==0)
                continue;
                else
                {
                min=cost[i][j];
                a=u=i;
                b=v=j;
                }
                }
                }
                }
                if(visited[u]==0||visited[v]==0)
                {
                System.out.println((ne++)+"edge("+a+","+b+")="+min);
```

```
                mincost+=min;
                visited[v]=1;
                }
                cost[a][b]=cost[b][a]=999;
                } //end of while
                System.out.println("The minimum cost of spanning tree is "+mincost);
                }
                }
```

==========

// 10) lab program(floyd's)

```
package floydsclass;
import java.util.Scanner;
public class floydsclass {

        static final int MAX = 20;    // max. size of cost matrix
        static int a[][];                     // cost matrix
        static int n;                         // actual matrix size

        public static void main(String args[]) {
                a = new int[MAX][MAX];
                ReadMatrix();
                Floyds();                             // find all pairs shortest path
                PrintMatrix();
        }

        static void ReadMatrix() {
                System.out.println("Enter the number of vertices\n");
                Scanner scanner = new Scanner(System.in);
                n = scanner.nextInt();
                System.out.println("Enter the Cost Matrix (999 for infinity) \n");
                for (int i = 1; i <= n; i++) {
                        for (int j = 1; j <= n; j++) {
                                a[i][j] = scanner.nextInt();
                        }
                }
                scanner.close();
        }

        static void Floyds() {
                for (int k = 1; k <= n; k++) {
                        for (int i = 1; i <= n; i++)
                                for (int j = 1; j <= n; j++)
                                        if ((a[i][k] + a[k][j]) < a[i][j])
                                                a[i][j] = a[i][k] + a[k][j];
                }
        }

        static void PrintMatrix() {
                System.out.println("The All Pair Shortest Path Matrix is:\n");
                for(int i=1; i<=n; i++)
                {
                        for(int j=1; j<=n; j++)
                                System.out.print(a[i][j] + "\t");
                        System.out.println("\n");
                }
        }
}
```