1.Write a Java program that takes as input three integers, a, b, and c, from the Java console and determines if they can be used in a correct arithmetic formula (in the given order), like “a + b = c,” “a = b − c,” or “a b = c.”

```
Public class pg1

{

Public static void main(String args[])

{

Scanner in =new Scanner(System.in);

System.out.println(“enter a,b,c”);

Int a =in.nextInt();

Int b =in.nextInt();

Int c =in.nextInt();

If(a+b==c)

{

System.out.println(“It is correct”);

}

Else if(b-c==a)

{

System.out.println(“It is correct”);

}

Else if(a*b==c)

{

System.out.println(“It is correct”);

}

Else if(a+b==c&&b-c==a&&a*b==c)

{

System.out.println(“all are correct”);

}

Else

{

System.out.println(“nothing is correct”);

}}
```

2. Write a Java program that can take a positive integer greater than 2 as input and write out the number of times one must repeatedly divide this number by 2 before getting a value less than 2

```
Import java.util.*;

Class pg2

{

Public static void main(String args[])

{

Scanner in =new Scanner(System.in);

System.out.println(“enter a”);

Int a =in.nextInt();

Int q =0;

If(n>=2)

{

While(n>=2)

 n =n/2;

q++;

System.out.println(q);

}}}
```

3. Write a Java program that outputs all possible strings formed by using the characters 'c', 'a', 'r', 'b', 'o', and 'n' exactly once

```
Class pg3

{

Public static void main(String args[])

 char alpha[] ={‘c’,’a’,’r’,’b’,’o’,’n’};

int I,j,k,l,m;

for(i=0;i<alpha.length;i++)

for(j=0;j<alpha.length;j++)

for(k=0;k<alpha.length;k++)

for(l=0;l<alpha.length;l++)

for(m=0;m<alpha.length;m++)

for(n=0;n<alpha.length;n++)

{
```

```
if(i!=j && i!=k && i!=l && i!=m && i!=n && j!=k && j!=l && j!=m && j!=n && k!=l && k!=m && k!=n
&& l!=m && l!=n && m!=n ){

System.out.println(""+ str[i] + str[j]+str[k]+str[l]+str[m]+str[n]);

count +=1;

System.out.println(count);

}}
```

4. Write a Java program that takes all the lines input to standard input and writes them to standard output in reverse order. That is, each line is output in the correct order, but the ordering of the lines is reversed.

```
 import java.util.*;

Class pg4

{

Public static void main(String args[])

{

Scanner in =new Scanner(System.in);

System.out.println("enter value of n");

 int n =in.nextInt();

String str[] =new String[n];

System.out.println("Enter the line");

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

str[i] =in.nextLine();

for(i=n-1;i>=0;i++)

System.out.println(str[i]);

}}
```

5. Write a Java method, isOdd, that takes an int i and returns true if and only if i is odd. Your method can't use the multiplication, modulus, or division operators, however.

```
import java.util.*;

Class pg5

{

Public static void main(String args[])

{

Scanner in =new Scanner(System.in);
System.out.println("enter value of n");
 int n =in.nextInt();
boolean b =isOdd(n);
if(b==true)
System.out.println("odd");
}
Public static  boolean isOdd(int n)
{
While(n>1)
{
 n =n-2;
}
If(n==1)
 return true;
 else
 return false;
}}
```

6. Write a Java method for finding the smallest and largest numbers in an array of integers and compare that to a java method that would do the same thing.

```
import java.util.*;
Class pg6
{
 public static  int min(int a[],int num)
{
Int min =a[0];
for(int i =1;i<num;i++)
{
If(a[i]<min)
{
  Min =a[i];
  return min;
}}}
Public static int max(int b[],int num)
{
Int max =b[0];
  for(int i=1;i<num;i++)
{
If(b[i]>max)
{
  max =b[i];
return max;
}}}
   public static void main(String args[])
{
Scanner in =new Scanner(System.in);
Int arr[] =new int[20];
System.out.println("How many no. do u wnt to enter");
        int arrcount =in.nextInt();
System.out.println("Enter elements");
   for(i=0;i<arrcount;i++)
{
System.out.println("enter"+i+"no.");
   arr[i] =in.nextInt();
}}}
```

7. Write a Java method that takes an array of int values and determines if there is a pair of distinct elements of the array whose product is odd.

```
import java.util.*;
Class pg7
{
 public static  void product(int a[],int c)
{
   for(int i =0;i<c;i++)
```

```
{
   for(int j =0;j<c;j++)
{
Int pro =0;
If(a[i]==a[j])
 continue;
pro =a[i]*a[j];
if(pro%2==1)
{
System.out.println("The product of"+a[i]+"and"+a[j]+"isOdd");
}}
public static void main(String args[])
{
Scanner in =new Scanner(System.in);
Int arr[] =new int[20];
System.out.println("How many no. do u wnt to enter");
Int c =in.nextInt();
for(i=0;i<c;i++)
{
System.out.println("enter"+(i+1)+"number");
a[i] =in.nextInt();
}
product(c,a);
}}
```

8. Write a Java program that takes two arrays a and b of length n storing int values, and returns the dot product of a and b. That is, it returns an array c of length n such that c[i] = a[i] · b[i], for i = 0, . . . , n − 1.

```
import java.util.*;
Class pg8
{
Public static void main(String args[])
{
```

```
Scanner in =new Scanner(System.in);

int a[] =new int[20];

int b[] =new int[20];

int c[] =new int[20];

System.out.println("How many no. do u wnt to enter");

Int n =in.nextInt();

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

{

System.out.println("enter elements in a[]");

A[i] =in.nextInt();

}

System.out.println("enter elements in b[]");

b[i] =in.nextInt();

{

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

{

C[i] =a[i]*b[i];

System.out.println(c[i]);

}}}
```

9. Create a class Student with instance variables name, roll, mark and instance methods setData(), display(). Write a Java program to create three objects of Student class to input details of three different students and display the details. Enclose main() method inside another class StudentDetails. (Use the setter method setData() to input details.)

```
public class Student {

        String name;

        int roll;

        double mark;

        public void setData(String n, int r, double m)

        {

                name = n;

                roll = r;
```

```
		mark = m;
	}
	public void display()
	{
		System.out.println(name+" "+roll+" "+mark+" ");
	}
}

public class StudentDetails {

	public static void main(String[] args) {
		Student s1 = new Student();
		//s1.name="manas";
		//s1.roll=123;
		//s1.mark=97.5;
		s1.setData("manas",123,97.5);
		s1.display();
		Student s2 = new Student();
		//s2.name="tapas";
		//s2.roll=456;
		//s2.mark=75;
		s2.setData("tapas",456,75);
		s2.display();

	}

}
```

11. Create a class Point with instance variables x, y to represent co-ordinates of point and instance method setPoint(). Write a Java program to find distance between two points using a method findDistance(Point,Point).

```
public class Point {
	double x,y;
	public void setPoint(double c1, double c2)
	{
		x=c1;
		y=c2;
	}
}

public class PointDistance {

	public static void main(String[] args) {
		Point p1 = new Point();
		Point p2 = new Point();
		p1.setPoint(2,2);
		p2.setPoint(4,4);
		findDistance(p1,p2);

	}

	public static void findDistance(Point p1, Point p2)
	{
		double dist;
		dist = Math.sqrt(Math.pow(p1.x-p2.x, 2)+Math.pow(p1.y-p2.y, 2));
		System.out.printf("%.2f",dist);
	}
}
```

12. Write a Java class Flower that has three instance variables of type String, int, and float, which respectively represent the name of the flower, its number of petals, and price. Your class must include a method that initializes each variable to an appropriate value, and your class should include methods for setting the value of each type, and getting the value of each type.

```
public class Flower {
	String name;
	int nopetals;
```

```
        float price;

        public void setData(String n, int nop, float p)
        {
                name = n;
                nopetals = nop;
                price = p;
        }

        public void getData()
        {
                System.out.println(name+" "+nopetals+" "+price);
        }
}.

public class FlowerDetails {

        public static void main(String[] args) {
                Flower f1 = new Flower();
                f1.setData("Rose", 15, 10);
                f1.getData();
                f1.setData("Lily", 10, 100);
                f1.getData();
        }

}
```

1.Write a class, Commission, which has: An instance variable, sales; an appropriate constructor; and a method, getCommission() that returns the commission. Now write a Demo class in Java to test the Commission class by reading a sale from the user, using it to create a Commission object after validating that the value is not negative. Finally, call the getcommission() method to get and print the commission. If the sales are negative, your Demo class should print the message “Invalid Input”.

```
import java.util.Scanner;

class comission
{
        double sale ,x;
        comission(int s)
        {
                sale=s;
        }

        double getComisson()
        {
                if(sale<500)
                        x=0.02*sale;
                if(sale>500&&sale<5000)
                        x=sale*0.05;
                if(sale>5000)
                        x=sale*0.08;
return x;
        }
}
public class democomission {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
Scanner sc=new Scanner(System.in);
System.out.println("sale is");
int s=sc.nextInt();
if(s<0)
        System.out.println("invlid output");
else
{
        comission obj=new comission(s);
        double res=obj.getComisson();
        System.out.println("COMISSION=  "+res);
}
        }

}
```

2. Define a class called Complex with instance variables real, imag and instance methods setData(), display(), add(). Write a Java program to add two complex numbers

```
import java.util.Scanner;


public class complex {
	int re,im;
	void setdata()
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("enter real and imagenary number");
		re=sc.nextInt();
		im=sc.nextInt();
	}
	void display()
	{
		System.out.println("the complex number is"+re+" +i "+im);
	}
	complex add(complex obj1,complex obj2)
	{
		complex obj=new complex();
		obj.re=obj1.re+obj2.re;
		obj.im=obj1.im+obj2.im;
		return obj;
	}
	public static void main(String[] args) {
		// TODO Auto-generated method stub
complex obj1=new complex();
complex obj2=new complex();

obj1.setdata();
obj2.setdata();
complex obj=obj1.add(obj1, obj2);
obj.display();


	}

}
```

3. Write a Java program to declare a Class named as Student which contains roll number, name and course as instance variables and input_Student () and display_Student () as instance methods. A derived class Exam is created from the class Student . The derived class contains mark1, mark2, mark3 as instance variables representing the marks of three subjects and input_Marks () and display_Result () as instance methods. Create an array of objects of the Exam class and display the result of 5 students.

```
import java.util.Scanner;

 class studnt1 {
int roll;String nm,cr;
void inputstudnt()
{
	Scanner sc =new Scanner(System.in);
	System.out.println("enter name ,roll,course");
	nm=sc.nextLine();
	roll=sc.nextInt();
	cr=sc.next();
}
void display()
{
	System.out.println("roll= "+roll+"name "+nm+"course "+cr);
}

}


class exam extends studnt1
{
	int m1,m2,m3,s;
	public void inputmrk()
	{
		Scanner sc =new Scanner(System.in);
		System.out.println("enter marks");
		m1=sc.nextInt();
		m2=sc.nextInt();
		m3=sc.nextInt();
	}
	public void dislay_result()
	{
		 s=m1+m2+m3;
		System.out.println(s);
	}
}
class studnt
{
	public static void main(String[] args) {
```

```
exam e[]=new exam[5];
for(int i=0;i<5;i++)
{
        e[i]=new exam();

}
for(int i=0;i<5;i++)
{
        e[i].inputstudnt();
        e[i].inputmrk();
        e[i].display();
        e[i].dislay_result();
}
}
}
```

4. A point in the x-y plane is represented by its x-coordinate and y-coordinate. Design a class, PointType in Java, that can store and process a point in the x-y plane. You should then perform operations on the point, such as showing the point, setting the coordinates of the point, printing the coordinates of the point, returning the x-coordinate, and returning the ycoordinate. Every circle has a center and a radius. Given the radius, we can determine the circle's area and circumference. Given the center, we can determine its position in the x-y plane. The center of a circle is a point in the x-y plane. Design a class, CircleType that can store the radius and center of the circle. Because the center is a point in the x-y plane and you designed the class to capture the properties of a point from PointType class. You must derive the class CircleType from the class PointType. You should be able to perform the usual operations on a circle, such as setting the radius, printing the radius, calculating and printing the area and circumference, and carrying out the usual operations on the center

```
import java.util.Scanner;

class pointtype
{
        int x,y;
        void set()
        {
                Scanner sc=new Scanner(System.in);
                System.out.println(" enter x,y coordinate");
                x=sc.nextInt();
                y=sc.nextInt();
        }
        void print()
        {
                System.out.println("  x,y coordinate are "+x+" "+y);
        }
}
class circletype extends pointtype
{
        int r;
void distance(circletype c1,circletype c2)
{
        r=(int) Math.sqrt(((c1.x-c2.x)*(c1.x-c2.x))-((c1.y-c2.y)*(c1.y-c2.y)));
        System.out.println("radius"+r);
}
void findarea()
{
        double area=3.14*r*r;
        System.out.println("area "+area);
}
void circ(){
        double c=2*3.14*r;
        System.out.println(" circumference"+c);
}
}
public class point_type {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
circletype c1=new circletype();
circletype c2=new circletype();
c1.set();
c1.print();
c2.set();
c2.print();
circletype c3=new circletype();
c3.distance(c1, c2);
c3.findarea();
c3.circ();
        }

}
```

5. Design a class in Java called CreditCard as shown in Figure 1. which creates the credit card objects that model a simplified version of traditional credit cards. They store information about the customer, issuing bank, account identifier, credit limit, and current balance. They do not charge interest or late payments, but they do restrict charges that would cause a card's balance to go over its credit limit.

```
Class creditcard
{
Private String customer;
Private String bank,account;
Private int limit;
Protected double balance;
CreditCard(String name,String bname,String accn,int lim,double bal)
{
Customer =name;
Bank =bname;
```

```
Account =accn;
Limit =lim;
Balance =bal;
}
Creditcard(String name,String bname,String accn,int lim)
{
This(name,bname,accn,lim,0);
}
String getCustomer()
{
Return customer;
}
String getLimit()
{
Return bank;
}
String getAccount()
{
Return account;
}
Int getLimit()
{
Return Limit;
}
Double getBalance()
{
Return balance;
}
public boolean charge(double price)
{
Boolean flag =true;
If(balance+price<=limit)
Flag =false;
Return flag;
}
Void makePayment(double amount)
{
Boolean flag =charge(amount)
If(flag==false)
Balance =balance+amount;
Else
System.out.println("limit of credit card crossed");
}
Public static void printSummary(CreditCard card)
{
System.out.println("name :"+card.getcustomer());
System.out.println("bank :"+card.getbank());
System.out.println("account :"+card.getAccount());
System.out.println("limit :"+card.getLimit());
System.out.println("balance :"+card.getBalance());

}
```

7. Let a class Person contains data members name and age. A constructor with two arguments is used to assign name and age. Person is of two types a) Student and b) Teacher. class Student contains data members i)course ii) Roll Number and iii)Marks and method display() to display data related to student. Similarly, class Teacher contains data members i) subject_assigned (May take this as a String) ii) contact_hour and method display () to display data related to teacher. Implement this program using base class constructor in derived class.

```
Class Person
{
Int age;
String name;
Person(int a,String n)
{
Age =a;
 name =n;
}
Void display()
{
System.out.println("name="+name);
System.out.println("Age="+age);
}}
Class Student extends Person
{
Int marks,roll;
String course;
Student(String n,int a,int m,int r,String c)
{
Super(a,n);
marks =m;
roll =r;
course =c;
}
void display()
{
super.display();
System.out.println("roll :"+roll);
System.out.println("course :"+course);
System.out.println("Marks :"+marks);
}
}


class Teacher
```

```
{
int contact_hour;
String subject_assignment;
Teacher(String n,int a,String S,int w)
{
super(a,n);
contact_hour =w;
subject_assigned =s;
}
Void display()
{
super display();
System.out.println(“Subject assigned”+ subject_assigned);
System.out.println(“contact hour”+ contact_hour);
}}
Public class test
{
Public static void main(String args[])
{
Student obj =new Student(‘’ABC’’,15,95,21,’’CSE’’);
obj.display();
Teacher obj2 =new Teacher(‘’DEF’’,40,’’COMPUTER’’,20);
obj2.display();
}}
```

8.Create an abstract class Shape and the derived classes Square, Triangle and Circle. Write a java program to display area of different shapes.

```
public abstract class Shape {
	public abstract void area();

}
public class Square extends Shape{
	int length,breadth;
	Square(int a,int b)
	{
		length =a;
		breadth = b;
	}
	public void area()
	{
		System.out.println("Area :"+(length*breadth));
	}
	/*public static void main(String[] args) {
		// TODO Auto-generated method stub
		Shape s = new Square(2,4);
		s.area();
	}*/

}
```

9. Define an interface to declare methods display ( ) & count ( ). Another class contains a static data member maxcount, instance member name & method display ( ) to display name of a person, count the no. of characters present in the name of the person.

```
import java.util.Scanner;

interface aa
{
	void display();
	void count();
}
class bb implements aa
{
	static int maxcount;
	String name;
	bb()
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("entr name of the person");
		name=sc.next();
	}
	public void count()
	{
		for(int i=0;i<name.length();i++)
		{
			maxcount++;
		}
	}
	public void display()
	{
		System.out.println("name"+name);
		System.out.println("no of character= "+maxcount);

	}

}
public class cc {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
bb obj=new bb();
obj.count();
obj.display();
	}

}
```

10. Define a Polygon interface that has methods area( ) and perimeter( ). Then implement classes for Triangle, Quadrilateral, Pentagon, Hexagon,

and Octagon, which implement this interface, with the obvious meanings for the area( ) and perimeter( ) methods. Also
implement classes, IsoscelesTriangle, EquilateralTriangle, Rectangle, and Square, which have the appropriate inheritance relationships. Finally, write a simple user interface, which allows users to create polygons of the various types, input their geometric dimensions, and then output their area and perimeter. For extra effort, allow users to input polygons by specifying their vertex coordinates and be able to test if two such polygons are similar

```
interface polygone
{
        void area();
        void perimeter();
}
class triangle12
{
        int l;
        triangle12(int l1)
        {
                l=l1;
        }
}
class isosceles extends triangle12 implements polygone
{
        int b;
        isosceles(int l1,int b1)
        {
                super(l1);
                b=b1;
        }
        public void area()
        {
        double h=Math.sqrt(l*l-(b*b/4));
        double ar=0.5*b*h;
        System.out.println("area of a isosceles triangle  = " + ar);
        }
        public void perimeter(){
                double peri=2*l+b;
                System.out.println("perimeter of a isosceles triangle = "
+peri);
        }
}
class equil extends triangle12 implements polygone
{
        equil(int l1){
                super(l1);
        }
        public void area(){
                double ar=Math.sqrt((3/4)*l*l);
```

```
		System.out.println("area of equil. triangle is = " +ar);
	}
	public void perimeter(){
		double peri=3*l;
		System.out.println("perimeter of equilateral is = " +peri);
	}
}
class quadrilateral
{
	int l;
	quadrilateral(int l1)
	{
		l=l1;
	}
}
class rectangle extends quadrilateral implements polygone
{
	int b;
	rectangle (int l1,int b1)
	{
		super(l1);
		b=b1;
	}
	public void area(){
		System.out.println("area of rectangle is = " +(l*b));
	}
	public void perimeter()
	{
		System.out.println("peri.of rectangle is = " +(2*(l+b)));
	}
}
class square extends quadrilateral implements polygone
{
	square(int l1){
		super(l1);
	}
	public void area(){
		System.out.println("area of the rectangle = " +l*l);
	}
	public void perimeter(){
		System.out.println("peri. of the rectangle = "+4*l);
	}
}
class pentagone implements polygone
{
	int s;
	pentagone(int s1)
	{
		s=s1;
	}
```

```
        public void area()
        {
                double ar=0.25*Math.sqrt(5*(5+2*Math.sqrt(5)))*s*s;
                System.out.println("Area of pentagone is = " +ar);
        }
        public void perimeter()
        {
                System.out.println("perimeter of pentagone is= " +5*s);
        }
}
class hexagone implements polygone
{
        int s;
        hexagone(int s1)
        {
                s=s1;
        }
public void area(){
        double ar=((3*Math.sqrt(3))/2)*s*s;
        System.out.println("area of hexagone is = " +ar);
}
public void perimeter(){
        System.out.println("perimeter of pentagon is = " +(6*s));
}
}
class octagone implements polygone
{
        int s;
        octagone(int s1)
        {
                s=s1;
        }
        public void area(){
                double ar=2*(1+Math.sqrt(2))*s*s;
                System.out.println("area of octagone is = " +ar);
        }
        public void perimeter(){
                System.out.println("perimeter of octagone is = " +(8*s));
        }
}
public class polygon {

        public static void main(String[] args) {
                // TODO Auto-generated method stub

                polygone p;
                p=new isosceles(5,6);
                p.perimeter();
                p.area();
                p=new equil(5);
```

```
		p.perimeter();
		p.area();
		p=new rectangle(5,6);
		p.perimeter();
		p.area();
		p=new square(5);
		p.perimeter();
		p.area();
		p=new pentagone(5);
		p.perimeter();
		p.area();
		p=new hexagone(5);
		p.perimeter();
		p.area();
		p=new octagone(5);
		p.perimeter();
		p.area();
	}

}
```

1. Write a Java program to read your lucky number from keyboard. Treat –ve no. as NumberFormatException. Write appropriate Exceptional handler.

```
package assignment3;
import java.util.Scanner;
public class question1 {

		public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
System.out.println("input your lucky number");
try
{
int n=sc.nextInt();
if(n<0)
		throw new ArithmeticException("Negative input");
}
catch(Exception e)
{
		System.out.println("Exception "+e);
}
}
}
```

2. Assign your favorite colors in an array. Identify 2 exceptions that may be generated & write exceptional handler in Java.

```
package assignment3;

import java.util.Scanner;

public class question2 {


	public static void main(String[] args) {

String s[]=new String[5];

try

{

System.out.println("Input your favourite colour");

String m="pink";

for(int i=0;i<5;i++)
{
        Scanner sc=new Scanner(System.in);
        s[i]=sc.nextLine();
}
for(int i=0;i<5;i++)
{
        if(i>2)
        {
                throw new ArrayIndexOutofBoundsException();
        }
        if(m==null)
        {
                throw new NullpointerException();
        }
        else{
                System.out.println(colour is +m);
        }
}



}
catch(ArrayIndexoutofBoundsException a)
System.out.println("");
}
}
```

3. Create a class Student & enter mark, name of the student. If mark is more than 100, create exception MarksOutOfBoundException & throw it using Java.

```
import java.util.Scanner;
class MarksOutofBoundExceptionextendsException
{
        MarksOutofBoundsException(String str)
```

```
		{
	class student{
		int mark;
		String name;
		void input()
		{
			Scanner sc=new Scanner(System.in);
			System.out.println("Input name and mark");
			name=sc.nextLine();
			mark=sc.nextInt();
		try
		{
			if(mark>100)
				throw new MarksOut Of Bounds Exception("Not a valid mark");
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
		}


	public class a303 {

		public static void main(String[] args) {
			StudentE ob=new StudentE();
	ob.input();
		}

	}
```

4. Write a simple main class in Java that contains an experiment that uses the generic Box<T> class to build boxes with different types and that verifies that this class works as advertised.

```
class genericbox<T>
{
	T a, b;
	genericbox(T ae,T be)
	{
		a=ae;
		b=be;
	}
	void display()
	{
	System.out.println(a+" "+b);
	}

}
```

```
public class ass3prg4 {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
genericbox<String> obj=new genericbox<String>("hhh","kkkk");
obj.display();
genericbox<Integer> obj1=new genericbox<Integer>(56,34);
obj1.display();
genericbox<Object> obj2=new genericbox<Object>("kkf",45);
obj2.display();
        }

}
```

5. Write a java program to print an array of different type using a single Generic method. The signature of printArray method is given below.

```
public class ass3prg5
{
        public static<E> void printarray(E[] inputar)
        {
                for(E x:inputar)
                {
                        System.out.println(x);
                }
        }



        public static void main(String[] args)
        {
                printarray(1,2,3,4,5,43);

                printarray(1.3,2.3,4.5,6.5);
                // TODO Auto-generated method stub

        }





}
```

6. Write a java method using Generics to count the occurrence of an element in an array of any type.
7. Write a recursive method in Java that computes the factorial of a given integer.

```
import java.util.Scanner;
public class q7
{
	static int fact(int n)
	{
		if(n==0)
			return 1;
		else
			return n*fact(n-1);
	}



	public static void main(String args[])
	{
		Scanner in =new Scanner(System.in);
		System.out.println("enter n value");
		int a =in.nextInt();
		int k =fact(a);
		System.out.println(k);
	}
}
```

8. Write a recursive method in Java which, given real value x and a positive integer n, returns the value of xn.

```
    import java.util.Scanner;
public class prg8ass3 {
        static int pow(int x,int n)
        {
                if(n==0)
                        return 1;
                if(n==1)
                        return x;
                else
                        return x*pow(x,n-1);
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Scanner in =new Scanner(System.in);
                System.out.println("enter x&n value");
                int x =in.nextInt();
                int n =in.nextInt();
                int k =pow(x,n);
                System.out.println(k);
        }

}
```

9. Write a recursive method in Java which, given an integer n, print it with its digits reversed. For example , given 4735, it prints 5374.

```
class pa3

{

void rev(int n)

{

            if(n<10)

                        System.out.print(n);

            else

            {

                        System.out.print(n%10);

                rev(n/10);

}

}

}

public class ass3prg9

{

            public static void main(String[] args) {

                        // TODO Auto-generated method stub


pa3 ob1 =new pa3();

ob1.rev(136);

}


}
```

10. The sequence of numbers 1, 1, 2, 3, 5, 8, 13 etc are called Fibonacci numbers, each is the sum of the preceding two. Write a recursive method in Java which, given n, returns the nth Fibonacci number.

```
public class ass3prg10 {
        static int nfibo(int n)
        {
                if(n==0)
                        return 0;
                else if(n==1)
                        return 1;
                else
                        return nfibo(n-1)+nfibo(n-2);
        }
        public static void main(String args[])
        {
                int res =nfibo(3);
                System.out.println(res);
        }

}
```

11.Write a recursive method in Java to return the greatest common divisor(gcd) of two integers m and n, given that in general, gcd(m,n)=gcd(n, m mod n).

```
   class GCD
{
int gcd(int m,int n)
{
if(n==0)
return m;
else if(n>m)
return gcd(n,m);
else
return gcd(n,m%n);
}
public static void main(String[] args)
{

GCD obj=new GCD();
System.out.println("GCD of "+6+" and "+3+" is "+obj.gcd(3,6));
}
}
```

12. Write a recursive method in Java to search an element of an array using binary search

```
public class ass3prg12 {
	static boolean binsch(int a[],int l,int h,int key)
	{
		if(l>h)
			return false;
		else
		{
			int mid =(l+h)/2;
			if(key==a[mid])
			{
				return true;
			}
			else if(key<a[mid])
			{
				return binsch(a,l,mid-1,key);
			}
			else
			{
				return binsch(a,mid+1,h,key);
			}
		}
	}
	public static void main(String args[])
	{
		int a[] ={1,2,3,4,5,6,7};
		int l=0,h=a.length-1,key =4;
		boolean t =binsch(a,l,h,key);
		if(t==true)
		{
			System.out.println("sch successful");
		}
		else
		{
			System.out.println("sch unsuccess");

		}
	}

}
```

13. Write a recursive method in Java to find the binary equivalent of a positive decimal integer

```
import java.util.Scanner;
class Convert
{
	void binary(int n)
	{
		if(n>0)
		{
			binary(n/2);
```

```
                    System.out.print(n%2+" ");
                }
        }
}
public class assgn3q13
{
        public static void main(String[] args)
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter number");
                int n=sc.nextInt();
                Convert ob=new Convert();
                ob.binary(n);
        }
}
```

14. Write a recursive method in Java to find the product of 2 numbers

```
import java.util.Scanner;
class Product
{
        int Pro(int x, int y, int p)
        {
                if(y==0)
                        return p;
                else
                {
                        p=p+x;
                        return Pro(x,y-1,p);
                }
        }
}
public class assgn3q14
{
        public static void main(String[] args)
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter two numbers");
                int n=sc.nextInt();
                int x=sc.nextInt();
                Product ob=new Product();
                int z=ob.Pro(n,x,0);
                System.out.println(z);
        }
}
```

15. Write a recursive Java method that takes a character string s and outputs its reverse. For example, the reverse of 'pots&pans' would be 'snap&stop'

```
import java.util.Scanner;
class ReverseS
{
        String Revs(String s,int i, String x)
        {
                if(i<0)
                {
                        return x;
                }
                else
                {
                        x=x+s.charAt(i);
```

```
				return Revs(s,--i,x);
			}
		}
}
public class assgn3q15
{
		public static void main(String[] args)
		{
			Scanner sc=new Scanner(System.in);
			System.out.println("Enter a string");
			String n=sc.next();
			ReverseS ob=new ReverseS();
			String s=ob.Revs(n,n.length()-1,"");
			System.out.println(s);
		}
}
```

16. Write a recursive Java method that determines if a string s is a palindrome, that is, it is equal to its reverse. Examples of palindromes include 'racecar' and 'gohangasalamiimalasagnahog'.

```
import java.util.Scanner;
class ReverseSt
{
		String Revst(String s,int i, String x)
		{
			if(i>=0)
			{
				x=x+s.charAt(i);
				return Revst(s,--i,x);
			}
			return x;
```

```
        }
}
public class assgn3q16
{
        public static void main(String[] args)
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter a string");
                String n=sc.next();
                ReverseSt ob=new ReverseSt();
                String s=ob.Revst(n,n.length()-1,"");
                if(n.compareTo(s)==0)
                        System.out.println("Palindrome");
                else
                        System.out.println("Not Palindrome");
        }
}
```

17. Given an unsorted array, A, of integers and an integer k, write recursive program using Java for rearranging the elements in A so that all elements less than or equal to k come before any elements larger than k.

```
import java.util.*;
public class assgn3q17
{
  static void kpartition(int a[], int l, int u, int k)
  {
        if(l<u)
        {
                if(a[l]<k)
                        l++;
                if(a[u]>k)
                u--;
            if(a[l]==a[u])
                l++;
            else
            {
                int t=a[l];
                a[l]=a[u];
                a[u]=t;
            }
            kpartition(a,l,u,k);
        }
    }
    static void display(int a[])
    {
        for(int i=0;i<a.length;i++)
        {
            System.out.print(a[i]+" ");
        }
    }
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int a[]={1,4,5,7,3,9,5,8};
        System.out.println("Enter element to partition");
        int k=sc.nextInt();
        kpartition(a,0,a.length-1,k);
        display(a);
    }
}
```

18. In the Towers of Hanoi puzzle, we are given a platform with three pegs, a, b, and c, sticking out of it. On peg a is a stack of n disks, each larger than the next, so that the smallest is on the top and the largest is on the bottom. The puzzle is to move all the disks from peg a to peg c, moving one disk at a time, so that we never place a larger disk on top of a smaller one. See Figure 1 for an example of the case n = 4. Write a recursive program using Java for solving the Towers of Hanoi puzzle for arbitrary n. (Hint: Consider first the subproblem of moving all but the nth disk from peg a to another peg using the third as “temporary storage.”)

```
import java.util.*;
public class assgn3q18
{
    public static void main(String[] args)
    {
            Scanner sc=new Scanner(System.in);
            System.out.println("Enter number of rings");
            int n=sc.nextInt();
            char A='A',B='B',C='C';
            tower(n,A,B,C);
    }
    public static void tower(int n,char A,char B,char C)
    {
            if(n>0)
            {
                    tower(n-1,A,B,C);
                    System.out.println("Move "+A+"->"+B);
                    tower(n-1,C,B,A);
            }
    }
}
```

CSE 2001: Data Structure & Algorithms

Programming Assignment-IV

(Singly Linked List)

1. Write a menu driven Java Program using class, methods and reference variables, to construct a singly linked list consisting of the following information in each node: student regd_no (int), mark secured in a subject (float).
2. Define the methods for each of the following operations to be supported by the above linked list are:

a) The insertion operation

I. At the beginning of the list

Method Prototype: public static Node InsBeg (Node start)

ii. At the end of the list

Method Prototype: public static Node InsEnd (Node start)

iii. At any position in the list

Method Prototype: public static Node InsAny (Node start)

b) The deletion operation

I. From the beginning of the list

Method Prototype: public static Node DelBeg (Node start)

ii. From the end of the list

Method Prototype: public static Node DelEnd (Node start)

iii. From any position in the list

Method Prototype: public static Node DelAny (Node start)

iv. Deleting a node based on student regd_no. If the specified node is not present in the list an error message should be displayed. Both the option should be demonstrated.

c) Search a node based on student regd_no and update the mark of the student. If the specified node is not present in the list an error message should be displayed.

Method Prototype: public static void search (Node start)

d) Sort the nodes of the linked list according to the mark secured by the student from higher to lower.

Method Prototype: public static void sort (Node start)

e) Count the number of nodes present in the linked list

Method Prototype: public static int count (Node start)

f) Reverse the linked list

Method Prototype: public static Node reverse (Node start)

g) Displaying all the nodes in the list

The prototype of the display function should be as follows.

public static void display (Node start)

Program: -

```
import java.util.Scanner;
class Node
{
        int info;
        Node link;
        Node(){
                info=0;
                link=null;
        }
}
```

```
public class doublell{
        static Node start=null;
        public static void main(String[] args) {
                int choice;
                Scanner x=new Scanner(System.in);
                while(true)
                {
                        System.out.println("****MENU*****");
                        System.out.println("0 : Exit");
                        System.out.println("1 : Creation");
                        System.out.println("2 : Display");
                        System.out.println("3 : Insert at begining");
                        System.out.println("4 : Insert at end");
                        System.out.println("5 : Insert at specific postion");
                        System.out.println("6 : Delete at begining");
                        System.out.println("7 : Delete at end");
                        System.out.println("8 : Delete at position");
                        System.out.println("9 : Search");
                        System.out.println("--------------------------------------");
                        System.out.println("Enter the choice");
                        choice=x.nextInt();
                        switch(choice)
                        {
                                case 0:
```

```
        System.exit(0);
case 1:
        System.out.println("Enter size ");
        createLinkedList(x.nextInt());
        break;
case 2:
        displayLinkedList();
        break;
case 3:
        insertAtBegin();
        break;
case 4 :
        insertAtEnd();
        break;
case 5:
        InsAny();
        break;
case 6:
        DelBeg();
        break;
case 7:
        DelEnd();
        break;
case 8 :
```

```
                    DelAny();

                    break;

                case 9 :

                    search();

                    break;

                default:

                    System.out.println("Wrong choice");

                    break;

            }

        }


    }

    public static void displayLinkedList(){

        System.out.println("Displaying the node......");


        for (Node n1=start;n1!=null;n1=n1.link){

            //for (Node n1=start;n1.getLink()!=null;n1=n1.getLink()){

            System.out.println(n1.info);

        }

    }

    public static void insertAtBegin(){

        Scanner x=new Scanner(System.in);

        Node n1=start;

        Node n2=new Node();
```

```
		System.out.println("Enter value to insert at beginging ");

		int n=x.nextInt();

		n2.info=n;

		n2.link=n1;

		start=n2;

		//n1.setLink(n2);

	}

	public static void insertAtEnd(){

		Scanner x=new Scanner(System.in);

		Node n1=start;

		while (n1.link!=null){

			n1=n1.link;

		}

		Node n2=new Node();

		System.out.println("Enter value to insert at end ");

		int n=x.nextInt();

		n2.info=n;

		n1.link=n2;

		n2.link=null;

	}

	public static void search(){

		Scanner x=new Scanner(System.in);

		System.out.println("Enter value to search  ");

		int nn=x.nextInt();
```

```
            boolean found=false;
            int pos=1;
            Node n1=start;
            while (n1!=null){
                        if (nn==n1.info)
                        {
                                    found=true;
                                    break;
                        }
                        else
                                    pos++;
              n1=n1.link;
            }
            if (!found)
                        System.out.println("Element not found");
            else
                        System.out.println("Element found at "+pos+" position");
}
public static void createLinkedList(int n){
            Scanner x=new Scanner(System.in);
            System.out.println("Enter values  ");
            int nn=x.nextInt();
            Node n1=new Node();
            start=n1;
            n1.info=nn;

            for (int i=1;i<n;i++){

                        Node n2=new Node();

                        n2.info=x.nextInt();

                        n1.link=n2;

                        n1=n2;

            }

            /*for (Node n5=start;n5!=null;n5=n5.getLink()){

             System.out.println(n5.getInfo());

             }*/

}

public static void DelBeg(){

            Node n1=start;

            start=n1.link;

            n1=null;

            System.out.println("First node deleted successfully");

}

public static void DelEnd(){

   Node n1=start;

            while (n1.link.link!=null){

                        n1=n1.link;

            }

            System.out.println("Last node deleted successfuly");

            n1.link=null;
```

```
}

public static void  DelAny(){

        Scanner x=new Scanner(System.in);

        System.out.println("Enter position to delete at");

        int pos=x.nextInt();

        if (pos==1)

                DelBeg();

        else

        {

                Node n1=start;

                Node n2=new Node();

                for (int i=0;i<pos-2;i++){

                        n1=n1.link;

                }

                n1.link=n1.link.link;

                System.out.println("Node deleted successfully");

        }


}

public static void InsAny(){

        Scanner x=new Scanner(System.in);

        System.out.println("Enter position to insert at");

        int pos=x.nextInt();

        if (pos==1)
```

```
                        insertAtBegin();
                else{
                        Node n1=start;
                        Node n2=new Node();
                        for (int i=0;i<pos-2;i++){
                                n1=n1.link;
                        }
                        Node n3=new Node();
                        System.out.println("Enter data for new node");
                        n3.info=x.nextInt();
                        n3.link=n1.link;
                        n1.link=n3;
                }
        }
        }
```

CSE 2001: Data Structure & Algorithms Programming Assignment-V (Doubly Linked List)

1. Write a menu driven Java Program using class, methods and reference variables, to construct a **doubly linked list** consisting of the following information in each node: student regd_no (int), mark secured in a subject (float).
2. Define the methods for each of the following operations to be supported by the above

linked list are:

a) The insertion operation

I. At the beginning of the list

Method Prototype: **public static** `Node InsBeg (Node start, Node end)`

ii. At the end of the list

Method Prototype: **public static** `Node InsEnd (Node start, Node end)`

iii. At any position in the list

Method Prototype: **public static** `Node InsAny (Node start, Node end)`

b) The deletion operation

I. From the beginning of the list

Method Prototype: **public static** `Node DelBeg (Node start, Node end)`

ii. From the end of the list

Method Prototype: **public static** `Node DelEnd (Node start, Node end)`

iii. From any position in the list

Method Prototype: **public static** `Node DelAny (Node start, Node end)`

c) Search a node based on student regd_no and update the mark of the student. If the specified node is not present in the list an error message should be displayed.

Method Prototype: **public static void** `search (Node start)`

d) Displaying all the nodes in the list

The prototype of the display method should be as follows.

public static void `display (Node start, Node end)`

Program: -

```
import java.util.Scanner;
class Node
{
    int info;
    Node next;
    Node previous;
    Node(){
        info=0;
        next=previous=null;
    }
}
public class doublell{

    static Node start=null;static Node end;
    public static void main(String[] args) {
        int choice;
        Scanner x=new Scanner(System.in);
        while(true)
        {
            System.out.println("****MENU*****");
            System.out.println("0 : Exit");
            System.out.println("1 : Creation");
            System.out.println("2 : Display");
            System.out.println("3 : Insert at
begining");
            System.out.println("4 : Insert at end");
            System.out.println("5 : Insert at
specific postion");
            System.out.println("6 : Delete at
begining");
            System.out.println("7 : Delete at end");
            System.out.println("8 : Delete at
position");
            System.out.println("9 : Search");
            System.out.println("----------------);
            System.out.println("Enter the choice");
            choice=x.nextInt();
            switch(choice)
            {
                case 0:
                    System.exit(0);
                case 1:
                    System.out.println("Enter size
");
                    createLinkedList(x.nextInt());
                    break;
                case 2:
                    displayLinkedList();
                    break;
                case 3:
                    insertAtBegin();
                    break;
                case 4 :
                    insertAtEnd();
                    break;
                case 5:
                    InsAny();
                    break;
                case 6:
                    DelBeg();
                    break;
                case 7:
                    DelEnd();
                    break;
                case 8 :
                    DelAny();
                    break;
                case 9 :
                    search();
                    break;
                default:
                    System.out.println("Wrong
choice");
                    break;
            }
        }
    }
    public static void displayLinkedList(){
        System.out.println("Displaying the
node......");
```

```
        for (Node n1=start;n1!=null;n1=n1.next){
            //for (Node n1=start;n1.getLink()!
=null;n1=n1.getLink()){
            System.out.println(n1.info);
        }
    }
    public static void insertAtBegin(){
        Scanner x=new Scanner(System.in);
        Node n1=start;
        Node n2=new Node();
        System.out.println("Enter value to insert at
beginging ");
        int n=x.nextInt();
        n2.info=n;

        n2.next=n1;
        n1.previous=n2;
        start=n2;
        //n1.setLink(n2);
    }
    public static void insertAtEnd(){
        Scanner x=new Scanner(System.in);
        Node n1=end;
        Node n2=new Node();
        System.out.println("Enter value to insert at
end ");
        int n=x.nextInt();
        n2.info=n;
        end.next=n2;
        n2.previous=end;
        n2.next=null;
    }
    public static void search(){
        Scanner x=new Scanner(System.in);
        System.out.println("Enter value to search
");
        int nn=x.nextInt();
        boolean found=false;
        int pos=1;
        Node n1=start;
        while (n1!=null){
            if (nn==n1.info)
            {
                found=true;
```

```
                    break;
                }
                else
                    pos++;
              n1=n1.next;
            }
            if (!found)
                System.out.println("Element not found");
            else
                System.out.println("Element found at 
"+pos+" position");
        }
        public static void createLinkedList(int n){
            Scanner x=new Scanner(System.in);
            System.out.println("Enter values  ");
            int nn=x.nextInt();
            Node n1=new Node();
            start=end=n1;
            n1.info=nn;
            for (int i=1;i<n;i++){
                Node n2=new Node();
                n2.info=x.nextInt();
                end=n2;
                n1.next=n2;
                n2.previous=n1;
                n1=n2;
            }
            /*for (Node n5=start;n5!
=null;n5=n5.getLink()){
             System.out.println(n5.getInfo());
             }*/
        }
        public static void DelBeg(){
            Node n1=start;
            start=n1.next;
            n1.next.previous=null;
            n1=null;
            System.out.println("First node deleted 
successfully");
        }
        public static void DelEnd(){
            Node n1=end;
            end=n1.previous;
            n1.previous.next=null;
        System.out.println("Last node deleted 
successfuly");
        n1=null;
    }
    public static void  DelAny(){
        Scanner x=new Scanner(System.in);
        System.out.println("Enter position to delete 
at");
        int pos=x.nextInt();
        if (pos==1)
            DelBeg();
        else
        {
            Node n1=start;
            Node n2=new Node();
            for (int i=0;i<pos-2;i++){
                n1=n1.next;
            }
            n1.next.next.previous=n1;
            n1.next=n1.next.next;
            n1=null;

            System.out.println("Node deleted 
successfully");
        }
    }
    public static void InsAny(){
        Scanner x=new Scanner(System.in);
        System.out.println("Enter position to insert 
at");
        int pos=x.nextInt();
        if (pos==1)
            insertAtBegin();
        else{
            Node n1=start;
            Node n2=new Node();
            for (int i=0;i<pos-2;i++){
                n1=n1.next;
            }
            Node n3=new Node();
            System.out.println("Enter data for new 
node");
            n3.info=x.nextInt();
            n3.next=n1.next;
            n1.next.previous=n3;
            n3.previous=n1;
            n1.next=n3;
        }
    } }
```

CSE 2001: Data Structure & Algorithms Programming Assignment-VI (Stack)

Stack is an ordered set of elements in which insertion and deletion are from one end of the stack called the **top** of the stack. It is a Last in First Out structure (**LIFO**).

PART-I
Static Implementation (Array Implementation)

A stack is implemented statically by using an array of size **MAX** to hold stack elements and an integer **top** storing the position of top of the stack. The stack elements can be integers, characters, strings or user defined data types.

The operations to be performed on a stack are
public static int `push (`**int** `S [], `**int** `top)` – adding an element x to the stack S
public static int `pop (`**int** `S [], `**int** `top)`– deletes and returns the top element from the stack
public static void `display (`**int** `S [],`**int** `top)-` display all the elements of Stack S
public static boolean `is Empty (`**int** `top)` – check if the stack is empty
public static boolean `is Full (`**int** `top)` – check if the stack is full when top equals MAX -1
Write a menu driven Java Program using class, methods and array, to construct a **Stack and implement the above five operations.**

Program: -

```
import java.util.Scanner;
class StackCodes
{
int size;
int top=-1;
int arr[];
public StackCodes(int s)
{
size=s;
arr=new int[size];
}
public void push()
{
Scanner sc=new Scanner(System.in);
System.out.println("Enter element ");
int elements=sc.nextInt();
if(top==size-1)
System.out.println("Stack Overflow");
else
{
top++;
arr[top]=elements;
}
System.out.println("Elements entered"+arr[top]);
}
public void pop()
{
if(top==-1)
System.out.println("Stack Underflow");
else
{
int x=arr[top];
top--;
System.out.println("Elements deleted is"+x);
}
}
public void display()
{
if (top != 1)
{
for(int i=0;i<=top;i++)
System.out.println(arr[i]);
}
else
System.out.println("Stack is empty");
}
}
```

```
public class stack
{
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
int choice;
StackCodes sc1 = new StackCodes(50);
while (true)
{
System.out.println("0:Exit");
System.out.println("1:Push");
System.out.println("2:Pop");
System.out.println("3:Display");
System.out.println("Enter the Choice");
choice=sc.nextInt();
switch(choice)
{
case 0:
System.exit(0);
case 1:
sc1.push();
break;
case 2:
sc1.pop();
break;
case 3:
sc1.display();
break;
default :
System.out.println("Wrong Choice");
}
}
}
}
```

PART-II

Dynamic Implementation (Linked List Implementation)

A stack is implemented dynamically by using a Linked list where each node in the linked list has two parts, the data element and the reference to the next element of the stack.

The stack elements can be integers, characters, strings or user defined data types. There is no restriction on how big the stack can grow.

The operations to be performed on a stack are
public static `Node push (Node top)` - Add an element x to the stack S requires creation of node containing x and putting it in front of the top node pointed by S.
public static `Node pop (Node top)`- Delete the top node from the stack S so that next element becomes the top.
public static void `display (Node top)`- Display all the elements of Stack S.
Write a menu driven Java Program using class, methods and list, to construct a Stack and implement the above three operations.

Program: -

```
import java.util.Scanner;
class Node
{
int info;
Node link;
}
class stackc extends Node
{
Node top;
Node start;
public void push()
{
Scanner sc=new Scanner(System.in);
Node n1=new Node();
n1.info=sc.nextInt();
if(top==null)
{
top=start=n1;
}
else
{
top.link=n1;
top=n1;
}
}
void pop()
{
if (top==null)
System.out.println("Stack underflow");
else if(start==top)
{
int x=top.info;
start=top=null;
System.out.println("Element Deleted is "+x);
}
else
{
Node n1=start;
while(n1.link.link != null)
{
n1=n1.link;
}
int x=top.info;
top=n1;
n1.link=null;
System.out.println("Element deleted is "+x);
}
}
void display()
{
Node n1=start;
while(n1 != null)
{
System.out.println(n1.info);
n1=n1.link;
}
}
void count_number_of_elements()
{
int count=0;
Node n1=start;
while(n1!=null)
{
count++;
n1=n1.link;
}
System.out.println("Size is"+count);
}
}

public class StackLl {
```

```
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
int choice;
stackc skc=new stackc();
while (true)
{
System.out.println("0:Exit");
System.out.println("1:Push");
System.out.println("2:Pop");
System.out.println("3:Display");
System.out.println("4:Count no.of elements");
System.out.println("Enter the Choice");
choice=sc.nextInt();
switch(choice)
{
case 0:
System.exit(0);
case 1:
skc.push();
break;
case 2:
skc.pop();
break;
case 3:
skc.display();
break;
case 4:
skc.count_number_of_elements();
break;
default :
System.out.println("Wrong Choice");
}
}
}
}
```

CSE 2001: Data Structure & Algorithms Programming Assignment-VII (Queue)

Queue is an ordered set of elements in which insertions are from the **rear** and deletions are from the **front**. It is a First in First Out structure (**FIFO**).

PART-I
Static Implementation (Array Implementation)

A Queue is implemented statically by using an array of size **MAX** to hold the elements and it has two ends (integers) – **front** and **rear**. The '**front**' stores the position of the current front element and '**rear**' stores the position of the current rear element of the queue. The Queue elements can be integers, characters, strings or user defined data types.

The operations to be performed on a Queue are

public static void `insert (`**int** `Q [])`-adding an element x to the rear end of the queue Q
public static void `delete (`**int** `Q [])`-deletes the element from the front of the queue Q
public static void `display (`**int** `Q [])`-display all the elements of the queue Q.
public static boolean `is_full ()`-check if the queue is full or not.
public static boolean `is_empty ()`-check if the queue is empty or not.

Write a menu driven Java Program using class, methods and array, to construct a **Queue and implement the above five operations.**

Program

```
import java.util.Scanner;
class queue
{
     int q[];
     int front=-1;
     int rear=-1;
```

```
	int size;
queue(int s)
{
	size=s;
	q=new int[size];
}
void insert()
{
	if(rear==size-1)
	{
		System.out.println("Queue overflow");
	}
	else
	{
		Scanner sc=new Scanner(System.in);
		int x=sc.nextInt();

		if(rear==-1)
		{
			front=rear=0;
			q[rear]=x;
		}
		else
		{
			rear++;
			q[rear]=x;
		}
	}
}

void delete()
{
	int x;
	if(front==-1)
	{
	System.out.println("Queue underflow");
	}
	else
	{
		if (front==rear)
		{
			x=q[front];
			front=rear=-1;
		}
		else
		{
			x=q[front];
			front++;
		}
		System.out.println("Element deleted is "+x);
	}
}

void display()
{
     if(front==-1)
          System.out.println("Queue is empty");
     else
     {
          for(int i=front;i<=rear;i++)
               System.out.println(q[i]);
     }
}

public static class Queue_Arr
{

     public static void main (String[]args)
     {
          Scanner sc=new Scanner(System.in);
          int choice;
          queue qu=new queue(5);
          while (true)
          {
               System.out.println("0:Exit");
               System.out.println("1:Insert");
               System.out.println("2:Delete");
               System.out.println("3:Display");
               System.out.println("Enter the Choice");
               choice=sc.nextInt();

               switch(choice)
               {
               case 0:
                    System.exit(0);
               case 1:
                    qu.insert();
                    break;
               case 2:
                    qu.delete();
                    break;
               case 3:
                    qu.display();
                    break;
               default :
                    System.out.println("Wrong Choice");
               }
          }

     }

}
```

```
}
```

PART-II
Dynamic Implementation (Linked List Implementation)

A *Queue* is implemented dynamically by using a Linked list where each node in the linked list has two parts, the data element and the reference to the next element of the queue.
The Queue elements can be integers, characters, strings or user defined types. There is no restriction on how big the Queue can grow.

The operations to be performed on a Queue:
public static `Node insert (Node rear, Node front)` - adding an element x to the queue Q requires creation of node containing x and putting it next to the rear and rear points to the newly added element.
public static `Node delete (Node rear, Node front)` - deletes the front node from the queue Q
public static void `display (Node rear, Node front)-display all the elements of the queue Q.`

Write a menu driven Java Program using class, methods and list, to construct a **Queue and implement the above three operations.**

Program

```
import java.util.Scanner;
class Nodeq
{
Nodeq link;
int info;
}
class Queue
{
Nodeq front;
Nodeq rear;
int size;
void insert()
{
Scanner sc=new Scanner(System.in);
Nodeq nq=new Nodeq();
if(front==null)
{
nq.info=sc.nextInt();
front=rear=nq;
}
else
{
Node n1=new Node();
```

```
n1.info=sc.nextInt();
rear.link=nq;
rear=nq;
}
}
void delete()
{
if(front==null)
System.out.println("Queue Underflow");
else
{
if(front==rear)
{
front=rear=null;
}
else
{
front=front.link;
}}
}
void display()
{
Nodeq nq=front;
while(nq != null)
{
System.out.println("n1.info");
nq=nq.link;
}}
}
public class Queue_Ll
{
public static void main(String[] args)
{
Queue q1=new Queue();
Scanner sc=new Scanner(System.in);
int choice;
while (true)
{
System.out.println("0:Exit");
System.out.println("1:Insert");
System.out.println("2:Delete");
System.out.println("3:Display");
System.out.println("Enter the Choice");
choice=sc.nextInt();
switch(choice)
{
case 0:
System.exit(0);
case 1:
q1.insert();
break;
case 2:
q1.delete();
break;
case 3:
q1.display();
break;
default :
System.out.println("Wrong Choice");
}}
}}
```