UPDS - 2016

TeamBook

Bolivia - Tarija

Contenido

MATEMATICAS

Torre De Hanoi

$$\text{Movimientos } = 2^{fichas} - 1$$

Dos elevado a la cantidad de fichas menos uno

Distancia Entre Dos Puntos En El Espacio

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} = d$$

```
Dis=Math.sqrt(Math.pow(dx2-dx1,2)+Math.pow(dy2-dy1,2)+Math.pow(dz2-dz1,2));
```

Criba De Erastotenes

```
	public static boolean[] criba(int n)
	{
		boolean primos[] = new boolean[n+1];
		Arrays.fill(primos,true);
		primos[0] = primos[1] = false;
		for(int i=2;i<(int)Math.sqrt(n)+1;i++)
		 if(primos[i])
		  for(int j=i*i;j<primos.length;j+=i)
		   primos[j] = false;
		return primos;
	}
public static void main(String[] args)
{
		Scanner in = new Scanner(System.in);
		boolean primos[] = new boolean[n+1];
		primos = criba(n);
}
```

Funcion Primos – Biginteger-Java

```
	BigInteger b = in.nextBigInteger();
		b.isProbablePrime(100); //booleano
```

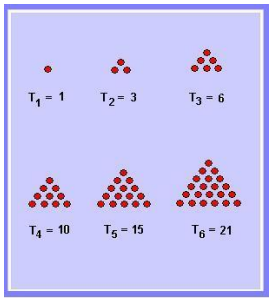
Triangulo De Pascal

```
		int a[]= new int[1];
		for (int i = 0; i <= 17; i++)
		{
			int x[]=new int[i];
			for (int j = 0; j < i; j++)
			{
				if (j==0 || j==(i-1))
					x[j]=1;
				else
					x[j]= a[j]+a[j-1];
				System.out.print(x[j]+" ");

			}
			a=x;
			System.out.println();
```

Numeros triangulares

$$T_n = \frac{n(n+1)}{2}$$



Raíces triangulares

Por analogía con la raíz cuadrada de x , se puede definir la raíz triangular (positivo) de x como el número n de tal manera que $T_n = x$:

$$n = \frac{\sqrt{8x+1}-1}{2}$$

```
long n =(long) ((Math.sqrt(1+8*x)-1)/2);
```

Area of the Koch Snowflake

```
int s = in.nextInt();
double area = (2 * Math.sqrt(3) * (s*s))/5;
System.out.printf("%.2f\n",area);
```

Numeros Consecutivos

nf=número final
ni=número inicial
=(nf-ni+1)*(nf+ni)/2

Gaussian Elimination (*Cp3 –Halim Pag 346*)

```
class AugmentedMatrix {
  public double[][] mat = new double[3][4];  // adjust this value as needed
  public AugmentedMatrix() {};
}
class ColumnVector {
  public double[] vec = new double[3];        // adjust this value as needed
  public ColumnVector() {};
}
class GaussianElimination {
  public static ColumnVector GE(int N, AugmentedMatrix Aug) {
    // input: N, Augmented Matrix Aug, output: Column vector X, the answer
    int i, j, k, l; double t;

    for (i = 0; i < N - 1; i++) {          // the forward elimination phase
      l = i;
      for (j = i + 1; j < N; j++)     // which row has largest column value
        if (Math.abs(Aug.mat[j][i]) > Math.abs(Aug.mat[l][i]))
          l = j;                                    // remember this row l
```

```
      // swap this pivot row, reason: minimize floating point error
      for (k = i; k <= N; k++) {        // t is a temporary double variable
        t = Aug.mat[i][k];
        Aug.mat[i][k] = Aug.mat[l][k];
        Aug.mat[l][k] = t;
      }
      for (j = i + 1; j < N; j++)   // the actual forward elimination phase
        for (k = N; k >= i; k--)
          Aug.mat[j][k] -= Aug.mat[i][k] * Aug.mat[j][i] / Aug.mat[i][i];
    }

    ColumnVector Ans = new ColumnVector();   // the back substitution phase
    for (j = N - 1; j >= 0; j--) {                        // start from back
      for (t = 0.0, k = j + 1; k < N; k++) t += Aug.mat[j][k] * Ans.vec[k];
      Ans.vec[j] = (Aug.mat[j][N] - t) / Aug.mat[j][j]; // the answer is here
    }
    return Ans;
  }

  public static void main(String[] args) {
    AugmentedMatrix Aug = new AugmentedMatrix();
    Aug.mat[0][0] = 1; Aug.mat[0][1] = 1; Aug.mat[0][2] = 2; Aug.mat[0][3] = 9;
    Aug.mat[1][0] = 2; Aug.mat[1][1] = 4; Aug.mat[1][2] = -3; Aug.mat[1][3] = 1;
    Aug.mat[2][0] = 3; Aug.mat[2][1] = 6; Aug.mat[2][2] = -5; Aug.mat[2][3] = 0;
    ColumnVector X = GE(3, Aug);
    System.out.printf("X = %.1f, Y = %.1f, Z = %.1f\n", X.vec[0], X.vec[1], X.vec[2]);
  }
}
```

Numeros Catalanes (*Cp3 –Halim Pag 205*)

```
public class Catalan{
   public static BigInteger[] vector = new BigInteger[6000];
   public static void coeficienteBinomial(){

            vector[0]=BigInteger.ONE;
        vector[1]=BigInteger.ONE;

        for(int i=2; i<=5100 ;i++) //GENERADO HASTA 5100
        {
            BigInteger numerador=BigInteger.valueOf(2*i*(2*i-1));
            BigInteger denominador=BigInteger.valueOf(i*(i+1));
            vector[i]=vector[i-1].multiply(numerador).divide(denominador);
        }
    }

    public static void main(String[] args) {
       Scanner in = new Scanner(System.in);
        int n=in.nextInt();
        coeficienteBinomial();
        System.out.println(vector[n+1]);
    }
}
```

Cuadrado Magico

Uva magicSquare1266

```
public class magicSquare1266 {
	public static void main(String[] args) {
		Scanner in = new Scanner(System.in);
		int uno=0;
		while (in.hasNext())
		{
			 if(uno==1) System.out.println();
			 uno = 1;
			int n = in.nextInt();
			System.out.printf("n = %d, sum = %d\n", n, n*(n*n+1)/2);
		  int matriz [][]= new int [15][15], x = 0, y = n/2;
		  for(int i = 1; i <= n*n; i++)
		  {
		      if(matriz[x][y]!=0)
		      {
		          x += 2; y--;
		          if(x >= n)    x -= n;
		          if(y < 0)     y += n;
		          matriz[x][y] = i;
		      } else
		          matriz[x][y] = i;
		      x--; y++;
		      if(x < 0)    x += n;
		      if(y >= n)    y -= n;
		  }
		  for(int i = 0; i < n; i++)
		  {
		      for(int j = 0; j < n; j++)
		          System.out.printf(" "+matriz[i][j]);
		      System.out.println();
} }}}
input  3
output n = 3, sum = 15
               8  1  6
               3  5  7
               4  9  2
```

Invertir Numeros

```
public static int reverse(int x)
        {
                int i=10, y=0, sum=0;
                do
                {
                        y=x%10;
                        x=x/10;
                        sum=sum*i+y;
                }while(x!=0);
                return sum;
        }
```

Redondeo

```
System.out.println(Math.rint(d*1000)/1000);  //redondea a 3 decimales hacienda por 1000

System.out.printf("%.3f\n",consumo); // redondea a 3 decimales
```

Math.ceil(2.8) vale 3 (redondea para arriba)

Math.floor(2.8) vale 2 (redondea para abajo)

Math.round(2.8) vale 3 (la forma clásica)

Euclides - GCD

1.

```
public static int Euclides(int a, int b) {
                    int r=b;
                    while (b> 0){
                    r=a%b;
                    a=b;
                    b=r;
                    }
                    return (a);
      }
```

2. //ligera

```
 public static long gcd(long a, long b)
 {
   return b == 0 ? a : gcd(b, a % b);
 } // standard gcd
```

3. //gcd biginteger

```
BigInteger a= in.nextBigInteger();
BigInteger b = in.nextBigInteger();
a =a.gcd(b);
```

Least Common Multiple - LCM

```
public static long gcd(long a, long b) {
    return b == 0 ? Math.abs(a) : gcd(b, a % b);
  }

  public static long lcm(long a, long b) {
    return Math.abs(a / gcd(a, b) * b);
  }
```

Fibonacci Iterativo

```
public static int fiboIterativo(int n)
{
	if(n <= 2){
		if(n==0)
			return 1;
		else
	return n;
	}
	int res = 0;
	int act = 1;
	int ant = 0;
	int i = 0;
	while( i < n )//n-1
	{
	int temp;
	res = act + ant;
	temp = act;
	act = res;
	ant = temp;
	i++;
	}
	return res;
}
```

Fibonacci Recursivo (Sin tomar en cuenta n=0)

```
public static int Fibonacci (int n)
	{
		if (n <= 2)
			return n;
		else
			return Fibonacci(n-1) + Fibonacci(n-2);
	}
```

Algoritmo Para Convertir Un Numero Decimal A Binario

1.

```
 int numero, exp, digito;
double binario;
Scanner sc = new Scanner(System.in);
numero = sc.nextInt();
exp=0;
binario=0;
while(numero!=0)
{
	digito = numero % 2;
	binario = binario + digito * Math.pow(10, exp);
	exp++;
	numero = numero/2;
}
System.out.printf("%.0f", binario);
```

2.

```
int n = in.nextInt(); System.out.println(Integer.toBinaryString(n));
```

Convertir una Fraccion a Fraccion Mixta

```
public static String func(int n, int d ) {
    int a = n / d;
    int b = n %d;
    if (a!=0)
             return (a + " " + b + " / " + d);
    else
       return  (a + " " +b + " / " + d);
}
```

Biginteger

Declarar la clase biginteger

```
BigInteger aux = null;
BigInteger entero2= new BigInteger(in.next());
BigInteger entero3= BigInteger.valueOf(1);
```

Mod

```
aux.mod(new BigInteger("2")).equals(BigInteger.ZERO)
```

Operaciones

```
z = x.add(y);
z = x.subtract(y);
z = x.multiply(y);

z = x.divide(y);

aux = aux.divide(new BigInteger("2"));

z = x.add(y).divide(new BigInteger("2")); // (z=x+y/2)
```

Por defecto en java

```
BigInteger.ONE
BigInteger.ZERO
BigInteger.TEN
```

SQTR

```
public static BigInteger sqrt(BigInteger x) {
    BigInteger div = BigInteger.ZERO.setBit(x.bitLength()/2);
    BigInteger div2 = div;
    // Loop until we hit the same value twice in a row, or wind
    // up alternating.
    for(;;) {
        BigInteger y = div.add(x.divide(div)).shiftRight(1);
        if (y.equals(div) || y.equals(div2))
            return y;
        div2 = div;
        div = y;
    }
}
```

Desviacion Estandar

En matemáticas, la desviación estándar de un conjunto de *n* números enteros se define como:

$$S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}}$$

donde $\overline{x}$ es la media del conjunto de *n* números enteros para el que se calcula la desviación estándar. Esta media se calcula como:

$$\overline{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

```
double n2 = n*2; //introducir n
double x_sq = 0, x, avg;
x_sq = n2 * (n2+1) * (2*n2+1) / 6 - 4 * (n) * (n+1) * (2*n+1) / 6;
x = n2 * (n2+1) / 2 - 2 * (n) * (n+1) / 2;
avg = x * 1.f / n;
double s = Math.sqrt((x_sq - 2 * avg * x + avg*avg * n) * 1.f/ (n-1));
```

Maximo Y Minimo

```
int maximo=0;
maximo=Math.max(maximo,x);
int minimo=0;
minimo=Math.min(minimo,x);
```

GEOMETRIA COMPUTACIONAL

```
public class Geom3D
{
// DISTANCE FROM POINT (X,Y,Z) TO PLANE AX + BY + CZ + D =0
      public static double ptPlaneDist(double x, double y, double z,
      double a, double b, double c, double d) {
      return Math.abs(a*x + b*y + c*z + d) / Math.sqrt(a*a + b*b + c*c);
}
// DISTANCE BETWEEN PARALLEL PLANES AX + BY + CZ + D1 = 0 AND
// AX + BY + CZ + D2 = 0
public static double planePlaneDist(double a, double b, double c,
double d1, double d2)
{
      return Math.abs(d1 - d2) / Math.sqrt(a*a + b*b + c*c);
}
// DISTANCE FROM POINT (PX, PY, PZ) TO LINE (X1, Y1, Z1)-(X2, Y2, Z2)
// (OR RAY, OR SEGMENT; IN THE CASE OF THE RAY, THE ENDPOINT IS THE
// FIRST POINT)
public static final int LINE = 0;
public static final int SEGMENT = 1;
public static final int RAY = 2;
public static double ptLineDistSq(double x1, double y1, double z1,
double x2, double y2, double z2, double px, double py, double pz,
int type)
{
      double pd2 = (x1-x2)*(x1-x2) + (y1-y2)*(y1-y2) + (z1-z2)*(z1-z2);
      double x, y, z;
      if (pd2 == 0)
      {
            x = x1;
            y = y1;
            z = z1;
      }
      else
      {
            double u = ((px-x1)*(x2-x1) + (py-y1)*(y2-y1) + (pz-z1)*(z2-z1)) / pd2;
            x = x1 + u * (x2 - x1);
            y = y1 + u * (y2 - y1);
            z = z1 + u * (z2 - z1);
            if (type != LINE && u < 0)
            {
                  x = x1;
                  y = y1;
                  z = z1;
            }
            if (type == SEGMENT && u > 1.0)
            {
                  x = x2;
                  y = y2;
                  z = z2;
            }
      }
      return (x-px)*(x-px) + (y-py)*(y-py) + (z-pz)*(z-pz);
}

public static double ptLineDist(double x1, double y1, double z1,
double x2, double y2, double z2, double px, double py, double pz,
int type)
{
      return Math.sqrt(ptLineDistSq(x1, y1, z1, x2, y2, z2, px, py, pz, type));
}}
```

VECTORES[]

Binarysearch - Java

Permite buscar un elemento de forma ultrarrápida en un array ordenado (en un array desordenado sus resultados son impredecibles).

```
Int x[] = {
1//[0],2//[1],3//[2],4//[3],5//[4],6//[5],7//[6],8//[7],9//[8],10//[9],11//[10],12/*[11]*/};
            System.out.println(Arrays.binarySearch(x,in.nextInt()));
            //la impresion entrega la posicion del numero ingresado
```

Pasar De String A Vector Char [ ]

```
String abc= "abcd";
      char[] vector;
      vector = abc.toCharArray(); //pasa lo del string a un vector char
```

Método Ordenamiento Sort

vector

```
int[]vector = new int[5];
Arrays.sort(vector);
```

arraylist

```
ArrayList<Integer>lista = new ArrayList<Integer>();
Collections.sort(lista);
```

Busqueda Binaria - Algoritmo

```
Public static int busquedaBinaria(int a[],int clave)
{
        int central, bajo, alto;

        int valorCentral;
        bajo = 0;
        alto = a.length - 1;
        while (bajo <= alto)
        {
                central = (bajo + alto)/2;          // índice de elemento central
                valorCentral = a[central];          // valor del índice central
                if (clave == valorCentral)
                        return central;             // encontrado, devuelve posición
                else if (clave < valorCentral)
                        alto = central -1;          // ir a sublista inferior
                else
                        bajo = central + 1;         // ir a sublista superior
        }
        return -1;          //elemento no encontrado
}
```

Busqueda Binaria (El Más Cercano)

```
public static int n ;
public static int A[]= {1,3,4,9,47};
public static int binarySearchProximo(int inicio, int fin, int valor)
{
	int mid, i;
	while(inicio <= fin)
	{
		if(A[inicio] > valor)
			return inicio;
		if(A[fin] <= valor)
			return fin+1;
		mid = (inicio + fin) / 2;
		if(A[mid] > valor)
			fin = mid - 1;
		else if(A[mid] < valor)
			inicio = mid + 1;
		else if(A[mid] == valor)
		{
			for(i=mid+1; i<n; i++)
				if(valor != A[i])
					return i;
			return i;
		}
	}
	return n;
}
public static void main(String[] args) {
	int n = A.length;
	 System.out.println(binarySearchProximo(0, A.length-1, 2));
}
```

STRING

Abecedario En Un For

```
public static int numeroAlfabeto(Character palabra)
{
        int i=0;
          for (char letra = 'A'; letra <= 'Z'; letra++)
          {
                i++;
                if (palabra == letra)
                     break;
          }
        return i;
}
```

Expresiones Regulares

letras y numerous convierte a “W” – operadores aritméticos convierte a “*”

```
patron = patron.replaceAll("\\w+","W").replaceAll("['+','/',-]","*");
```

El siguiente es un ejemplo del uso del método replaceAll sobre una cadena. El ejemplo sustituye todas las apariciones que concuerden con el patron "a*b" por la cadena "-".

```
Pattern patron = Pattern.compile("a*b");
// creamos el Matcher a partir del patron, la cadena como parametro
Matcher encaja = patron.matcher("aabmanoloaabmanoloabmanolob");
// invocamos el metodo replaceAll
String resultado = encaja.replaceAll("-");
System.out.println(resultado);
//input  aabmanoloaabmanoloabmanolob
//output -manolo-manolo-manolo-
```

Comprobar si el String *cadena* contiene “abc”

```
String cadena=en.next();
             Pattern pat = Pattern.compile(".*abc.*");
            Matcher mat = pat.matcher(cadena);
            if (mat.matches()) {
                System.out.println("SI");
            } else {
                System.out.println("NO");
            }
```

Comprobar si el String *cadena* empieza por “abc”

```
Pattern pat = Pattern.compile("^abc.*");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {
    System.out.println("Válido");
} else {
    System.out.println("No Válido");
      }
```

Comprobar si el String *cadena* empieza por “abc” ó “Abc”

```
Pattern pat = Pattern.compile("^[aA]bc.*");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {
```

```
        System.out.println("SI");
    } else {
        System.out.println("NO");
          }
```

Comprobar si el String *cadena* no empieza por un dígito

```
Pattern pat = Pattern.compile("^[^\\d].*");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {
    System.out.println("SI");
} else {
    System.out.println("NO");
      }
```

Comprobar si el String cadena no acaba con un dígito

```
Pattern pat = Pattern.compile(".*[^\\d]$");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {
    System.out.println("SI");
} else {
    System.out.println("NO");
}
```

Comprobar si el String cadena contiene un 1 y ese 1 no está seguido por un 2

```
Pattern pat = Pattern.compile(".*1(?!2).*");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {
    System.out.println("SI");
} else {
    System.out.println("NO");
      }
```

String copyValueOf() Method

//pasar de un vector de caracteres a un string

Example:

```
public class Test {

   public static void main(String args[]) {
      char[] Str1 = {'h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd'};
      String Str2 = "";

      Str2 = Str2.copyValueOf( Str1 );
      System.out.println("Returned String: " + Str2);

      Str2 = Str2.copyValueOf( Str1, 2, 6 );
      System.out.println("Returned String: " + Str2);
   }
}
```

This produces the following result:

```
Returned String: hello world
Returned String: llo wo
```

String endsWith() Method

Example:

```
public class Test{

   public static void main(String args[]){
      String Str = new String("This is really not immutable!!");
      boolean retVal;

      retVal = Str.endsWith( "immutable!!" );
      System.out.println("Returned Value = " + retVal );

      retVal = Str.endsWith( "immu" );
      System.out.println("Returned Value = " + retVal );
   }
}
```

This produces the following result:

```
Returned Value = true
Returned Value = false
```

String indexOf() Method

Example:

```
import java.io.*;

public class Test {

   public static void main(String args[]) {
      String Str = new String("Welcome to Tutorialspoint.com");
      String SubStr1 = new String("Tutorials");
      String SubStr2 = new String("Sutorials");

      System.out.print("Found Index :" );
      System.out.println(Str.indexOf( 'o' ));
      System.out.print("Found Index :" );
      System.out.println(Str.indexOf( 'o', 5 ));
      System.out.print("Found Index :" );
      System.out.println( Str.indexOf( SubStr1 ));
      System.out.print("Found Index :" );
      System.out.println( Str.indexOf( SubStr1, 15 ));
      System.out.print("Found Index :" );
      System.out.println(Str.indexOf( SubStr2 ));
   }
}
```

This produces the following result:

```
Found Index :4
Found Index :9
Found Index :11
Found Index :-1
Found Index :-1
```

String valueOf() Method

Example:

```
import java.io.*;

public class Test{
   public static void main(String args[]){
      double d = 102939939.939;
      boolean b = true;
      long l = 1232874;
      char[] arr = {'a', 'b', 'c', 'd', 'e', 'f','g' };

      System.out.println("Return Value : " + String.valueOf(d) );
      System.out.println("Return Value : " + String.valueOf(b) );
      System.out.println("Return Value : " + String.valueOf(l) );
      System.out.println("Return Value : " + String.valueOf(arr) );
   }
}
```

This produces the following result:

```
Return Value : 1.02939939939E8
Return Value : true
Return Value : 1232874
Return Value : abcdefg
```

String lastIndexOf() Method

Example:

```
import java.io.*;

public class Test {

   public static void main(String args[]) {
      String Str = new String("Welcome to Tutorialspoint.com");
      String SubStr1 = new String("Tutorials" );
      String SubStr2 = new String("Sutorials" );

      System.out.print("Found Last Index :" );
      System.out.println(Str.lastIndexOf( 'o' ));
      System.out.print("Found Last Index :" );
      System.out.println(Str.lastIndexOf( 'o', 5 ));
      System.out.print("Found Last Index :" );
      System.out.println( Str.lastIndexOf( SubStr1 ));
      System.out.print("Found Last Index :" );
      System.out.println( Str.lastIndexOf( SubStr1, 15 ));
      System.out.print("Found Last Index :" );
      System.out.println(Str.lastIndexOf( SubStr2 ));
   }
}
```

This produces the following result:

```
Found Last Index :27
Found Last Index :4
Found Last Index :11
Found Last Index :11
Found Last Index :-1
```

String replace() Method

Example:

```
import java.io.*;

public class Test{
   public static void main(String args[]){
      String Str = new String("Welcome to Tutorialspoint.com");

      System.out.print("Return Value :" );
      System.out.println(Str.replace('o', 'T'));

      System.out.print("Return Value :" );
      System.out.println(Str.replace('l', 'D'));
   }
}
```

This produces the following result:

```
Return Value :WelcTme tT TutTrialspTint.cTm
Return Value :WeDcome to TutoriaDspoint.com
```

String replaceAll() Method

Example:

```
import java.io.*;

public class Test{
   public static void main(String args[]){
      String Str = new String("Welcome to Tutorialspoint.com");

      System.out.print("Return Value :" );
      System.out.println(Str.replaceAll("(.*)Tutorials(.*)",
                         "AMROOD" ));
   }
}
```

This produces the following result:

```
Return Value :AMROOD
```

String replaceFirst() Method

Example:

```
import java.io.*;

public class Test{
   public static void main(String args[]){
      String Str = new String("Welcome to Tutorialspoint.com");

      System.out.print("Return Value :" );
      System.out.println(Str.replaceFirst("(.*)Tutorials(.*)",
                         "AMROOD" ));

      System.out.print("Return Value :" );
      System.out.println(Str.replaceFirst("Tutorials", "AMROOD" ));
   }
}
```

This produces the following result:

```
Return Value :AMROOD
Return Value :Welcome to AMROODpoint.com
```

String split() Method

Example:

```
import java.io.*;

public class Test{
   public static void main(String args[]){
      String Str = new String("Welcome-to-Tutorialspoint.com");

      System.out.println("Return Value :" );
      for (String retval: Str.split("-", 2)){
         System.out.println(retval);
      }
      System.out.println("");
      System.out.println("Return Value :" );
      for (String retval: Str.split("-", 3)){
         System.out.println(retval);
      }
      System.out.println("");
      System.out.println("Return Value :" );
      for (String retval: Str.split("-", 0)){
         System.out.println(retval);
      }
      System.out.println("");
      System.out.println("Return Value :" );
      for (String retval: Str.split("-")){
         System.out.println(retval);
      }
   }
}
```

This produces the following result:

```
Return Value :
Welcome
to-Tutorialspoint.com

Return Value :
Welcome
to
Tutorialspoint.com

Return Value:
Welcome
to
Tutorialspoint.com

Return Value :
Welcome
to
Tutorialspoint.com
```

String startsWith() Method

Example:

```
import java.io.*;

public class Test{
   public static void main(String args[]){
      String Str = new String("Welcome to Tutorialspoint.com");

      System.out.print("Return Value :" );
      System.out.println(Str.startsWith("Welcome") );

      System.out.print("Return Value :" );
      System.out.println(Str.startsWith("Tutorials") );

      System.out.print("Return Value :" );
      System.out.println(Str.startsWith("Tutorials", 11) );
   }
}
```

This produces the following result:

```
Return Value :true
Return Value :false
Return Value :true
```

String subSequence() Method

Example:

```
import java.io.*;

public class Test{
   public static void main(String args[]){
      String Str = new String("Welcome to Tutorialspoint.com");

      System.out.print("Return Value :" );
      System.out.println(Str.subSequence(0, 10) );

      System.out.print("Return Value :" );
      System.out.println(Str.subSequence(10, 15) );
   }
}
```

This produces the following result:

```
Return Value :Welcome to
Return Value : Tuto
```

String substring() Method

Example:

```
import java.io.*;

public class Test{
   public static void main(String args[]){
      String Str = new String("Welcome to Tutorialspoint.com");

      System.out.print("Return Value :" );
      System.out.println(Str.substring(10) );

      System.out.print("Return Value :" );
      System.out.println(Str.substring(10, 15) );
   }
}
```

This produces the following result:

```
Return Value : Tutorialspoint.com
Return Value : Tuto
```

String trim() Method

Example:

```
import java.io.*;

public class Test{
   public static void main(String args[]){
      String Str = new String("   Welcome to Tutorialspoint.com   ");

      System.out.print("Return Value :" );
      System.out.println(Str.trim() );
   }
}
```

This produces the following result:

```
Return Value :Welcome to Tutorialspoint.com
```

kmp

```
public class KMP {
       static char[] T, P;
       static int n, m;
       static int [] b;
       static void kmpPreprocess()
         {
           int i = 0, j = -1; b[0] = -1;
           while (i < m)
           {
             while (j >= 0 && P[i] != P[j])
               j = b[j];
             i++; j++;
             b[i] = j;
           }
          }
         static int kmpSearch()
         {
           int i = 0, j = 0,cont=0;
           while (i < n)
           {
              while (j >= 0 && T[i] != P[j])
                j = b[j];
              i++; j++;
              if (j == m)
              {
                 cont++;
                j = b[j];
              }
             }
           return cont;
          }
       public static void main(String[] args) {
              Scanner in = new Scanner(System.in);
              String cad1 = in.next();
              String cad2 = in.next();
              T  = cad1.toCharArray();
              P  = cad2.toCharArray();
           n = T.length;
           m = P.length;
```

```
            b = new int[400001];
            kmpPreprocess();
               int contador = kmpSearch();
            System.out.println(contador);
        }
}
```

Pattern y Matcher

```
//busca cuantas veces aparece la palabra coderoad

            Pattern p = Pattern.compile("CodeRoad");
            Matcher m = p.matcher(cadena);
             while (m.find())
                  cont++;
```

Longest Common Substring

```
public class LongestCommonSubstring {
      public static int longestComSubstr(String s, String t) { //algoritmo 1
            if (s.isEmpty() || t.isEmpty())
                  return 0;
            int m = s.length();
            int n = t.length();
            int cost = 0;
            int maxLen = 0;
            int[] p = new int[n];
            int[] d = new int[n];
            for (int i = 0; i < m; ++i)
            {
                  for (int j = 0; j < n; ++j)
                  {
                        // calculate cost/score
                        if (s.charAt(i) != t.charAt(j))
                              cost = 0;
                         else
                         {
                              if ((i == 0) || (j == 0))
                                    cost = 1;
                               else
                                    cost = p[j - 1] + 1;

                        }
                        d[j] = cost;

                        if (cost > maxLen)
                              maxLen = cost;
                  }
                  int[] swap = p;
                  p = d;
                  d = swap;
            }
            return maxLen;
      }
public static String longestCommonSubstring(String S1, String S2) //algoritmo 2
      {
          int Start = 0;
```

```
            int Max = 0;
            for (int i = 0; i < S1.length(); i++)
            {
                for (int j = 0; j < S2.length(); j++)
                {
                    int x = 0;
                    while (S1.charAt(i + x) == S2.charAt(j + x))
                    {
                        x++;
                        if (((i + x) >= S1.length()) || ((j + x) >= S2.length()))
                            break;
                    }
                    if (x > Max)
                    {
                        Max = x;
                        Start = i;
                    }
                 }
            }
            return S1.substring(Start, (Start + Max));
        }
        public static void main(String[] args)
        {
               Scanner in = new Scanner(System.in);
                      String a = in.next();
                      String b = in.next();
                      System.out.println(longestComSubstr(a, b));
                      System.out.println(longestCommonSubstring(a, b));
        }
}
```

Longest Common Subsequence

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class LongestCommonSubsequence {
        public static String lcs2(String a, String b){//algoritmo 1
            int aLen = a.length();
            int bLen = b.length();
            if(aLen == 0 || bLen == 0){
                return "";
            }else if(a.charAt(aLen-1) == b.charAt(bLen-1)){
                return lcs2(a.substring(0,aLen-1),b.substring(0,bLen-1))
                    + a.charAt(aLen-1);
            }else{
                String x = lcs(a, b.substring(0,bLen-1));
                String y = lcs(a.substring(0,aLen-1), b);

                return (x.length() > y.length()) ? x : y;
            }
        }
        public static String lcs(String a, String b) {//algoritmo 2
            int[][] lengths = new int[a.length()+1][b.length()+1];

            for (int i = 0; i < a.length(); i++)
                for (int j = 0; j < b.length(); j++)
                    if (a.charAt(i) == b.charAt(j))
```

```
                    lengths[i+1][j+1] = lengths[i][j] + 1;
                else
                    lengths[i+1][j+1] =
                        Math.max(lengths[i+1][j], lengths[i][j+1]);
        StringBuffer sb = new StringBuffer();
        for (int x = a.length(), y = b.length();
             x != 0 && y != 0; ) {
            if (lengths[x][y] == lengths[x-1][y])
                x--;
            else if (lengths[x][y] == lengths[x][y-1])
                y--;
            else {
                assert a.charAt(x-1) == b.charAt(y-1);
                sb.append(a.charAt(x-1));
                x--;
                y--;
            }
        }
        return sb.reverse().toString();
    }
    public static void main(String[] args) throws IOException {
           Scanner in = new Scanner(System.in);
           String a = in.next();
           String b = in.next();
           System.out.println(lcs2(a,b));
    }
}
```

Longest Palindromic Subsequence

```
public static String maxPalindrome(String p) {
        int n = p.length();
        char[] s = p.toCharArray();
        int[][] dp = new int[n + 1][n + 1];
        for (int i = 0; i <= n; i++)
          dp[i][0] = dp[0][i] = i;
        for (int i = 0; i < n; i++) {
          for (int j = 0; j < n - 1 - i; j++)
            dp[i + 1][j + 1] = (s[i] == s[n - 1 - j]) ? dp[i][j] : Math.min(dp[i][j + 1] + 1, dp[i
+ 1][j] + 1);
        }
        int min = n;
        int x = 0;
        int y = n;
        for (int i = 0; i <= n; i++) {
          if (min > dp[i][n - i]) {
            min = dp[i][n - i];
            x = i;
            y = n - i;
          }
        }
        String middle = "";
        for (int i = 0; i < n; i++)
        {
          if (min > dp[i][n - i - 1])
          {
            min = dp[i][n - i - 1];
            x = i;
```

```
            y = n - i - 1;
            middle = "" + s[i];
          }
        }
        StringBuilder res = new StringBuilder();
        while (x > 0 && y > 0) {
          int a = dp[x - 1][y - 1];
          int b = dp[x - 1][y];
          int c = dp[x][y - 1];
          int m = Math.min(a, Math.min(b, c));
          if (m == a)
          {
            res.append(s[x - 1]);
            --x;
            --y;
          }
          else if (m == b)
            --x;
          else
            --y;
        }
        return res.reverse() + middle + res;
      }
```

Longest Palindromic Substring

1.

```
//Time O(n^2) Space O(n^2)      DP
public class LongestPalindromicSubstring {
      public static String longestPalindrome1(String s) {
          if(s==null || s.length()<=1)
              return s;
          int len = s.length(),maxLen = 1;
          boolean [][] dp = new boolean[len][len];
          String longest = null;
          for(int l=0; l<s.length(); l++){
              for(int i=0; i<len-l; i++){
                  int j = i+l;
                  if(s.charAt(i)==s.charAt(j) && (j-i<=2||dp[i+1][j-1])){
                      dp[i][j]=true;
                      if(j-i+1>maxLen)
                      {
                         maxLen = j-i+1;
                         longest = s.substring(i, j+1);
                      }
                  }
              }
          }
          return longest;
      }
```

2.

```
//Time O(n^2), Space O(1)        A Simple Algorithm
      public String longestPalindrome2(String s) {
            if (s.isEmpty())
                  return null;
```

```
        if (s.length() == 1)
            return s;
        String longest = s.substring(0, 1);
        for (int i = 0; i < s.length(); i++) {
            // get longest palindrome with center of i
            String tmp = helper(s, i, i);
            if (tmp.length() > longest.length())
                longest = tmp;
            // get longest palindrome with center of i, i+1
            tmp = helper(s, i, i + 1);
            if (tmp.length() > longest.length())
                longest = tmp;
        }
        return longest;
    }
    // Given a center, either one letter or two letter,
    // Find longest palindrome
    public String helper(String s, int begin, int end)
    {
        while (begin >= 0 && end <= s.length() - 1 && s.charAt(begin) == s.charAt(end))
        {
            begin--;
            end++;
        }
        return s.substring(begin + 1, end);
    }
```

Longest Substring Without Repeating Characters

1.

```
public static int lengthOfLongestSubstring1(String s)
    {
      if(s==null)
          return 0;
    char[] arr = s.toCharArray();
    int pre = 0;
    HashMap<Character, Integer> map = new HashMap<Character, Integer>();
    for (int i = 0; i < arr.length; i++)
    {
        if (!map.containsKey(arr[i]))
            map.put(arr[i], i);
        else
        {
            pre = Math.max(pre, map.size());
            i = map.get(arr[i]);
            map.clear();
        }
    }
    return Math.max(pre, map.size());
    }
```

2.

```
    public static int lengthOfLongestSubstring2(String s) {
      if(s==null)
          return 0;
    boolean[] flag = new boolean[256];
    int result = 0;
```

```
        int start = 0;
        char[] arr = s.toCharArray();
        for (int i = 0; i < arr.length; i++) {
              char current = arr[i];
              if (flag[current]) {
                    result = Math.max(result, i - start);
        // the loop update the new start point and reset flag array for example, abccab, when it comes
to 2nd c, it update start from 0 to 3, reset flag for a,b
                    for (int k = start; k < i; k++) {
                          if (arr[k] == current) {
                                start = k + 1;
                                break;
                          }
                          flag[arr[k]] = false;
                    }
              } else
                    flag[current] = true;
        }
        result = Math.max(arr.length - start, result);
        return result;
}
```

Shortest Palindrome

Hacia la izquerda

```
            public static String shortestPalindrome(String s) {
          int i=0;
          int j=s.length()-1;
          while(j>=0)
          {
              if(s.charAt(i)==s.charAt(j))
                  i++;
              j--;
          }
          if(i==s.length())
              return s;
          String suffix = s.substring(i);
          String prefix = new StringBuilder(suffix).reverse().toString();
          String mid = shortestPalindrome(s.substring(0, i));
          return prefix+mid+suffix;
        }
        //in          xyz
        //out         zyxyz
        //in          abcd
        //out         dcbabcd
```

Hacia la derecha

```
        public static String s;
        public static String solve(){
          int j=s.length()-1;
          int u=-1;
          boolean sw=true;
          for(int i=0;i<s.length();i++){
              if(!(i<=j))break;
              if(s.charAt(i)==s.charAt(j)){
                  j--;
                  if(sw){
                      u=i;sw=false;
                  }
```

```
            }
            else{
                j=s.length()-1;sw=true;
            }
        }
        String aux = s.substring(0,0+u);//(s.begin(),s.begin()+u);
        //reverse(aux.begin(),aux.end());
        StringBuilder aux2 = new StringBuilder(aux);
        aux2.reverse();
        return s+aux2;
    }
```

Longest Valid Parentheses

```
    public static int longestValidParentheses(String s) {
        Stack<int[]> stack = new Stack<int[]>();
        int result = 0;
/*Given a string containing just the characters '(' and ')', find the length of the longest valid
(well-formed) parentheses substring.
For "(()", the longest valid parentheses substring is "()", which has length = 2.
Another example is ")()())", where the longest valid parentheses substring is "()()", which has
length = 4.
*/
        for(int i=0; i<=s.length()-1; i++){
            char c = s.charAt(i);
            if(c=='('){
                int[] a = {i,0};
                stack.push(a);
            }else{
                if(stack.empty()||stack.peek()[1]==1){
                    int[] a = {i,1};
                    stack.push(a);
                }else{
                    stack.pop();
                    int currentLen=0;
                    if(stack.empty()){
                        currentLen = i+1;
                    }else{
                        currentLen = i-stack.peek()[0];
                    }
                    result = Math.max(result, currentLen);
                }
            }
        }
        return result;
    }
```

Permutaciones

```
public class Permute {
static int cont=0;
public static void main(String[] args)
{
   String str = "abc";
   StringBuffer strBuf = new StringBuffer(str);
   doPerm(strBuf,str.length());
```

```
   System.out.println(cont);
}
private static void doPerm(StringBuffer str, int index){
   if(index <= 0)
   {
       System.out.println(str);
       cont++;
   }
   else { //recursively solve this by placing all other chars at current first pos
       doPerm(str, index-1);
       int currPos = str.length()-index;
       for (int i = currPos+1; i < str.length(); i++) {//start swapping all other chars with current
first char
           swap(str,currPos, i);
           doPerm(str, index-1);
           swap(str,i, currPos);//restore back my string buffer
       }
   }
}
private static void swap(StringBuffer str, int pos1, int pos2){
   char t1 = str.charAt(pos1);
   str.setCharAt(pos1, str.charAt(pos2));
   str.setCharAt(pos2, t1);
}
}
```

EJEMPLO Para abc = abc acb bac bca cba cab

Combinatoria

```
class Combinatoria
{
static int count=0;
public static void printCombinations(String initial, String combined) {
   System.out.println(combined + " ");
   count++;
   for (int i = 0; i < initial.length(); i++) {
       printCombinations(initial.substring(i + 1),
               combined + initial.charAt(i));
         }
}
public static void main(String[] args) {
       printCombinations("abcd", "");
       System.out.print(count-1);
   }
}
```

Para abcd = a ab abc abcd abd ac acd ad b bc bcd bd c cd d (15 combinaciones)

Todos los posibles substrings de un string

```
import java.util.Scanner;

class SubstringsOfAString
{
static int count=0;
   public static void main(String args[])
   {
      String string, sub;
```

```
      int i, c, length;

      Scanner in = new Scanner(System.in);
      System.out.println("Enter a string to print it's all substrings");
      string  = in.nextLine();
      length = string.length();
      System.out.println("Substrings of \""+string+"\" are :-");

      for( c = 0 ; c < length ; c++ )
      {
         for( i = 1 ; i <= length - c ; i++ )
         {
            sub = string.substring(c, c+i);
            System.out.println(sub);
            count++;
         }
      }
      System.out.println(count);
   }
}
```

Para abcd = a ab abc abcd b bc bcd c cd d (10 substrings)

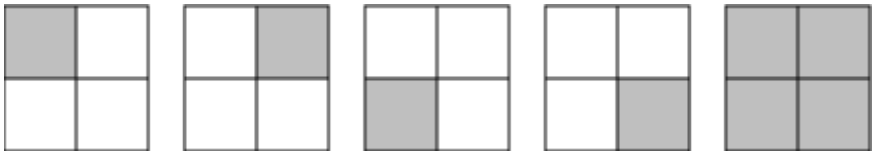
PROBLEMAS

Problema Feyman

```
public static int recur(int n)
	{
		int acu = 0;
		 if(n==1)
			return n;
		return n*n+recur(n-1);
	}
```

Ejemplo:

Este es el caso para 2,es decir si meto un dos por teclado como resultado tengo 5 cuadrados



((n)*(n+1)*(n * 2 + 1))/6

Calcular Mes Dias Y Anios (dato de entrada dias)

```
public static void main(String[] args)
	{
		Scanner in = new Scanner(System.in);
		int dias = in.nextInt();
		int ano =0, mes = 0;
		while (dias>=365)
		{
			dias = dias-365;
			ano++;
		}
		while(dias>=30)
		{
			dias = dias-30;
			mes++;
		}
		System.out.println(ano+" ano(s)");
		System.out.println(mes+" mes(es)");
		System.out.println(dias+" dia(s)");
	}
```

Calcular Hora Minuto Y Segundo (dato de entrada segundos)

```
public class TimeConversion {
	public static void main(String[] args)
	{
		Scanner in = new Scanner(System.in);
		int n = in.nextInt();
		int hora =0, minuto = 0;
		while (n>=3600)
		{
			n = n-3600;
			hora++;
		}
		while(n>=60)
```

```
            {
                  n = n-60;
                  minuto++;
            }
            System.out.println(hora+":"+minuto+":"+n);
      }  }
```

Numeros Sin Amigos

```
import java.util.Scanner;
 class SinAmigos
 {
      public static void main(String[] args)
      {
            Scanner in = new Scanner(System.in);
                  while (in.hasNextInt())
                  {
                        int n=in.nextInt();
                        int a1=0;
                        int a2=0;
                        for (int j = 1; j <= n/2; j++)
                        {
                              if(n%j==0)
                                    a1+=j;
                        }
                        for (int i = 1; i <= a1/2; i++)
                        {
                              if(a1%i==0)
                                    a2+=i;
                        }
                        if(a2==n)
                              System.out.println(a1);
                        else
                              System.out.println("-1");
                   }
      }
}
```

Peters Smokes 10346 - UVA

```
                  int n = in.nextInt();
                  int k = in.nextInt();
                  System.out.println( n + (n - 1) / (k - 1));
```

Peter tiene n cigarrillos. Él les fuma uno a uno manteniendo todas las colillas. Fuera de k> 1 colillas que puede rodar un nuevo cigarrillo. ¿Cuántos cigarrillos puede tener Peter?

Magic Formula 11934 - UVA

You are given a quadratic function,

$$f(x) = ax^2 + bx + c$$

You are also given a divisor d and a limit L. How many of the function values $f(0), f(1), \ldots, f(L)$ are divisible by d?

```
int r = 0;
    for (int i = 0; i <= l; i++) {
      if ((a * i * i + b * i + c) % d == 0)
        r++;
```

```
            }
               System.out.println(r);
```

Nacional 2013

Ajtaphi

```
import java.util.Scanner;

class Ajtapi
{
	int[] vectorPosiciones;

	static class Nodo
	{
		String[] palabras;
		Nodo siguiente;
	}
	Nodo primero = null;
	Nodo ultimo = null;

	public void adicionarEnLista(String palabra1, String palabra2)
	{
		Nodo cajaNueva = new Nodo();
		cajaNueva.palabras = new String[2];
		cajaNueva.palabras[0] = palabra1;
		cajaNueva.palabras[1] = palabra2;
		cajaNueva.siguiente = null;
		if (this.primero == null)
		{
			this.primero = cajaNueva;
			this.ultimo = cajaNueva;
		}
		else
		{
			this.ultimo.siguiente = cajaNueva;
			this.ultimo = cajaNueva;
		}
	}


	public void aumentaPosiciones(int posicion)
	{
		if (posicion>0)
		{
			vectorPosiciones[posicion-1]++;
			if(vectorPosiciones[posicion-1]>1)
			{
				vectorPosiciones[posicion-1]=0;
				aumentaPosiciones(--posicion);
			}
		}
	}

	public boolean verificaVector()
	{
		boolean respuesta = true;
		for(int i=0;i<this.vectorPosiciones.length;i++)
```

```
		{
			if(vectorPosiciones[i]==0)
			{
				respuesta = false;
				break;
			}
		}
		return respuesta;
	}

	public static void main(String[] args)
	{
		Scanner entrada = new Scanner(System.in);
		do
		{
			int cantidad = entrada.nextInt();
			Ajtapi lista = new Ajtapi();
			if (cantidad == 0)
				break;
			lista.vectorPosiciones = new int[cantidad];
			String primeraCadena="";
			for (int i = 0; i<cantidad; i++)
			{
				String palabra1 = entrada.next();
				String palabra2 = entrada.next();
				lista.adicionarEnLista(palabra1,palabra2);
				primeraCadena = primeraCadena+palabra1+palabra2;
			}
			boolean bandera=true;
			String salida = "NO EXISTE";

	if(primeraCadena.contains("a")&&primeraCadena.contains("e")&&primeraCadena.contains("i")&&prim
eraCadena.contains("o")&&primeraCadena.contains("u"))
				do
				{
					String cadena = lista.primero.palabras[lista.vectorPosiciones[0]];
				Nodo aux = lista.primero.siguiente;

				if (lista.verificaVector())
						bandera = false;

				int posicion = 1;
					while(aux!=null)
					{
						int subIndice = lista.vectorPosiciones[posicion];
						cadena = cadena+aux.palabras[subIndice];
						posicion++;
						aux = aux.siguiente;
					}
					lista.aumentaPosiciones(cantidad);


	if(cadena.contains("a")&&cadena.contains("e")&&cadena.contains("i")&&cadena.contains("o")&&cad
ena.contains("u"))
					{
						salida = "EXISTE";
						break;
					}
				}while(bandera);
```

```
                System.out.println(salida);
            }while(true);
        }
}
```

Cachinita ball

```
import java.util.Scanner;
public class CachinitaBallComentado
{
	int[][] lanzamientos;
	int izqX,izqY,derX,derY;
	double
A,B,C,MP,A1,B1,C1,C2,distanciaEje,distancia,distanciaPR1,distanciaPR2,distanciaP1,distanciaP2;
	static class NodoSemilla
	{
		int posX;
		int posY;
		boolean marcado;
		NodoSemilla siguiente;
	}
	NodoSemilla primero=null;

	public void adicionarEnLista(int x,int y)
	{
		NodoSemilla cajaNueva = new NodoSemilla();
		cajaNueva.posX = x;
		cajaNueva.posY = y;
		cajaNueva.marcado = true;
		cajaNueva.siguiente = null;
		if (this.primero == null)
			this.primero = cajaNueva;
		else
		{
			NodoSemilla aux=primero;
			while (aux.siguiente!=null)
				aux = aux.siguiente;
			aux.siguiente = cajaNueva;
		}
	}
	public int contarSemillas()
	{
		int cont = 0;
		NodoSemilla aux = primero;
		while(aux!=null)
		{
			if(!aux.marcado)
				cont++;
			aux = aux.siguiente;
		}
		return cont;
	}
	//metodo que devuelve la distancia de un punto a una recta
	public static void main(String[] args)
	{
		Scanner sc = new Scanner( System.in );
		//para lectura de datos
		int NC = sc.nextInt();
		int contador=1;
```

```
		do
		{
			CachinitaBallComentado caso = new CachinitaBallComentado();
			int N = sc.nextInt();
			int M = sc.nextInt();
			caso.lanzamientos = new int[N][6];
			for(int i=0 ; i<N ; i++)
			{
				caso.lanzamientos[i][0]=sc.nextInt();
				caso.lanzamientos[i][1]=sc.nextInt();
				caso.lanzamientos[i][2]=sc.nextInt();
				caso.lanzamientos[i][3]=sc.nextInt();
				caso.lanzamientos[i][4]=sc.nextInt();
				caso.lanzamientos[i][5]=sc.nextInt();
			}
			for(int j=0 ; j<M ; j++)
			{
				int posSemX = sc.nextInt();
				int posSemY = sc.nextInt();
				caso.adicionarEnLista(posSemX, posSemY);
//aqui se resuelve el problema defino la ecuacion de una recta que pasa por 2 puntos calculando A, B
y C
			for (int k=0 ; k<N; k++)
			{
				//System.out.println(caso.A+"-"+caso.B+"-"+caso.C);
				NodoSemilla aux = caso.primero;
				//defino punto izquierda y derecha finales del recorrido de la canica
				if(caso.lanzamientos[k][0]<caso.lanzamientos[k][3])
				{
					caso.izqX=caso.lanzamientos[k][0];
					caso.izqY=caso.lanzamientos[k][1];

	caso.derX=(caso.lanzamientos[k][3]*caso.lanzamientos[k][5])+caso.lanzamientos[k][0];

	caso.derY=(caso.lanzamientos[k][4]*caso.lanzamientos[k][5])+caso.lanzamientos[k][1];
				}
				else
				{
					caso.derX=caso.lanzamientos[k][0];
					caso.derY=caso.lanzamientos[k][1];

	caso.izqX=(caso.lanzamientos[k][3]*caso.lanzamientos[k][5])+caso.lanzamientos[k][0];

	caso.izqY=(caso.lanzamientos[k][4]*caso.lanzamientos[k][5])+caso.lanzamientos[k][1];
				}
				//se calcula los coeficientes
				caso.A=caso.derY-caso.izqY;
				caso.B=caso.izqX-caso.derX;
				caso.C=(caso.izqY*(caso.derX-caso.izqX))+(caso.izqX*(caso.izqY-caso.derY));

//la pendiente de la recta original es -A/B y su perpencicular resulta de multiplicarla por -1/m
luego queda B/A
				caso.MP = caso.B/caso.A;
				//calculo los coeficientes de ABC de las 2 rectas perpendiculares
				caso.A1 = caso.MP;
				caso.B1 = -1;
				caso.C1 = caso.izqY-(caso.MP*caso.izqX);
				caso.C2 = caso.derY-(caso.MP*caso.derX);
```

```
                        caso.distanciaEje = Math.sqrt(Math.pow(caso.derX-
caso.izqX,2)+Math.pow(caso.derY-caso.izqY,2));
                        while (aux!=null)
                        {
                                //calculo la distancia siempre y cuando la semilla este disponible
                                if(aux.marcado)
                                {
                                        caso.distanciaPR1 =
Math.abs((caso.A1*aux.posX)+(caso.B1*aux.posY)+caso.C1)/Math.sqrt((caso.A1*caso.A1)+(caso.B1*caso.B1)
);
                                        caso.distanciaPR2 =
Math.abs((caso.A1*aux.posX)+(caso.B1*aux.posY)+caso.C2)/Math.sqrt((caso.A1*caso.A1)+(caso.B1*caso.B1)
);
                                        if
(Math.round(caso.distanciaEje)==Math.round(caso.distanciaPR1+caso.distanciaPR2))
                                        {
                                                caso.distancia =
Math.abs((caso.A*aux.posX)+(caso.B*aux.posY)+caso.C)/Math.sqrt((caso.A*caso.A)+(caso.B*caso.B));
                                                if(caso.distancia<=caso.lanzamientos[k][2])
                                                        aux.marcado = false;
                                        }
                                        else
                                        {
                                                caso.distanciaP1 = Math.sqrt(Math.pow(caso.izqX-
aux.posX,2)+Math.pow(caso.izqY-aux.posY,2));
                                                caso.distanciaP2 = Math.sqrt(Math.pow(caso.derX-
aux.posX,2)+Math.pow(caso.derY-aux.posY,2));
                                                if
(caso.distanciaP1<=caso.lanzamientos[k][2]||caso.distanciaP2<=caso.lanzamientos[k][2])
                                                                aux.marcado = false;
                                        }
                                }
                                aux = aux.siguiente;
                        }
                }
                System.out.printf("Caso %d: %d\n",contador,caso.contarSemillas());
                contador++;
        }while (contador<=NC);
    }
}
```

Cuantos Grafos

```
import java.util.Scanner;
class CuantosGrafos
{
    int dimension;
    boolean [] vectorConexionesFila;
    boolean [] vectorConexionesColumna;
    boolean [][] matrizAdyacencias;
    public void buscarAdyacencias(char fc, int posicion)
    {
        if ((fc == 'f')&&(!vectorConexionesFila[posicion]))
        {
            this.vectorConexionesFila[posicion]=true;
            for (int i = 0; i<this.dimension; i++)
            if(matrizAdyacencias[posicion][i])
            {
                buscarAdyacencias('f',i);
                buscarAdyacencias('c',i);
```

```
			}

		}
		if ((fc == 'c')&&(!vectorConexionesColumna[posicion]))
		{
			this.vectorConexionesColumna[posicion]=true;
			for (int j = 0; j<this.dimension; j++)
			if(matrizAdyacencias[j][posicion])
			{
				buscarAdyacencias('f',j);
				buscarAdyacencias('c',j);
			}
		}
}
		public static void main(String[] args)
{
		Scanner entrada = new Scanner(System.in);
		int tamanio = 10000;
		int[] primos = new int[tamanio];
		for (int i=4; i<tamanio; i=i+2)
			primos[i]=1;
		int p = 3;
		//Se coloca 3 por que el subindice que necesitamos en
		while (p<tamanio/p)
		{
			if (primos[p]==0)
			{
				for(int j=p*p;j<tamanio;j=j+p+p)
					primos[j]=1;
			}
			p=p+2;
		}
		do
		{
			int cantidad1 = entrada.nextInt();
			int cantidad2 = entrada.nextInt();
			CuantosGrafos variables = new CuantosGrafos();
			if (cantidad1 == 0 && cantidad2 == 0)
				break;
			int dimension=cantidad2-cantidad1+1;
			variables.dimension = dimension;
			variables.vectorConexionesFila = new boolean[dimension];
			variables.vectorConexionesColumna = new boolean[dimension];
			variables.matrizAdyacencias = new boolean[dimension][dimension];

			for (int i = 0; i<dimension; i++)
				for (int j = i+1; j<dimension; j++)
				{
					int comparador = i+cantidad1+j+cantidad1;
					if(primos[comparador]==0)
						variables.matrizAdyacencias[i][j]=true;

				}
			int contador = 0;
			for (int h=0; h<dimension; h++)
			{
				if(!variables.vectorConexionesFila[h])
				{
					variables.buscarAdyacencias('f', h);
```

```
                                contador++;
                            }
                        }
                        System.out.println(contador);
                }while(true);
        }
}
```

Inundacion

```
import java.util.Arrays;
import java.util.Comparator;
import java.util.Scanner;

class Inundacion
{

        static int padre[];

        static void MakeSet( int n )
        {
            for( int i = 1 ; i <= n ; ++i )
              padre[ i ] = i;
        }

        static double [][] coordenadas;
        static boolean [][] puentesAdyacentes;
        static int contador=1;

        static int Find( int x )
        {
            return ( x == padre[ x ] ) ? x : ( padre[ x ] = Find( padre[ x ] ) );
        }

        static void Union( int x , int y )
        {
            padre[ Find( x ) ] = Find( y );
        }
        static boolean sameComponent( int x , int y )
        {
            if( Find( x ) == Find( y ) ) return true;
            return false;
        }

        static int V , E;

        static class Edge implements Comparator<Edge>
        {
            int origen;
            int destino;
            double peso;
            Edge(){}

              @Override
              public int compare(Edge e1 , Edge e2 )
              {

                    return (int) (e1.peso - e2.peso);
              }
```

```
};

static Edge arista[];
static Edge MST[];

static void KruskalMST()
{
    int origen , destino;
    double peso;
    int maximo = 0;
    double total = 0;
    int numAristas = 0;

    MakeSet( V );
    Arrays.sort( arista , 0 , E , new Edge() );

    for( int i = 0 ; i < E ; ++i )
    {
        origen = arista[ i ].origen;
        destino = arista[ i ].destino;
        peso = arista[ i ].peso;

        if( !sameComponent( origen , destino ) )
        {
            total += peso;
            MST[ numAristas++ ] = arista[ i ];
            Union( origen , destino );
        }
    }
    for( int i = numAristas-1 ; i >= 0 ; --i )
    {
       if(!puentesAdyacentes[MST[i].origen][MST[i].destino])
       {
             maximo =(int) MST[ i ].peso;
             break;
       }
    }
  System.out.printf("Caso %d: %d\n",contador,maximo);
}

public static double calculaDistancia(int i, int j)
{
      double valorX = Math.pow(coordenadas[i][0]-coordenadas[j][0],2);
      double valorY = Math.pow(coordenadas[i][1]-coordenadas[j][1],2);
      return(Math.sqrt(valorX+valorY));
}

public static void main(String[] args)
{
      Scanner sc = new Scanner( System.in );

      int NC = sc.nextInt();
      do
      {
            V = sc.nextInt();

            E = V*(V-1)/2;
            padre = new int[E+1];
            arista = new Edge[E+1];
```

```
                MST = new Edge[E+1];

                coordenadas = new double[V][2];
                puentesAdyacentes = new boolean[V][V];
                 for( int i = 0 ; i < V ; ++i )
                 {
                        String a = sc.next();
                        String b = sc.next();
                        coordenadas[i][0]=Double.parseDouble(a);
                        coordenadas[i][1]=Double.parseDouble(b);
                 }

                int M = sc.nextInt();
                for(int h=0 ; h<M ; ++h)
                {
                       int origen = sc.nextInt();
                       int destino = sc.nextInt();
                       puentesAdyacentes[origen-1][destino-1]=true;
                       puentesAdyacentes[destino-1][origen-1]=true;
                }

                int indice = 0;
              for( int i = 0 ; i < V ; ++i )
                for( int j = i+1 ; j < V ; ++j )
                    {
                        arista[ indice ] = new Edge();
                       arista[ indice ].origen = i;
                        arista[ indice ].destino = j;
                        if(puentesAdyacentes[i][j])
                             arista[ indice ].peso = 0;
                        else
                             arista[ indice ].peso = calculaDistancia(i, j);

                        indice++;
                    }

                KruskalMST();

                contador++;

                NC--;
            }while (NC>0);
      }
}
```

Llama Ola que hace

```
import java.util.Scanner;

public class Llama {

      public static void main(String[] args)
      {
            Scanner entrada = new Scanner(System.in);
            int cantidad;
            int ax,ay,bx,by,cx,cy;
            int consulx,consuly;
            String result ="";
            do
```

```
	{
		ax=entrada.nextInt();
		ay=entrada.nextInt();
		bx=entrada.nextInt();
		by=entrada.nextInt();
		cx=entrada.nextInt();
		cy=entrada.nextInt();
		if(ax==0&&ay==0&&bx==0&&by==0&&cx==0&&cy==0)
			break;
		cantidad=entrada.nextInt();
		for (int i = 0; i < cantidad; i++)
		{
			consulx = entrada.nextInt();
			consuly = entrada.nextInt();
			result += calcular(ax, ay, bx, by, cx, cy, consulx, consuly);
		}


	}while(ax!=0&&ay!=0&&bx!=0&&by!=0&&cx!=0&&cy!=0);
	System.out.println(result);

}

public static String calcular(int ax,int ay,int bx,int by,int cx,int cy,int x,int y)
{
	boolean[] regla = new boolean[30];
	boolean[] reglaA = new boolean[30];
	boolean[] reglaB = new boolean[30];
	boolean[] reglaC = new boolean[30];
	for (int i = ax; i < regla.length; i=i+ay)
	{
		regla[i]=true;
		reglaA[i]=true;
	}
	for (int j = bx; j < regla.length; j=j+by)
	{
		regla[j]=true;
		reglaB[j]=true;
	}
	for (int k = cx; k < regla.length; k=k+cy)
	{
		regla[k]=true;
		reglaC[k]=true;
	}
	if(regla[x]==true&&regla[y]==true)
	{
		if(reglaA[x]==true&&reglaA[y]==true)
		{
			return "ola k ase\n";
		}
		else if(reglaB[x]==true&&reglaB[y]==true)
		{
			return "ola k ase\n";
		}
		else if(reglaC[x]==true&&reglaC[y]==true)
		{
			return "ola k ase\n";
		}
		else if(reglaA[x]==true)
```

```
{
	if(reglaB[y]==true)
	{
		for (int i = ax; i < regla.length; i=i+ay)
		{
			if(reglaB[i]==true)
				return "ola k ase\n";
		}
		for (int j = ax; j < regla.length; j+=ay)
		{
			if(reglaC[j]==true)
			{
				for (int k = cx; k < regla.length; k=k+cy)
				{
					if(reglaB[k]==true)
						return "ola k ase\n";
				}
			}
		}
	}
}
	else if (reglaB[x]==true)
	{
		if(reglaC[y]==true)
		{
			for (int i = bx; i < regla.length; i+=by)
			{
				if(reglaC[i]==true)
					return "ola k ase\n";
			}
			for (int j = bx; j < regla.length; j+=by)
			{
				if(reglaA[j]==true)
				{
					for (int k = ax; k < regla.length; k+=ay)
					{
						if(reglaC[k]==true)
							return "ola k ase\n";
					}
				}
			}
		}
	}
		else if(reglaC[x]==true)
		{
			if(reglaA[y]==true)
			{
				for (int i = cx; i < regla.length; i+=cy)
				{
					if(reglaA[i]==true)
						return "ola k ase\n";
				}
				for (int j = cx; j < regla.length; j+=cy)
				{
					if(reglaB[j]==true)
					{
						for (int k = bx; k < regla.length; k+=by)
						{
							if(reglaA[k]==true)
```

```
                                                                        return "ola k ase\n";
                                                                    }
                                                                }
                                                            }
                                                        }
                                                    }
                                return "no\n";
                        }
                        else
                                return "no\n";
        }
}
```

La mejor empresa

```
import java.util.Scanner;
public class Rendimiento {
        //public static String cadena = "";
        public static void main(String[] args)
        {
                Scanner entrada = new Scanner(System.in);
            int n, mesIni, mesFin, suma, mayor, currMI, currMF, v;
             n = entrada.nextInt();
            while (n > 0) {

                currMF = 0;
                currMI = 0;
                mesIni = 0;
                mesFin = 0;
                suma = 0;
                mayor = -100;
                boolean sirve = false;

                for (int i = 0; i < n; i++) {
                    v = entrada.nextInt();
                    suma += v;

                    if(v > 0)sirve = true;

                    if (suma > 0) {
                        currMF = i;
                    }else{
                        suma = 0;
                        currMF = i+1;
                        currMI = i+1;
                    }
                    if (suma == mayor) {
                        if (mesFin < currMF) {
                            mesFin = currMF;
                            mesIni = currMI;
                        }else if(mesFin == currMF){
                            if (mesFin - mesIni > currMF - currMI) {
                                mesFin = currMF;
                                mesIni = currMI;
```

```
                        }
                    }
                }
                if (suma > mayor) {
                    mayor = suma;
                    mesIni = currMI;
                    mesFin = currMF;
                }
            }

            if (!sirve) {
               System.out.print("0 -1 -1\n");
            }else
                 System.out.print( mayor+ " " + mesIni + " " + mesFin + "\n");
            n = entrada.nextInt();
        }
    }
}
```

Poquer de huevo

```
class poquer {

	public static void main(String[] args) {
		Scanner in = new Scanner(System.in);
		int[] x=new int[5];
		for (int i = 0; i < 5; i++)
			x[i]=in.nextInt();
		while (x[0]!=0)
		{
			if(x[0]==0)
				break;
			int a = 0;
			for (int i = 0; i < 5; i++)
			{
				int cont=0;
				for (int j = 0; j < 5; j++)
				{
					if (x[i]==x[j])
						cont++;
				}
				if (cont==3)
				 a=x[i];
			}
			if(a==0)
				System.out.println("No");
			else
			{
			int contaux=0;
			for (int i = 0; i < 5; i++)
			{
				if ((7-x[i])==a)
					contaux++;
			}
			if (contaux>=1)
				System.out.println("Poquer de huevo");
			else
				System.out.println("No");
			}
				for (int i = 0; i < 5; i++)
```

```
                                x[i]=in.nextInt();
                }
        }

}
```

CUMPLE

```
import java.util.Arrays;
import java.util.Scanner;
public class Cumple {
	public static void main(String[] args)
	{
		Scanner in = new Scanner(System.in);
		while (in.hasNextInt())
		{
			int casos= in.nextInt();
				while(casos>0)
				{
					int nRegalos= in.nextInt(), PesoTotal= in.nextInt();
					int c=-1;
					int auxPeso=0;
					int[] pesos= new int[nRegalos];
					for (int i = 0; i < pesos.length; i++)
						pesos[i]=in.nextInt();
					Arrays.sort(pesos);

					for (int i = 0; i < pesos.length; i++)
					{
						if (auxPeso<=PesoTotal)
						{
							auxPeso = auxPeso+ pesos[i];
							c++;
						}
					}
					System.out.println(c);
					casos--;
				}
			}
	}
                                                }
```

PIEDRA PAPEL TIJERA

```
import java.util.Scanner;
class piedrapapeltijera {
	public static void main(String[] args)
	{
		Scanner in = new Scanner(System.in);
		while(in.hasNextInt())
		{
			int casos = in.nextInt();
			for (int i = 0; i < casos; i++)
			{
				int n = in.nextInt();
				int angela=0,bernardo = 0;
				String cad = in.next();
				for (int j = 0; j < cad.length(); j+=2)
				{
					if(cad.charAt(j)=='I' && cad.charAt(j+1)=='A')
						bernardo++;
					else if(cad.charAt(j)=='I' && cad.charAt(j+1)=='T')
```

```
                                        angela++;
                                  else if(cad.charAt(j)=='A' && cad.charAt(j+1)=='I')
                                        angela++;
                                  else if(cad.charAt(j)=='A' && cad.charAt(j+1)=='T')
                                        bernardo++;
                                  else if(cad.charAt(j)=='T' && cad.charAt(j+1)=='I')
                                        bernardo++;
                                  else if(cad.charAt(j)=='T' && cad.charAt(j+1)=='A')
                                        angela++;
                            }
                            if(angela>bernardo)
                                  System.out.println("Angela gana");
                            else if(bernardo>angela)
                                  System.out.println("Bernardo gana");
                            else
                                  System.out.println("Empate");
                      }
                }
          }
}
```

SPRITES

```
import java.util.Scanner;

public class SPRITES
{
      public static void main(String[] args)
      {
            Scanner entrada= new Scanner(System.in);
            while(entrada.hasNextInt())
            {
                  int cont=1;
                  int casos = entrada.nextInt();
                  for (int i = 0; i < casos; i++)
                  {
                        int filas = entrada.nextInt();
                        int columnas = entrada.nextInt();
                        int[][]sprite = new int[filas][columnas];
                        boolean[][]matrizAux = new boolean[filas][columnas];
                        for (int j = 0; j < sprite.length; j++)
                        {
                              for (int j2 = 0; j2 < sprite[1].length; j2++)
                              {
                                    sprite[j][j2]=entrada.nextInt();
                              }
                        }
                        int frames=0;
                        int []color  = new int[10];
                        for (int j = 0; j < sprite.length; j++)
                        {
                              for (int j2 = 0; j2 < sprite[1].length; j2++)
                              {
                                    if(matrizAux[j][j2]==false&&sprite[j][j2]!=0)
                                    {
                                          frames++;
                                          matrizAux[j][j2]=true;
                                          int pos = recorrer(matrizAux, sprite, j, j2);
                                          color[pos]++;
```

```
                                        }
                                }
                        }
                        System.out.println("Caso #"+cont+": "+frames);
                        for (int k = 1; k < color.length; k++)
                        {
                                if(k!=color.length-1)
                                        System.out.print(color[k]+" ");
                                else
                                        System.out.print(color[k]+"\n");
                        }
                        cont++;
                }
        }
    }

    public static int recorrer(boolean matrizAux [][],int sprite [][],int j,int j2)
    {
                //DERECHA
                if (j2+1<=matrizAux[1].length-1 &&matrizAux[j][j2+1]==false &&
sprite[j][j2+1]==sprite[j][j2])
                {
                        matrizAux[j][j2+1]=true;
                        recorrer(matrizAux, sprite, j, j2+1);
                }
                //IZQUIERDA
                else if(j2-1>=0&&matrizAux[j][j2-1]==false && sprite[j][j2-1]==sprite[j][j2])
                {
                        matrizAux[j][j2-1]=true;
                        recorrer(matrizAux, sprite, j, j2-1);
                }
                //ARRIBA
                else if(j-1>=0&&matrizAux[j-1][j2]==false && sprite[j-1][j2]==sprite[j][j2])
                {
                        matrizAux[j-1][j2]=true;
                        recorrer(matrizAux, sprite, j-1, j2);
                }
                //ABAJO
                else if(j+1<=matrizAux.length-1&&matrizAux[j+1][j2]==false &&
sprite[j+1][j2]==sprite[j][j2])
                {
                        matrizAux[j+1][j2]=true;
                        recorrer(matrizAux, sprite, j+1, j2);
                }
                //DIAGONAL ARRIBA DERECHA
                else if(j-1>=0 && j2+1<=sprite[1].length-1 &&matrizAux[j-1][j2+1]==false &&
sprite[j-1][j2+1]==sprite[j][j2])
                {
                        matrizAux[j-1][j2+1]=true;
                        recorrer(matrizAux, sprite, j-1, j2+1);
                }
                //DIAGONAL ARRIBA IZQUIERDA
                else if(j-1>=0 && j2-1>=0&&matrizAux[j-1][j2-1]==false && sprite[j-1][j2-
1]==sprite[j][j2])
                {
                        matrizAux[j-1][j2-1]=true;
                        recorrer(matrizAux, sprite, j-1, j2-1);
                }
                //DIAGONAL ABAJO IZQUIERDA
```

```
                    else if(j+1<=sprite.length-1 && j2-1>=0&&matrizAux[j+1][j2-1]==false &&
sprite[j+1][j2-1]==sprite[j][j2])
                    {
                          matrizAux[j+1][j2-1]=true;
                          recorrer(matrizAux, sprite, j+1, j2-1);
                    }
                    //DIAGONAL ABAJO DERECHA
                    else if(j+1<=sprite.length-1 &&
j2+1<=sprite[1].length&&matrizAux[j+1][j2+1]==false && sprite[j+1][j2+1]==sprite[j][j2])
                    {
                          matrizAux[j+1][j2+1]=true;
                          recorrer(matrizAux, sprite, j+1, j2+1);
                    }
                    else
                    {
                          return sprite[j][j2];
                    }
                    return sprite[j][j2];
      }
```

Nacional 2014

Area Region

```
public static void main(String[] args) {
            Scanner in = new Scanner(System.in);
            int casos = in.nextInt();
            int caso=1;
            while (casos>0)
            {
                  int n = in.nextInt();
                  String letras = in.next();
                  int [][]vec = new int [n+1][2];
                  vec[0][0]=0;
                  vec[0][0]=0;
                  for (int i = 0; i < n; i++)
                  {
                        if (letras.charAt(i)=='n') {
                              vec[i+1][0] = vec[i][0];
                              vec[i+1][1] = vec[i][1]+1;
                        }
                        else
                        {
                              if (letras.charAt(i)=='s') {
                                    vec[i+1][0] = vec[i][0];
                                    vec[i+1][1] = vec[i][1]-1;
                              }
                              else
                              {
                                    if (letras.charAt(i)=='e') {
                                          vec[i+1][0] = vec[i][0]+1;
                                          vec[i+1][1] = vec[i][1];
                                    }
                                    else
                                    {
                                          if (letras.charAt(i)=='o') {
                                                vec[i+1][0] = vec[i][0]-1;
                                                vec[i+1][1] = vec[i][1];
```

```
                                        }
                                }
                        }
                }
        }
        int result = 0;
        for (int i = 0; i < n-1; i++)
        {
                result = result + ((vec[i][0]*vec[i+1][1])-(vec[i+1][0]*vec[i][1]));
        }
        if(result<0)
                result = result/(-2);
        else
                result = result/2;
        System.out.println("Caso #"+caso+": "+result);
        casos--;
        caso++;
}}
```

Bloques

```
Scanner entrada = new Scanner(System.in);
        int[]visibles = new int[1000];
        visibles[0]=0;
        int cantidad=6;
        for (int i = 1; i < visibles.length; i++)
        {
                visibles[i]+=cantidad;
                cantidad +=4;
        }
        int casos;
        do
        {
                casos = entrada.nextInt();
                if(casos==0)
                        break;
                int[]edificios = new int[casos];
                for (int i = 0; i < edificios.length; i++)
                {
                        edificios[i] = entrada.nextInt();
                }
                int suma=0;
                for (int i = 0; i < edificios.length; i++)
                {
                        if(edificios[i]!=0)
                        {
                                int aux = visibles[edificios[i]];
                                if((i-1)>=0)
                                {
                                        if(edificios[i-1]<=edificios[i])
                                        {
                                                aux-=edificios[i-1];
                                        }
                                        else
                                        {
                                                aux-=edificios[i];
                                        }
                                }
                                if((i+1)<=edificios.length-1)
                                {
                                        if(edificios[i+1]<=edificios[i])
```

```
                                {
                                        aux-=edificios[i+1];
                                }
                                else
                                {
                                        aux-=edificios[i];
                                }
                        }
                        suma+=aux;
                }
        }
        System.out.println(suma);
}while(casos!=0);
```

Deudas

```
Scanner entrada = new Scanner(System.in);
        int personas;
        do
        {
                personas = entrada.nextInt();
                if(personas==0)
                        break;
                int []deudas = new int[personas];
                for (int i = 0; i < deudas.length; i++)
                        deudas[i]=entrada.nextInt();
                int numConsul = entrada.nextInt();
                while(numConsul>0)
                {
                        int consulta = entrada.nextInt();
                        ArrayList<Integer> array = new ArrayList<Integer>();
                        for (int i = 0; i < deudas.length; i++)
                        {
                                int suma = deudas[i];
                                if(suma==consulta)
                                {
                                        array.add(i);
                                        array.add(i);
                                }
                                for (int j = i+1; j < deudas.length; j++)
                                {
                                        if(suma+deudas[j]<=consulta)
                                        {
                                                suma +=deudas[j];
                                                if(suma==consulta)
                                                {
                                                        array.add(i);
                                                        array.add(j);
                                                        break;
                                                }
                                        }
                                        else
                                                break;
                                }
                        }
                        if(array.size()>0)
                        {
                                for (int i = 0; i < array.size(); i+=2)
                                        System.out.println(array.get(i)+" "+array.get(i+1));
                        }
```

```
                                else
                                        System.out.println("-1");
                                numConsul--;
                            }
                        }while(personas!=0);
```

Nuevas Palabras

```
import java.util.TreeSet;
public class nuevaspalabrasultimo {
	public static String s;
	int n,m;
	public static TreeSet<String>st = new TreeSet<String>();
	public static String solve()
	{
	    int lim=0;
	    while(true)
	    {
	        lim++;
	        if(lim==s.length())
	             break;
	        String a="";
	        for(int i=0;i<lim;i++)
	        {
	            if(lim==s.length())
	             break;
	                a+=s.charAt(i);
	        }
	        String b="";
	        if(lim==s.length())
	             break;
	        for(int i=lim;i<s.length();i++)
	            b+=s.charAt(i);
	        if(a.length()>0&&b.length()>0&&st.contains(a)&&st.contains(b))
	        {
	             String aux=a+b;
	                st.add(aux);
	                return "SI";
	        }
	    }
	    return "NO";
	}
	public static void main(String[] args)
	{
	       Scanner in = new Scanner(System.in);
	       int n = in.nextInt();
	  st.clear();
	  for(int i=0;i<n;i++)
	  {
	      s = in.next();
	      st.add(s);
	  }
	  int m = in.nextInt();
	  for(int i=0;i<m;i++)
	  {
	      s =in.next();
	      System.out.println(solve());
	  }}
	}
```

shaskpeare

```
Scanner entrada = new Scanner(System.in);
		while(entrada.hasNextInt())
		{
			int n = entrada.nextInt();
			int [] vector = new int[100010];
			int [] cont = new int[100010];
			int aux =1;
			for (int i = 0; i < n; i++)
			{
				int x1 = entrada.nextInt();
				int x2 = entrada.nextInt();
				int y = entrada.nextInt();
				for (int j = x1; j <= x2; j++)
				{
					if(y>vector[j])
					{
						vector[j]=y;
						cont[j]=aux;
					}
				}
				aux++;
			}
			Arrays.sort(cont);
			for (int i = 1; i < aux; i++)
			{
				int aux2 = 0;
				for (int j = 0; j < cont.length; j++)
				{
					if(cont[j]==i)
						aux2++;
				}
				System.out.println(aux2);
			}
		}
```

Nacional 2015

Next Palindromic Numbers

```
public static String procesar(String numero){
			String result="";
			String s = numero;
			int n = s.length();
			char[] num = s.toCharArray();
			char[] original = num.clone();
			int medio = n/2;
			int x = medio - 1;
			int y = x+2;
			if(n % 2 == 0) y = x+1;
			while(x >= 0 && y < n) {
				if(num[x] != num[y]) {
					num[y] = num[x];
				}
				x--; y++;
			}
			if(mayor(original, num)) tocarVector(num, medio-(n%2==0?1:0), medio);
			if(num[0] == '0' || (num[0] == '1' && num.length == 1)) result +="1";;
```

```
            return result+=new String(num);
}

public static boolean mayor(char[] num1, char[] num2) {
      for(int i=0; i<num1.length; i++) {
            if(num1[i] < num2[i]) return false;
            if(num1[i] > num2[i]) return true;
      }
      return true;
}

public static void tocarVector(char[] num, int izq, int der) {
      if(num[izq] < '9') {
            num[izq] = num[der] = (char) (num[izq]+1);
      }
      else {
            while(izq >=0 && num[izq] == '9') {
                  num[izq] = num[der] = '0';
                  izq--; der++;
            }
            if(izq >= 0) {
                  num[izq] = num[der] = (char) (num[izq]+1);
            }
      }
      if(izq < 0) {
            num[num.length-1] = '1';
      }
}

public static void main(String[] args){
      // TODO Auto-generated method stub
      Scanner entrada = new Scanner(System.in);
      while(entrada.hasNextInt())
      {
            int cantidad = entrada.nextInt();
            String numero = entrada.next();
            while(cantidad-->0)
            {
                  if(numero.length()==1)
                  {
                        int n = Integer.parseInt(numero.charAt(0)+"");
                        if(n==9)
                        {
                              numero = (n+1)+"";
                              cantidad++;
                        }
                        else
                        {
                              System.out.println(n+1);
                              numero = (n+1) + "";
                        }
                  }
                  else
                  {
                        numero = procesar(numero);
                        System.out.println(numero);
                  }
            }
      }     }
```

Sub-expression Counting

```
      static char[] T, P;
        static int n, m;
        static int [] b;
        static void kmpPreprocess() {
          int i = 0, j = -1; b[0] = -1;
          while (i < m) {
            while (j >= 0 && P[i] != P[j]) j = b[j];
            i++; j++;
            b[i] = j;
        } }
        static int kmpSearch() {
          int i = 0, j = 0,cont=0;
          while (i < n) {
            while (j >= 0 && T[i] != P[j]) j = b[j];
            i++; j++;
            if (j == m) {

               cont++;
              j = b[j];
            }
            }
          return cont;
          }
      public static void main(String[] args)
      {
            Scanner entrada = new Scanner(System.in);
            while(entrada.hasNext())
            {
                  String patron = entrada.next();
                  String texto = entrada.next();
                  patron = patron.replaceAll("\\d+",
"W").replaceAll("\\w+","W").replaceAll("['+','/',-]","*");
                  texto = texto.replaceAll("\\d+",
"W").replaceAll("\\w+","W").replaceAll("['+','/',-]","*");
                    T = new String(texto).toCharArray();
                P = new String(patron).toCharArray();
                n = T.length;
                m = P.length;
                b = new int[400001];
                kmpPreprocess();
                   int contador = kmpSearch();
                System.out.println(contador);
            }
      }
```

Farmer Jane

```
int N, suma, sx, sy, *w, *x, *y;

    while (scanf("%d", &N) != EOF) {
        suma = 0;
        sx=0;
        sy=0;
        w = new int[N];
        x = new int[N];
        y = new int[N];
        for (int i = 0; i < N; i++) {
```

```
            scanf("%d%d%d", &x[i], &y[i], &w[i]);
            suma += w[i];
            sx += 2*w[i]*x[i];
            sy += 2*w[i]*y[i];
        }
        double rx = (double)sx / (2.0*suma);
        double ry = (double)sy / (2.0*suma);
        double r = 0;
        for (int i = 0; i < N; i++) {
            r += w[i]*((x[i] - rx)*(x[i] - rx) + (y[i] - ry)*(y[i] - ry));
        }
        printf("%.3f\n", r);
```

Peanoland contacting Gaussland

```
#include <stdio.h>
#include <iostream>
#include <string>
#include <complex>
using namespace std;
string bin(long d){
    string r = "";
    while (d > 0) {
        if (d%2 == 0) {
            r = '0' + r;
        }else
            r = '1' + r;
        d /= 2;
    }
    return r;
}
int main(){
    long p;
    while (scanf("%ld", &p) != EOF) {
        string b = bin(p);

        complex<double> g(-1, 1);
        complex<double> r(0, 0);
        for (int i = 0; i < (int)b.length(); i++) {
            if(b[b.length()- 1 - i] == '1')
                r += pow(g, i);
        }
        printf("%.0f %.0f\n", r.real(), r.imag());
    }
    return 0;
}
```

Interstellar Travel

```
#include <iostream>
#include <vector>
#include <map>
#include <string>
#include <queue>
#include <functional>
#include <stdio.h>
using namespace std;
#define inf 2000000000
struct edge{
    int costo, tiempo, destino;
```

```
};
struct nodo{
    int costo, tiempo, planeta, paradas;
};
auto comp = [](const nodo &a, const nodo &t) -> bool {
    if(a.costo == t.costo){
        return a.tiempo > t.tiempo;
    }else{
        return  a.costo > t.costo;
    }
};
void dijkstra(vector<vector<edge>> grafo, int o, int d, int n){
    vector<int> cost(grafo.size(), inf);
    vector<int> time(grafo.size(), inf);
    vector<int> paradas(grafo.size(), 300);
    priority_queue<nodo, vector<nodo>, decltype(comp)> cola(comp);
    nodo u;
    u.costo = 0;
    u.planeta = o;
    u.paradas = 0;
    u.tiempo = 0;
    cola.push(u);
    nodo v;
    cost[o] = 0;
    time[o] = 0;
    bool sirve = false;
    while (!cola.empty()) {
        v = cola.top();
        cola.pop();
        if(v.paradas > n+1) continue; // mas de n paradas
        if (v.planeta == d) {
            printf("%d %d\n", v.costo, v.tiempo);
            sirve = true;
            break;
        }
        for (int i = 0; i < grafo[v.planeta].size(); i++) {
            edge e = grafo[v.planeta][i];
            if (cost[v.planeta] + e.costo < cost[e.destino]) {
                cost[e.destino] = cost[v.planeta] + e.costo;
                time[e.destino] = time[v.planeta] + e.tiempo;
                nodo w;
                w.planeta = e.destino;
                w.costo = v.costo + e.costo;//cost[e.destino];
                w.tiempo = v.tiempo + e.tiempo;//time[e.destino];
                w.paradas = v.paradas + 1;
                cola.push(w);
            }else if(cost[v.planeta] + e.costo == cost[e.destino]){
                if (time[v.planeta] + e.tiempo <= time[e.destino]) {
                    cost[e.destino] = cost[v.planeta] + e.costo;
                    time[e.destino] = time[v.planeta] + e.tiempo;
                    nodo w;
                    w.planeta = e.destino;
                    w.costo = v.costo + e.costo;//cost[e.destino];
                    w.tiempo = v.tiempo + e.tiempo;//time[e.destino];
                    w.paradas = v.paradas + 1;
                    cola.push(w);
                }
            }else if(paradas[e.destino] > v.paradas + 1){
                nodo w;
```

```
                w.planeta = e.destino;
                w.costo = v.costo + e.costo;//cost[e.destino];
                w.tiempo = v.tiempo + e.tiempo;//time[e.destino];
                w.paradas = v.paradas + 1;
                cola.push(w);
            }
        }
    }
    if (!sirve) {
        printf("* *\n");
    }
}
int main(int argc, const char * argv[]) {
    int p, f, q, i, si, n;
    bool primero = true;
    string str, origen, destino;
    while (scanf("%d%d%d", &p, &f, &q) != EOF) {
        if (primero) {
            primero = false;
        }else
            printf(".\n");
        cin.ignore();
        map<string, int> planetas;
        vector<vector<edge> > grafo(p, vector<edge>());
        for (i=0; i<p; i++) {
            cin >> str;
            planetas[str] = i;
        }
        for (i=0; i < f; i++) {
            edge e;
            cin >> origen >> destino >> e.costo >> e.tiempo;
            e.destino = planetas[destino];
            grafo[planetas[origen]].push_back(e);
        }
        cin >> origen;
        si = planetas[origen];
        vector<int> cost(p, inf);
        vector<int> time(p, inf);
        vector<int> paradas(p, inf);
        priority_queue<nodo, vector<nodo>, decltype(comp)> cola(comp);
        nodo u;
        u.costo = 0;
        u.planeta = si;
        u.paradas = 0;
        u.tiempo = 0;
        cola.push(u);
        nodo v;

        cost[si] = 0;
        time[si] = 0;
        paradas[si] = 0;
        while (!cola.empty()) {
            v = cola.top();
            cola.pop();

            if (v.costo > cost[v.planeta]) { // segun halim este check es importante
                continue;
            }else if(v.costo == cost[v.planeta]){
                if (v.tiempo > time[v.planeta]) {
```

```
                continue;
            }
        }
        for (int i = 0; i < grafo[v.planeta].size(); i++) {
            edge e = grafo[v.planeta][i];
            if (cost[v.planeta] + e.costo < cost[e.destino]) {
                cost[e.destino] = cost[v.planeta] + e.costo;
                time[e.destino] = time[v.planeta] + e.tiempo;
                paradas[e.destino] = paradas[v.planeta] + 1;
                nodo w;
                w.planeta = e.destino;
                w.costo = cost[e.destino];
                w.tiempo = time[e.destino];
                w.paradas = v.paradas + 1;
                cola.push(w);
            }else if(cost[v.planeta] + e.costo == cost[e.destino]){
                if (time[v.planeta] + e.tiempo <= time[e.destino]) {
                    cost[e.destino] = cost[v.planeta] + e.costo;
                    time[e.destino] = time[v.planeta] + e.tiempo;
                    paradas[e.destino] = paradas[v.planeta] + 1;
                    nodo w;
                    w.planeta = e.destino;
                    w.costo = cost[e.destino];
                    w.tiempo = time[e.destino];
                    w.paradas = v.paradas + 1;
                    cola.push(w);
                }
            }
        }
    }
    int d;
    for (i=0; i<q; i++) {
        cin >> destino >> n;
        d = planetas[destino];
        if (cost[d] == inf) {
            printf("* *\n");
        }else if(paradas[d] <= n+1){
            printf("%d %d\n", cost[d], time[d]);
        }else{
            dijkstra(grafo, si, d, n);
        }
    }
}
return 0;}
```

FORMULARIO

FastIO

```
import java.io.*;
import java.util.StringTokenizer;
public class FastIO {
  public static void main(String [] args) throws IOException {
    Scanner2 sc = new Scanner2(System.in);
    int in = sc.nextInt();
    // If you don't use the \n the writer will print trash
    // The fast input should be enough
    OutputWriter out = new OutputWriter(System.out);
    out.print(in + "\n");
       out.print(String.format("%.6f",s) +"\n");//impresion con decimales

    out.close();
  }
}

class Scanner2{
  public BufferedReader reader;
  public StringTokenizer st;

  public Scanner2(InputStream stream){
    reader = new BufferedReader(new InputStreamReader(stream));
    st = null;
  }
  public String next(){
    while(st == null || !st.hasMoreTokens())
    {
      try
      {
        String line  = reader.readLine();
        if(line == null) return null;
        st =  new StringTokenizer(line);
      }
      catch (Exception e)
      {
        throw (new RuntimeException());
      }
    }
    return st.nextToken();
  }
  public int nextInt(){
    return Integer.parseInt(next());
  }
  public long nextLong(){
    return Long.parseLong(next());
  }
  public double nextDouble(){
    return Double.parseDouble(next());
  }
  public BigInteger nextBigInteger(){
        BigInteger hola = new BigInteger(next());
          return  hola;
        }
}
class OutputWriter{
  BufferedWriter writer;
```

```
  public OutputWriter(OutputStream stream){
    writer = new BufferedWriter(new OutputStreamWriter(stream));
  }
  public void print(int i) throws IOException {
    writer.write(i);
  }
  public void print(String s) throws IOException {
    writer.write(s);
  }
  public void print(char []c) throws IOException {
    writer.write(c);
  }
  public  void close() throws IOException {
    writer.close();
  }
}
```

Buffered Reader

```
BufferedReader entrada =new BufferedReader (new InputStreamReader(System.in));
int casos =0;
casos=Integer.parseInt(entrada.readLine());
String x = ""; x= entrada.readLine();
```

Tipos De Datos Primitivos

Tipo de variable	Bytes que ocupa	Rango de valores
boolean	**2**	true, false
byte	**1**	**-128 a 127**
short	**2**	**-32.768 a 32.767**
int	**4**	**-2.147.483.648 a 2.147.483.649**
long	**8**	**$-9 \cdot 10^{18}$ a $9 \cdot 10^{18}$**
double	**8**	**$-1{,}79 \cdot 10^{308}$ a $1{,}79 \cdot 10^{308}$**
float	**4**	**$-3{,}4 \cdot 10^{38}$ a $3{,}4 \cdot 10^{38}$**
char	**2**	**Caracteres (en Unicode)**

Lista De Secuencias De Escape:

- \n -----> Nueva Linea.
- \t -----> Tabulador.
- \r -----> Retroceso de Carro.
- \f -----> Comienzo de Pagina.
- \b -----> Borrado a la Izquierda.

- \\ -----> El carácter barra inversa (\ ).
- \' -----> El carácter prima simple (').
- \" -----> El carácter prima doble o bi-prima (").

Areas De Figuras Geometricas

```
triangulo = base * altura / 2
circulo = 3.14159 ( π) * r²
trapecio =(a + b * c) / 2
cuadrado = lado * lado
rectángulo = lado A * lado B
```

Interesante Manera De Resolver Operaciones Con Fracciones

Sum: (N1*D2 + N2*D1) / (D1*D2)
Subtraction: (N1*D2 - N2*D1) / (D1*D2)
Multiplication: (N1*N2) / (D1*D2)
Division: (N1/D1) / (N2/D2), that means (N1*D2)/(N2*D1)

Formula De Bhaskara

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

double r1 = ((-1*b)+(Math.sqrt((b*b)-4*a*c)))/(2*a);

double r2 = ((-1*b)-(Math.sqrt((b*b)-4*a*c)))/(2*a);

Suma de filas triangulo de pascal

La suma de los elementos de cualquier fila es el resultado de elevar 2 al número que define a esa fila. Así:

$2^0 = 1$
$2^1 = 1+1 = 2$
$2^2 = 1+2+1 = 4$
$2^3 = 1+3+3+1 = 8$
$2^4 = 1+4+6+4+1 = 16$

Números poligonales

$$n + \frac{n(n-1)b}{2}.$$

Tipos de números

Los **naturales** (0, 1, 2, 3,…)

Los **enteros** (…, -3, -2, -1, 0, 1, 2, 3,…)

Los **racionales** (todo número que puede ponerse en forma de fracción)

Los **irracionales** (todo número que no puede ponerse en forma de fracción)

Los **reales** (el conjunto de todos los anteriores)

Los **complejos**:

- **Número primo**: todo número natural mayor que 1 que cumple que sus únicos divisores son el 1 y el propio número. Ejemplos: 2, 3, 5,… Éste es el más grande que se conoce.
- **Número compuesto**: todo número natural mayor que 1 que no es primo. Ejemplos: 4, 6, 10, …
- **Número primo probable**: todo número del cual no se sabe si es primo o no pero que verifica alguna condición que verifican todos los números primos
- **Número pseudoprimo**: todo primo probable que acaba siendo compuesto.
- **Número perfecto**: todo número natural que es igual a la suma de sus divisores propios (es decir, todos sus divisores excepto el propio número). Por ejemplo, 6 es un número perfecto ya que sus divisores propios son 1, 2, y 3 y se cumple que 1+2+3=6. Los números 28, 496 y 8128 también son perfectos.
- **Número semiperfecto**: todo número natural que cumple que es igual a la suma de algunos de sus divisores propios. Por ejemplo, 18 es semiperfecto ya que sus divisores son 1, 2, 3, 6, 9 y se cumple que 3+6+9=18.
- **Número abundante**: todo número natural que cumple que la suma de sus divisores propios es mayor que el propio número. Por ejemplo, 12 es abundante ya que sus divisores son 1, 2, 3, 4 y 6 y se cumple que 1+2+3+4+6=16, que es mayor que el propio 12.
- **Número deficiente**: todo número natural que cumple que la suma de sus divisores propios es menor que el propio número. Por ejemplo, 16 es un número deficiente ya que sus divisores propios son 1, 2, 4 y 8 y se cumple que 1+2+4+8=15, que es menor que 16.
- **Números amigos**: parejas de números que cumplen que la suma de los divisores propios de cada uno de ellos da como resultado el otro número. Por ejemplo, 220 y 284 son números amigos.
- **Números sociables**: cumplen lo mismo que los números amigos pero en vez de ir en parejas van en grupos más grandes. La suma de los divisores del primer número da el segundo, la suma de los del segundo da el tercero, y así sucesivamente. La suma de los divisores del último da el primer número de la lista. Por ejemplo los números 12496, 14288, 15472, 14536 y 14264 son números sociables.
- **Número apocalíptico**: todo número natural n que cumple que 2^n contiene la secuencia 666. Por ejemplo, los números 157 y 192 son números apocalípticos.
- **Número ambicioso**: todo número que cumple que la secuencia que se forma al sumar sus divisores propios, después los divisores propios del resultado de esa suma, después los del número obtenido…acaba en un número perfecto. Por ejemplo, 25 es un*aspiring number* ya que sus divisores propios son 1 y 5 y se cumple que 1+5=6, que es un número perfecto.
- **Número curioso**: todo número natural n que cumple que n^2 tiene al propio n como última cifra. Por ejemplo, 25 y 36 son números curiosos.
- **Número de Carmichael**: todo número compuesto n que cumpla que $b^{n-1} \equiv 1 \pmod{(n)}$ (véase Congruencias) .para todo natural b que sea primo relativo con n. Por ejemplo, 561 y 1105 son números de Carmichael.
- **Cuadrado**: todo número natural que es el cuadrado de otro número natural. Por ejemplo, 9 es un cuadrado ya que $9=3^2$.
- **Cubo**: todo número natural que es el cubo de otro número natural. Por ejemplo, 125 es un cubo ya que $125=5^3$.

- **Número malvado**: todo número natural cuya expresión en base 2 (binaria) contiene un número par de unos. Por ejemplo, y 15 son números malvados ya que $12=1100_2$ y $15=1111_2$.
- **Número feliz**: todo número natural que cumple que si sumamos los cuadrados de sus dígitos y seguimos el proceso con los resultados obtenidos el resultado es 1. Por ejemplo, el número 203 es un número feliz ya que $2^2+0^2+3^2=13$; $12+32=10$; $1^2+0^2=1$.
- **Número infeliz**: todo número natural que no es un número feliz. Por ejemplo, el número 16 es un número infeliz.
- **Número hambriento**: el k-ésimo número hambriento es el más pequeño número natural *n* que cumple que 2^n contiene los primeros k dígitos de Pi. Los primeros números hambrientos son: 5, 17, 74, 144, 144, 2003,…
- **Número afortunado**: Tomemos la secuencia de todos los naturales a partir del 1: 1, 2, 3, 4, 5,… Tachemos los que aparecen en las posiciones pares. Queda: 1, 3, 5, 7, 9, 11, 13,… Como el segundo número que ha quedado es el 3 tachemos todos los que aparecen en las posiciones múltiplo de 3. Queda: 1, 3, 7, 9, 13,… Como el siguiente número que quedó es el 7 tachamos ahora todos los que aparecen en las posiciones múltiplos de 7. Así sucesivamente. Los números que sobreviven se denominan números afortunados.
- **Número de Fermat**: todo número natural de la forma $2^{2n}+1$ para algún *n*. Si ese número resulta ser primo se denomina **primo de Fermat**.
- **Número de Mersenne**: todo número natural de la forma 2^p-1, siendo *p* un número primo. Si ese número resulta ser primo se denomina **primo de Mersenne**.
- **Número narcisista**: todo número de k dígitos que cumple que es igual a la suma de las potencias k de sus dígitos es un número narceisita. Por ejemplo, 153 es un número narcisita de 3 dígitos, ya que $1^3+5^3+3^3=153$.
- **Número odioso**: todo número cuya expresión en base 2 (binaria) contiene un número impar de unos. Por ejemplo, $11=1011_2$ es un número odioso.
- **Número palindrómico**: número natural que se lee igual de derecha a izquierda y de izquierda a derecha. Por ejemplo 1348431.
- **Número poderoso**: todo número natural *n* que cumple que si un primo *p* es un divisor suyo entonces p^2 también lo es. Por ejemplo, el número 36 es un número poderoso ya que los únicos primos que son divisores suyos son 2 y 3 y se cumple que 4 y 9 también son divisores de 36.
- **Número oblongo**: todo número natural que cumple que es el producto de dos naturales consecutivos. Por ejemplo, los números 30, 42 y 56 son *pronic numbers*:
- **Número repunit**: todo número natural que está formado solamente por unos: 1, 11, 111, 1111,…
- **Número de Smith**: todo número natural que cumple que la suma de sus dígitos es igual a la suma de los dígitos de sus divisores primos contando su multiplicidad (es decir, el número de veces que aparece cada uno de ellos). Por ejemplo, el número 27 es un número de Smith ya que 2+7=9 y su único divisor primo es 3, que aparece tres veces, y por tanto 3+3+3=9.
- **Número libre de cuadrados**: todo número natural que cumple que en su descomposición en factores primos no aparece ningún factor repetido. Por ejemplo, el número 30 es un número libre de cuadrados.
- **Número ondulado**: todo número natural de la forma *ababab…*. Por ejemplo, los números 121 y 13131 son números ondulados.
- **Número intocable**: todo número natural que no es la suma de los divisores propios de ningún número. Por ejemplo, los número 52 y 88 son números intocables.
- **Número vampiro**: todo número natural para el cual exista una factorización formada por lo dígitos del propio número. Por ejemplo, el número 126 es un número vampiro ya que lo podemos factorizar así: $126=21\cdot6$.
- **Número raro**: todo número natural que es abundante pero que no es igual a la suma de ningún subconjunto de sus divisores propios. Por ejemplo, los número 70 y 836 son raros.

Abbreviations

A* : A Star

GCD : Greatest Common Divisor

AC : Accepted

APSP : All-Pairs Shortest Paths

AVL : Adelson-Velskii Landis (BST)

BNF : Backus Naur Form

BFS : Breadth First Search

BI : Big Integer

BIT : Binary Indexed Tree

BST : Binary Search Tree

CC : Coin Change

CCW : Counter ClockWise

CF : Cumulative Frequency

CH : Convex Hull

CS : Computer Science

DAG : Directed Acyclic Graph

DAT : Direct Addressing Table

D&C : Divide and Conquer

DFS : Depth First Search

DLS : Depth Limited Search

DP : Dynamic Programming

ED : Edit Distance

FT : Fenwick Tree

ICPC : Intl Collegiate Programming Contest

IDS : Iterative Deepening Search

IDA* : Iterative Deepening A Star

IOI : International Olympiad in Informatics

IPSC : Internet Problem Solving Contest

LA : Live Archive [20]

LCA : Lowest Common Ancestor

LCM : Least Common Multiple

LCP : Longest Common Prefix

LCS1 : Longest Common Subsequence

LCS2 : Longest Common Substring

LIS : Longest Increasing Subsequence

LRS : Longest Repeated Substring

MCBM : Max Cardinality Bip Matching

MCM : Matrix Chain Multiplication

MCMF : Min-Cost Max-Flow

MIS : Maximum Independent Set

MLE : Memory Limit Exceeded

MPC : Minimum Path Cover

MSSP : Multi-Sources Shortest Paths

MST : Minimum Spanning Tree

MWIS : Max Weighted Independent Set

Bracket Matching

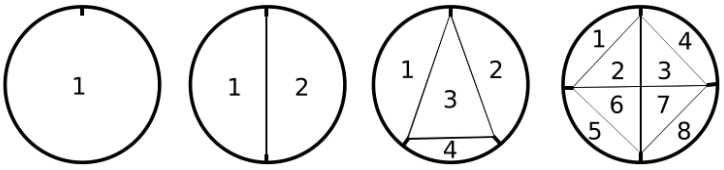
Uva Parentheses Balance 673

```
3                                   OUT
([])                                yes
(([()])))                           no
([()[]()])()                        yes
```

```
        int n = Integer.parseInt(in.nextLine());
        while (n-- > 0)
        {
            String exp = in.nextLine();
            Stack<Character> pilita = new Stack<Character>();
            for (int i = 0; i < exp.length(); i++)
            {
                char c = exp.charAt(i);
                if (!pilita.isEmpty() && pilita.peek() == '(' && c == ')')
                   pilita.pop();
                else if (!pilita.isEmpty() && pilita.peek() == '[' && c == ']')
                   pilita.pop();
                else
                   pilita.push(c);
            }
            if (pilita.isEmpty())
             System.out.println("Yes");
            else
             System.out.println("No");
      }
```

Moser's circle

uva 10213 how many pieces



Campus with 4 drinking fountains and the resulting regions numbered.

```
      int n  = in.nextInt();
                  //1+(N^4-6*N^3+23*N^2-18*N)/24
BigInteger a = new BigInteger(n +"").pow(4).subtract(new BigInteger(n +"").pow(3).multiply(new
BigInteger(6+""))).add(new BigInteger(23+""). multiply(new BigInteger(n +"").pow(2) ).subtract(new
BigInteger(18+""). multiply(new BigInteger(n +"")))).divide(new BigInteger(24+"")).add(new
BigInteger(1+"")) ;
                  System.out.println(a);
```

Complete bipartite graph

count the number of spanning tree in a complete bipartite graph $K_{n,m}$ is $m^{n-1} \times n^{m-1}$.

La Fórmula De Cayley

que establece que para cualquier entero positivo *n*, el número de árboles en ***n*** vértices etiquetados es n^{n-2}.

Equivalentemente, la fórmula cuenta el número de árboles de expansión de un grafo completo con vértices etiquetados.

```
                  int num = in.nextInt();
```

```
n = BigInteger.valueOf(num);
n = n.pow(Math.abs(num-2));
```

ESTRUCTURA DE DATOS

Estructura De Datos Modernas

Colas

```
LinkedList cola = new LinkedList();//declarar una cola

cola.offer(in.next());// ingresar datos a la cola

cola.peek();//ver el ultimo dato que se introdujo

cola.poll();//para quitar un dato de la cola

if (cola.peek()!=null) //verifica si la cola no esta vacia
{
}
```

Pilas

```
Stack pila = new Stack();//declarar una pila

pila.push(in.nextInt());// ingresar datos a la pila

pila.pop(); // quitar el ultimo dato

pila.peek(); //ver el ultimo dato que se introdujo

while(pila.empty()==false)

{//verificamos si la pila esta vacia, hace mientra no este vacia}
```

Lista Doblemente Enlazada

```
import java.util.Scanner;

public class ListaDoblementeEnlazada
{
	static class Nodo
	{
		int dato;
		Nodo puntero;
		Nodo anterior;
	}

	Nodo primero = null;
	Nodo ultimo = null;

	public void adicionarNodo(int valor)
	{
		Nodo cajaNueva = new Nodo();
		cajaNueva.dato =valor;
		cajaNueva.puntero = null;
		cajaNueva.anterior = null;
		if (this.primero == null)
		{
			this.primero = cajaNueva;
			this.ultimo = cajaNueva;
		}
		else
		{
			this.ultimo.puntero = cajaNueva;
			cajaNueva.anterior = this.ultimo;
			this.ultimo = cajaNueva;
		}
	}

	public void mostrarLista()
	{
		if (this.primero == null)
			System.out.println("La lista esta vacia");
		else
		{
			Nodo aux = new Nodo();
			aux = this.primero;
			while (aux!= null)
			{
				System.out.print(aux.dato);
				aux = aux.puntero;
			}
		}
	}

	public void mostrarListaInvertida()
	{
		if (this.primero == null)
			System.out.println("La lista esta vacia");
```

```
	else
	{
		Nodo aux = new Nodo();
		aux = this.ultimo;
		while (aux!= null)
		{
			System.out.print(aux.dato);
			aux = aux.anterior;
		}
	}
}

public static void main(String[] args)
{
	Scanner entrada = new Scanner(System.in);
	String num1,num2;
	ListaDoblementeEnlazada numero1 = new ListaDoblementeEnlazada();
	ListaDoblementeEnlazada numero2 = new ListaDoblementeEnlazada();
	num1 = entrada.next();
	long longitud = num1.length();
	num2 = entrada.next();
	longitud = longitud - num2.length();
	for (int i = 0; i < longitud; i++)
	{
		numero2.adicionarNodo(0);
	}
	for (int i = 0; i < num1.length(); i++)
	{
		numero1.adicionarNodo(Integer.parseInt(""+num1.charAt(i)));
	}
	for (int i = 0; i < num2.length(); i++)
	{
		numero2.adicionarNodo(Integer.parseInt(""+num2.charAt(i)));
	}
	numero1.mostrarLista();
	System.out.println("\n+");
	numero2.mostrarLista();
	ListaDoblementeEnlazada result = new ListaDoblementeEnlazada();
	Nodo aux1 = numero1.ultimo;
	Nodo aux2 = numero2.ultimo;
	boolean acumulador = false;
	while(aux1!=null&&aux2!=null)
	{
		int suma;
		if(acumulador)
			suma = aux1.dato+aux2.dato+1;
		else
			suma = aux1.dato+aux2.dato;
		if(suma>=10)
		{
			acumulador=true;
			suma = suma%10;
		}
		else
			acumulador = false;
```

```
                result.adicionarNodo(suma);
                aux1 = aux1.anterior;
                aux2 = aux2.anterior;
          }
          if(acumulador)
                result.adicionarNodo(1);
          System.out.println("\n---------------------------------------");
          result.mostrarListaInvertida();
     }
}
```

Lista Enlazada

```
public class ListaEnlazada
{
     static class Nodo
     {
          int dato;
          Nodo puntero;
     }

     Nodo primero = null;
     Nodo ultimo = null;

     public void adicionarNodo(int valor)
     {
          Nodo cajaNueva = new Nodo();
          cajaNueva.dato =valor;
          cajaNueva.puntero = null;
          if (this.primero == null)
          {
                this.primero = cajaNueva;
                this.ultimo = cajaNueva;
          }
          else
          {
                this.ultimo.puntero = cajaNueva;
                this.ultimo = cajaNueva;
          }
     }

     public void mostrarLista()
     {
          if (this.primero == null)
                System.out.println("La lista esta vacia");
          else
          {
                Nodo aux = new Nodo();
                aux = this.primero;
                int c=1;
                while (aux!= null)
                {
                     System.out.println("Valor elemento "+c+"= "+aux.dato);
                     aux = aux.puntero;
                     c++;
```

```
                }
            }
      }

      public static void main(String[] args)
      {
            ListaEnlazada lista1 = new ListaEnlazada();
            lista1.mostrarLista();
            System.out.println("-------------------------");
            lista1.adicionarNodo(123);
            lista1.mostrarLista();
            System.out.println("-------------------------");
            lista1.adicionarNodo(456);
            lista1.adicionarNodo(789);
            lista1.adicionarNodo(101112);
            lista1.mostrarLista();
            System.out.println("-------------------------");
      }
}
```

Arbol Binario

```
import java.util.Scanner;

public class arbolBinario
{
      public static class nodo
      {
            char dato;
            nodo HijoIzquierda;
            nodo HijoDerecha;
      }
      private nodo Padre;
      public void llenarArbol(nodo nodoGuia)
      {
            char respuesta;
            Scanner entrada = new Scanner(System.in);
            nodo nuevo = new nodo();
            System.out.println("Ingrese Nodo");
            nuevo.dato = entrada.next().charAt(0);
            nuevo.HijoDerecha = null;
            nuevo.HijoIzquierda = null;
            if (this.Padre == null)
            {
                  nodoGuia = nuevo;
                  this.Padre = nodoGuia;
            }
         System.out.print("El nodo "+nodoGuia.dato+" tiene hijo izquierdo?");
         respuesta = entrada.next().charAt(0);
         if (respuesta == 'S'| respuesta == 's')
         {
            nodo nuevoHijo= new nodo();
            nodoGuia.HijoIzquierda = nuevoHijo;
            this.llenarArbol(nodoGuia.HijoIzquierda);
```

```
      }
   System.out.print("El nodo "+nodoGuia.dato+" tiene hijo derecho?");
   respuesta = entrada.next().charAt(0);
   if (respuesta == 'S'| respuesta == 's')
   {
      nodo nuevoHijo= new nodo();
      nodoGuia.HijoDerecha = nuevoHijo;
      this.llenarArbol(nodoGuia.HijoDerecha);
      }
}

public void mostrarPreOrden(nodo nodoGuia)
{
      if(this.Padre == null)
            System.out.println("El arbol esta vacio");
      else
      {
            System.out.print(nodoGuia.dato);
            this.mostrarPreOrden(nodoGuia.HijoIzquierda);
            this.mostrarPreOrden(nodoGuia.HijoDerecha);
      }
}

public void mostrarInOrden(nodo nodoGuia)
{
      if(this.Padre == null)
            System.out.println("El arbol esta vacio");
      else
      {
            this.mostrarPreOrden(nodoGuia.HijoIzquierda);
            System.out.print(nodoGuia.dato);
            this.mostrarPreOrden(nodoGuia.HijoDerecha);
      }
}

public void mostrarPostOrden(nodo nodoGuia)
{
      if(this.Padre == null)
            System.out.println("El arbol esta vacio");
      else
      {
            this.mostrarPreOrden(nodoGuia.HijoIzquierda);
            this.mostrarPreOrden(nodoGuia.HijoDerecha);
            System.out.print(nodoGuia.dato);
      }
}

public static void main(String[] args)
{
      arbolBinario arbol1 = new arbolBinario();
      arbol1.llenarArbol(null);
      System.out.println("Recorrido en PreOrden");
      arbol1.mostrarPreOrden(arbol1.Padre);
      System.out.println("Recorrido en InOrden");
      arbol1.mostrarInOrden(arbol1.Padre);
```

```
            System.out.println("Recorrido en PostOrden");
            arbol1.mostrarPostOrden(arbol1.Padre);
      }
}
```

Unión Find

```
public class friends10608 {
      static int[] pset;
      static int[] rank;

      static void initSet(int N) {
            pset = new int[N];
            rank = new int[N];

            for (int i = 0; i < N; ++i)
            {
                  pset[i] = i;
                  rank[i] = 0;
            }
      }

      static int findSet(int i) {
            return (pset[i] == i) ? i : (pset[i] = findSet(pset[i]));
      }
      static void unionSet(int i, int j) {
            int pi = findSet(i);
            int pj = findSet(j);

            if (rank[pi] > rank[pj])
                  pset[pj] = pi;
            else
                  pset[pi] = pj;
            if (rank[pi] == rank[pj])
                  ++rank[pj];
      }

      public static void main(String[] args) throws IOException {
            Scanner in = new Scanner(System.in);
            int casos = in.nextInt();
            while (casos-- > 0) {

                  int n = in.nextInt();
                  int m = in.nextInt();

                  if (n == 0 && m == 0)
                        break;

                  initSet(n);

                  for (int i = 0; i < m; ++i)
                  {
                        int x = in.nextInt() - 1;
                        int y = in.nextInt() - 1;
                        unionSet(x, y);
                  }
```

```
            for (int i = 0; i < n; i++)
                  findSet(i);


      }
}}
```

GRAFOS

BFS

```java
import java.util.*;

public class BFS {
	static final int MAX = 500;
	static int ady[][] = new int[ MAX ][ MAX ];	//matriz de adyacencia
	static Scanner sc = new Scanner( System.in );
	static int V, prev[] = new int[ MAX ];


	public static void bfs(){
		int ini , fin , pasos = 0, max = 0, actual;
		boolean visitado[ ] = new boolean[ MAX ];
		Arrays.fill( visitado , false );


		System.out.println("Nodo raiz: ");
		ini = sc.nextInt();
		System.out.println("Nodo final: ");
		fin = sc.nextInt();

		prev[ ini ] = -1;

		Queue<Integer> Q = new LinkedList<Integer>();
		Q.add( ini );
		while( !Q.isEmpty() ){
			max = Math.max( max , Q.size() ); //ver memoria usada en cola
			actual = Q.remove();
			pasos++;

			if( actual == fin )break; //si se llego al destino

			visitado[ actual ] = true;

			for( int i = 0 ; i < V ; ++i ){ //vemos adyacentes a nodo actual
				int v = ady[ actual ][ i ];
				if( v != 0 && !visitado[ i ] ){ //no visitado agregamos a cola
				System.out.println( actual +" -> "+ i); //vemos recorrido de todo BFS
					prev[ i ] = actual;//para ver recorrido de nodo inicio a fin
					Q.add( i );
				}
			}
		}

		System.out.println("Memoria maxima: " + max );
		System.out.println("Nro Pasos: " + pasos);
		PrintRecorrido( ini , fin );

	}

	//Imprimimos recorrido para llegar de nodo ini a fin
	static void PrintRecorrido( int ini , int fin ){
		System.out.println("Recorrido de nodos para llegar de nodo "+ini+" a " +fin);
		List<Integer> path = new ArrayList<Integer>();
		for( ;; ){
			path.add( fin );
			if( prev[ fin ] == -1 )break;
```

```
                fin = prev[ fin ];
            }

            for( int i = path.size() - 1 , k = 0 ; i >= 0 ; --i ){
                if( k != 0 ) System.out.print( "->");
                System.out.print( path.get( i ) );
                k = 1;
            }
            System.out.println();
        }

        public static void main( String args[] ){
            int E , u , v;
            V = sc.nextInt();   //Numero de vertices
            E = sc.nextInt();   //Numero de aristas

            for( int i = 0 ; i < E ; ++i ){
                u = sc.nextInt(); v = sc.nextInt(); //enlace origen - destino
                ady[ u ][ v ] = 1;
            }
            bfs();
        }
}
```

Floyd Wharshall

```
import java.util.Scanner;
public class FloydWarshall
{
        public static void main(String[] args)
        {
            Scanner entrada = new Scanner(System.in);
            int nodo = entrada.nextInt();//cantidad de nodos
            int n = entrada.nextInt();//cantidad de vertices - conexiones
            int [][] matrizAdyacencia = new int[nodo][nodo];
            // inicio todas las posiciones de mi matriz en 10000
            for (int i = 0; i < matrizAdyacencia.length; i++)
            {
                for (int j = 0; j < matrizAdyacencia.length; j++) {
                    matrizAdyacencia[i][j]=10000;
                }
            }
            //cargamos inicio, fin y pesos hasta n (cant de conexiones)
            for (int i = 0; i < n; i++)
            {
                    int a = entrada.nextInt();//nodo inical
                    int b = entrada.nextInt();//nodo final
                    int peso = entrada.nextInt();//su peso
                    matrizAdyacencia[a-1][b-1]=peso;
            }
            //mostramos matriz sin recorrer en el algoritmo
            for (int i = 0; i < matrizAdyacencia.length; i++)
            {
                for (int j = 0; j < matrizAdyacencia.length; j++)
                {
                    System.out.print(matrizAdyacencia[i][j]+"\t");
                }
                System.out.println();
```

```
            }
            System.out.println("----------------------hola :)-------------------------");
            //recorremos nuestra matriz en el algoritmo de floyd
            int[][]nueva = floyd(matrizAdyacencia,nodo);
            //mostramos matriz luego de recorrer en floyd
            for (int i = 0; i < nueva.length; i++)
            {
                  for (int j = 0; j < nueva.length; j++)
                  {
                        System.out.print(nueva[i][j]+"\t");
                  }
                  System.out.println();
            }
      }
      //algoritmo de floyd-warshall
      static int[][] floyd (int w[][],int n)
      {
            for (int k = 0; k < n; k++)
            {
                  for (int i = 0; i < n; i++)
                  {
                        for (int j = 0; j < n; j++)
                        {
                              if(i!=j)
                  w[i][j]=Math.min(w[i][j],w[i][k]+w[k][j]);//anadimos ruta minima en la posision
                        }
                  }
            }
            return w;
      }
}
```

Segment Tree

```
#include <iostream>
#include <sstream>
#include <utility>
#include <cstdlib>
#include <cstdio>
#include <cctype>
#include <cmath>
#include <functional>
#include <algorithm>
#include <numeric>
#include <string>
#include <vector>
#include <queue>
#include <stack>
#include <list>
#include <map>
#include <set>
#include <stdio.h>
#include <string.h>
using namespace std;
typedef long long   LL;

int T[100000];
int v[100000];
const int N = 6;

void update(int pos, int val, int node = 1, int ini = 0, int end = N-1) {
    if(ini == end) {
```

```
        v[ini] = val;
        T[node] = val;
        return;
    }
    int mid = (ini+end)/2;
    int left = node*2;
    int right = left+1;
    if(ini <= pos && pos <= mid) {
        update(pos, val, left, ini, mid);
        T[node] = T[left] + T[right];
    }
    else {
        update(pos, val, right, mid+1, end);
        T[node] = T[left] + T[right];
    }
}

int querySum(int a, int b, int node = 1, int ini = 0, int end = N-1) {
    if(a == ini && b == end) {
        return T[node];
    }
    int mid = (ini+end)/2;
    int left = node*2;
    int right = left+1;
    int sum = 0;
    if(b <= mid)
        sum += querySum(a, b, left, ini, mid);
    else if(a > mid)
        sum += querySum(a, b, right, mid+1, end);
    else {
        sum += querySum(a, mid, left, ini, mid);
        sum += querySum(mid+1, b, right, mid+1, end);
    }
    return sum;
}

int main() {
    update(5, 5); // pos, val
    cout << querySum(1, 2) << endl;
    update(2, -4); // pos, val
    cout << querySum(1, 2) << endl;
    update(3, 8); // pos, val
    cout << querySum(1, 2) << endl;
    update(1, 21); // pos, val
    cout << querySum(1, 2) << endl;
    update(4, -1); // pos, val
    cout << querySum(1, 2) << endl;
}
```

Binary tree

```
#include <iostream>
#include <sstream>
#include <utility>
#include <cstdlib>
#include <cstdio>
#include <cctype>
#include <cmath>
#include <functional>
#include <algorithm>
#include <numeric>
#include <string>
#include <vector>
#include <queue>
#include <stack>
#include <list>
#include <map>
```

```
#include <set>
#include <stdio.h>
#include <string.h>
using namespace std;
typedef long long   LL;

int T[10000];
int Tson0[10000];
int Tson1[10000];
const int rootT = 0;
int it;

void insert(int val, int node = 1) {
    if(T[node] == -1) {
        T[node] = val;
        Tson0[node] = ++it;
        Tson1[node] = ++it;
        return;
    }
    if(val <= T[node]) {
        insert(val, Tson0[node]);
    }
    else {
        insert(val, Tson1[node]);
    }
}

int querySum(int q, int node = 1) {
    int sum = 0;
    if(T[node] == -1) return 0;
    if(T[node] < q) {
        sum += T[node];
    }
    if(Tson0[node] != -1) {
        sum += querySum(q, Tson0[node]);
    }
    if(Tson1[node] != -1 && T[node] < q) {
        sum += querySum(q, Tson1[node]);
    }
    return sum;
}

int main() {
    memset(T, -1, sizeof(T));
    memset(Tson0, -1, sizeof(Tson0));
    memset(Tson1, -1, sizeof(Tson1));
    T[rootT] = -1;
    it = 1;
    insert(5);
    insert(12);
    insert(1);
    insert(3);
    insert(9);
    cout << querySum(7) << endl;
}
```

Binary Interval tree (bit)

```
#include <iostream>
#include <sstream>
#include <utility>
#include <cstdlib>
#include <cstdio>
#include <cctype>
#include <cmath>
#include <functional>
#include <algorithm>
```

```
#include <numeric>
#include <string>
#include <vector>
#include <queue>
#include <stack>
#include <list>
#include <map>
#include <set>
#include <stdio.h>
#include <string.h>
using namespace std;
typedef long long   LL;

const int MAX_N = 200005;
int n;
int t[MAX_N], v[MAX_N];

int sum(int x) {
    int result = 0;
    for(int i = x; i >= 0; i = (i & (i+1)) - 1)
        result += t[i];
    return result;
}

void inc(int x, int delta) {
    v[x] += delta;
    for(int i = x; i <= n; i = (i | (i+1)))
        t[i] += delta;
}

int main() {
     n = 10;
     inc(2, 4);
     inc(4, -3);
     inc(5, 2);
     int x = 3;
     int y = 5;
     cout << sum(y) - sum(x-1) << endl; // x, y
}
```