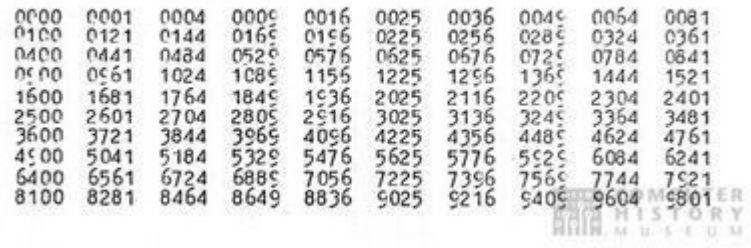
Comenzado el	jueves, 24 de noviembre de 2022, 16:28
Estado	Finalizado
Finalizado en	lunes, 28 de noviembre de 2022, 19:03
Tiempo empleado	4 días 2 horas
Puntos	8,00/8,00
Calificación	**10,00** de 10,00 (**100**%)

Pregunta **1**

Correcta

Se puntúa 1,00 sobre 1,00

El 6 de mayo de 1949, Maurice Wilkes creó el primer programa almacenado en una máquina lo suficientemente potente para realizar cálculos matemáticos de forma práctica. Esta máquina, llamada EDSAC, calculó e imprimió la tabla de cuadrados de los primeros 100 números (0-99) mostrada a continuación:



Haz un programa que genere la misma salida que el programa de Wilkes.

Fíjate que cada resultado está formateado a cuatro dígitos y la separación entre columnas es de dos espacios.

Respuesta: (sistema de penalización: 0 %)

```
1  public class Prog {
2      public static void main(String[] args){
3          for(int i = 0; i <= 99; i++){
4              System.out.print(String.format("%04d  ", (i*i)));
5              if((i+1) % 10 == 0){
6                  System.out.println();
7              }
8          }
9      }
10 }
```

	Esperado	Se obtuvo	
✔	0000 0001 0004 0009 0016 0025 0036 0049 0064 0081 0100 0121 0144 0169 0196 0225 0256 0289 0324 0361 0400 0441 0484 0529 0576 0625 0676 0729 0784 0841 0900 0961 1024 1089 1156 1225 1296 1369 1444 1521 1600 1681 1764 1849 1936 2025 2116 2209 2304 2401 2500 2601 2704 2809 2916 3025 3136 3249 3364 3481 3600 3721 3844 3969 4096 4225 4356 4489 4624 4761 4900 5041 5184 5329 5476 5625 5776 5929 6084 6241 6400 6561 6724 6889 7056 7225 7396 7569 7744 7921 8100 8281 8464 8649 8836 9025 9216 9409 9604 9801	0000 0001 0004 0009 0016 0025 0036 0049 0064 0081 0100 0121 0144 0169 0196 0225 0256 0289 0324 0361 0400 0441 0484 0529 0576 0625 0676 0729 0784 0841 0900 0961 1024 1089 1156 1225 1296 1369 1444 1521 1600 1681 1764 1849 1936 2025 2116 2209 2304 2401 2500 2601 2704 2809 2916 3025 3136 3249 3364 3481 3600 3721 3844 3969 4096 4225 4356 4489 4624 4761 4900 5041 5184 5329 5476 5625 5776 5929 6084 6241 6400 6561 6724 6889 7056 7225 7396 7569 7744 7921 8100 8281 8464 8649 8836 9025 9216 9409 9604 9801	✔

Todas las pruebas superadas. ✔

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **2**

Correcta

Se puntúa 1,00 sobre 1,00

Se pide en este caso escribir un método que imita al indexOf de forma que se le pasa un string y un caracter y devuelve el índice de la primera ocurrencia de dicho caracter o bien -1 si el caracter no tiene está en el string.

Evidentemente se pretende que resuelvas el problema procesando el String en un bucle, ¡sin usar el método indexOf()!

El método se llamará myIndexOf() y su funcionamiento se ejemplifica a continuación. El método debes definirlo como static.

Por ejemplo:

Test	Resultado
`System.out.println(myIndexOf("hola a todos",'s'));`	11

Respuesta: (sistema de penalización: 0 %)

```
1  ▾    private static int myIndexOf(String st, char c) {
2  ▾        for(int i =0; i < st.length(); i++){
3  ▾                if(st.charAt(i) == c){
4                       return i;
5                   }
6           }
7           return -1;
8       }
```

	Test	Esperado	Se obtuvo	
✔	`System.out.println(myIndexOf("hola a todos",'s'));`	11	11	✔
✔	`System.out.println(myIndexOf("hola a todos",' '));`	4	4	✔
✔	`System.out.println(myIndexOf("hola a todos",'H'));`	-1	-1	✔
✔	`System.out.println(myIndexOf("hola a todos",'a'));`	3	3	✔
✔	`System.out.println(myIndexOf("DAM",'M'));`	2	2	✔
✔	`System.out.println(myIndexOf("1234",'4'));`	3	3	✔

Todas las pruebas superadas. ✔

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **3**

Correcta

Se puntúa 1,00 sobre 1,00

Radares de tramo

basado en

https://www.aceptaelreto.com/pub/problems/v001/12/st/statements/Spanish/index.html

La Dirección Particular de Tráfico (DPT) está empeñada en hacer que los conductores respeten los límites de velocidad. Sin entrar en si es por razones de seguridad, por ahorrar combustible, o con un mero afán recaudatorio, ahora sabemos que además de los radares fijos tradicionales, están poniendo en funcionamiento los radares de tramo.

Desde un punto de vista formal, estos radares se basan en el teorema de Lagrange (también llamado de *valor medio* o de Bonnet-Lagrange), que dice que si tienes una función continua en un intervalo cerrado y derivable en el intervalo abierto, entonces algún punto de ese intervalo abierto tendrá derivada instantánea igual a la pendiente media de la curva en el intervalo cerrado.

Aunque asuste a primera vista, la repercusión es sencilla: si hacemos un viaje desde Madrid a Zaragoza y nuestra velocidad media es de 111Km/h, *forzosamente* en algún punto del camino, nuestra velocidad ha sido de 111Km/h.

Los radares de tramo consisten en colocar dos cámaras en dos puntos alejados de una carretera para poder comprobar cuánto tiempo ha tardado el coche en recorrer ese tramo. Si la velocidad media supera la velocidad máxima permitida, gracias al teorema anterior podremos saber (aunque no le hayamos visto) que en algún punto del trayecto ha superado esa velocidad. Por ejemplo, si colocamos las cámaras separadas 10Km en un tramo cuya velocidad está limitada a 110Km/h, y un coche tarda 5 minutos en ser visto por la segunda cámara, sabremos que su velocidad media ha sido de 120Km/h, y por tanto en algún sitio ha superado el límite de velocidad aunque al pasar por debajo de las dos cámaras el coche fuera a 80Km/h.

ENTRADA:

La entrada debe responder a una serie de casos. Cada caso de prueba consistirá en tres números: el primero será la distancia (en metros) que separan las dos cámaras, el segundo indicará la velocidad máxima permitida en todo ese tramo

(en Km/h) y el tercer y último número indicará el número de segundos que ha tardado un coche en recorrer el tramo. Todos esos números serán enteros.

La entrada terminará cuando todos los números sean cero.

SALIDA:

Para cada caso de prueba, el programa generará una línea, indicando si el coche debe ser multado o no. En concreto, indicará "OK" si el coche no superó la velocidad máxima, indicará "MULTA" si se superó esa velocidad en menos de un 20% de la velocidad máxima permitida, y "PUNTOS" si la velocidad fue superada en un 20% o más de esa velocidad; en ese caso además de la multa se le quitarán puntos del carnet.

El sistema de radar puede fallar y registrar entradas incorrectas. Se consideran incorrectos los valores de entrada menores o igual a 0. En esos casos, el sistema mostrará la cadena "ERROR".

Nota: Utiliza aritmética punto flotante con double para hacer las operaciones.

Por ejemplo:

Entrada	Resultado
9165 110 300 9165 110 299 0 0 0	OK MULTA

Respuesta: (sistema de penalización: 0 %)

```
1  import java.util.Scanner;
2
3  public class Prog {
4      // 9165 110 300 -> 109.98 km/h media
5      // 9165 110 299 -> 110.35 km/h media
6      public static void main(String[] args) {
7          Scanner sc = new Scanner(System.in);
8          int distMetros;
9          int velMax;
10         int segs;
```

	Entrada	Esperado	Se obtuvo	
✔	9165 110 300 9165 110 299 0 0 0	OK MULTA	OK MULTA	✔
✔	12000 100 433 12000 100 431 12000 100 359 -1000 -50 -100 0 15 0 0 0 0	OK MULTA PUNTOS ERROR ERROR	OK MULTA PUNTOS ERROR ERROR	✔

Todas las pruebas superadas. ✔

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **4**

Correcta

Se puntúa 1,00 sobre 1,00

La sucesión de Fibonacci es la sucesión de números:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Cada número se calcula sumando los dos anteriores a él.

- el 2 se obtiene sumando los dos números anteriores (1+1),
- el 5 es (2+3),
- etc.

A la sucesión de Fibonacci se le atribuyen poderes casi "esotéricos". Si tienes tiempo te puede resultar agradable y curioso leer el siguiente artículo

http://www.neoteo.com/la-sucesion-de-fibonacci-en-la-naturaleza/

La serie de fibonacci tiene la siguiente definición matemática

$$f_0 = 0$$

$$f_1 = 1$$

$$f_n = f_{n-1} + f_{n-2}$$

que es una definición de naturaleza recursiva.

SE PIDE: escribir un método estático en Java llamado fibonacci() si le pasa un entero n>=0 que representa una posición en la serie de fibonacci devuelve el valor correspondiente a dicha posición.

Aunque la solución de Fibonacci se puede obtener escribiendo un bucle, se pide en este caso que resuelvas el ejercicio utilizando recursividad.

Por ejemplo:

Test	Resultado
System.out.println(fibonacci(1));	1
System.out.println(fibonacci(2));	1
System.out.println(fibonacci(13));	233

Respuesta: (sistema de penalización: 0 %)

```
  private static int fibonacci(int pos) {
       if (pos <= 1) {
           return pos;
       } else {
           return fibonacci(pos - 1) + fibonacci(pos - 2);
       }
   }
```

	Test	Esperado	Se obtuvo	
✔	`System.out.println(fibonacci(1));`	1	1	✔
✔	`System.out.println(fibonacci(2));`	1	1	✔
✔	`System.out.println(fibonacci(13));`	233	233	✔
✔	`System.out.println(fibonacci(20));`	6765	6765	✔

Todas las pruebas superadas. ✔

Correcta

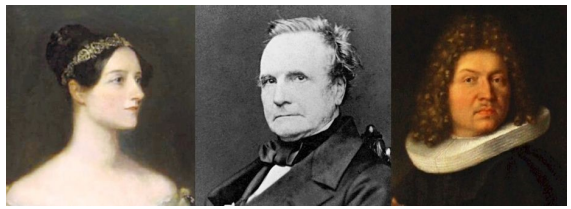
Puntos para este envío: 1,00/1,00.

Pregunta **5**

Correcta

Se puntúa 1,00 sobre 1,00

3.



En 1815 nacía Ada Byron, conocida después como Ada Lovelace, destinada a convertirse en la primera programadora de la historia. Amiga de Charles Babbage, siguió con interés los trabajos de éste relacionados con su *máquina analítica*, que hoy se considera un hito en la historia de la computación.

Con la ayuda de Babbage, estudió y mejoró algunas de sus ideas, centrándose en lo que hoy llamaríamos *software*, mientras que Babbage se preocupaba principalmente por el *hardware*. Fue Ada quien describió el primer *algoritmo* pensado específicamente para ser ejecutado por un "ordenador", por lo que se la reconoce como la primera programadora.

El objetivo del algoritmo era el cálculo de los números de Bernouilli. Esta es una secuencia de números racionales que tienen conexiones muy interesantes con teoría de números. Su cálculo es complejo, por lo que a pesar de que han pasado muchos años desde que Ada "programó" cómo obtenerlos, nos conformaremos con uno de sus usos, el cálculo de la fórmula de Faulhaber, o suma de los n primeros números elevados a un valor constante p:

$$1^p + 2^p + 3^p + \ldots + n^p$$

Crea un programa que lea de la entrada estándar múltiples casos de prueba, cada uno en una línea. Un caso de prueba se compondrá de dos números, n y p, entre 1 y 10 ambos incluídos.

El programa finalizará cuando $n = p = 0$

Por ejemplo:

Entrada	Resultado

Entrada	Resultado
1 1	1
2 2	5
3 3	36
4 3	100
0 0	

Respuesta: (sistema de penalización: 0 %)

```
import java.util.Scanner;

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

        int n = sc.nextInt();
        int p = sc.nextInt();
        while (n != 0 && p != 0) {
            if (n >= 1 && p >= 1 && n <= 10 && p <= 10) {
                long result = 1;
                for (int i = 2; i <= n; i++) {
                    long pot = 1;
                    for (int j = 1; j <= p; j++) {
                        pot *= i;
                    }
                    result += pot;
                }
                System.out.println(result);
            }

            n = sc.nextInt();
```

	Entrada	Esperado	Se obtuvo	
✔	1 1 2 2 3 3 4 3 0 0	1 5 36 100	1 5 36 100	✔
✔	10 10 0 0	14914341925	14914341925	✔
✔	5 11 -2 8 0 0			✔

Todas las pruebas superadas. ✔

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **6**

Correcta

Se puntúa 1,00 sobre 1,00

Crea un **método estático** denominado **diamante(n, c)** que dibuje un diamante de la altura **n** y relleno **c**, siempre y cuando la altura sea impar y comprendida entre 1 y 33. Un diamante de tamaño 1 es simplemente un caracter c. Para diamantes con altura <=0 o bien altura par simplemente no se imprime nada.

Por ejemplo:

Test	Resultado
diamante(5, '#')	# ### ##### ### #

Respuesta: (sistema de penalización: 0 %)

```
1  private static void diamante(int n, char c) {
2      // impar y comprendida entre 1 y 33.
3      if (n >= 1 && n <= 33 && n % 2 != 0) {
4          int i, limit;
5
6          for (i = 0; i <= n / 2; i++) {
7              limit = (n / 2) - i;
8              for (int j = 0; j < limit; j++) {
9                  System.out.print(" ");
10             }
11
12             limit = (i * 2) + 1;
13             for (int j = 0; j < limit; j++) {
14                 System.out.print(c);
15             }
16             System.out.println();
```

	Test	Esperado	Se obtuvo	

✔	`diamante(5, '#')`	`  #`
		`  ###`
		`#####`
		`  ###`
		`  #`
			`  #`
			`  ###`
			`#####`
			`  ###`
			`  #`
				✔

✔	`diamante(8, '*')`			✔
✔	`diamante(1, 'B')`	`B`	`B`	✔

✔	`diamante(7,'\u2661');`	`   ♡`
		`  ♡♡♡`
		`♡♡♡♡♡`
		`♡♡♡♡♡♡♡`
		`♡♡♡♡♡`
		`♡♡♡`
		`♡`
			`   ♡`
			`  ♡♡♡`
			`♡♡♡♡♡`
			`♡♡♡♡♡♡♡`
			`♡♡♡♡♡`
			`♡♡♡`
			`♡`
				✔

Todas las pruebas superadas. ✔

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **7**

Correcta

Se puntúa 1,00 sobre 1,00

Miker Chiménez ve cosas escondidas en cualquier lado. Una mancha de humedad en la pared se le antoja la cara de la anterior propietaria de una casa; el sonido del viento le parece un susurro venido del más allá; una nube con forma peculiar le convence de la existencia de vida extraterrestre...

Ahora le ha dado por ver mensajes ocultos en cualquier sitio. Por poner un ejemplo, si lee el siguiente titular: "*El presidente del Gobierno se somete esta noche al escrutinio de varios periodistas en Televisión Española*.", se las ingenia para leer un "te odio" oculto que le mantiene en vela toda la noche:

*El presiden**TE** del Gobierno se s**O**mete esta noche al escrutinio **D**e var**I**os peri**O**distas en Televisión Española.*

Crea un programa para ayudar a Miker Chiménez a automatizar la búsqueda de estos mensajes.

El programa leerá inicialmente un entero que le indicará cuántos casos de prueba vendrán a continuación. Cada uno de estos casos de prueba estará formado por dos líneas; la primera indica el ***titular*** donde buscar un mensaje oculto y la segunda indica el ***mensaje*** a buscar. Ten en cuenta que no hace falta distinguir entre mayúsculas y minúsculas y que los espacios del mensaje oculto no son relevantes, es decir, no hace falta que existan en el mensaje original, pero sí deben aparecer el resto de caracteres (signos de puntuación, comillas, etc.).

La entrada contendrá únicamente letras del alfabeto inglés, por lo que no aparecerán vocales con tilde. Además, podrían aparecer múltiples espacios consecutivos.

Por ejemplo:

Entrada	Resultado

Entrada	Resultado
4 ...dente ...somete ... de varios periodistas ... te odio. Teo dijo "si". te odio. Y adios, que ya viene el alba. te odio. Teo subio al podio. te odio.	SI SI NO SI

Respuesta: (sistema de penalización: 0 %)

```
1
2  import java.util.Scanner;
3
4  public class Prog {
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7          int casos = Integer.parseInt(sc.nextLine());
8
9          // CASOS A PROBAR
10         while (casos-- > 0) {
11             // INPUT
12             // pasamos a minúsculas la frase original, para comparar sin problemas
13             String frase = sc.nextLine().toLowerCase();
14
15             // pasamos a minúsculas la frase original, y quitamos espacios
16             String find = sc.nextLine().toLowerCase().replace(" ", "").replace("\t", "");
17
18             String solution = "";
19             int limit = 0;
20
21             // recorremos la frase
22             for (int i = 0; i < find.length(); i++) {
```

	Entrada	Esperado	Se obtuvo	
✔	4 ...dente ...somete ... de varios periodistas ... te odio. Teo dijo "si". te odio. Y adios, que ya viene el alba. te odio. Teo subio al podio. te odio.	SI SI NO SI	SI SI NO SI	✔
✔	1 donde las dan las toman DAM	SI	SI	✔
✔	1 tt ttt	NO	NO	✔
✔	0			✔

Todas las pruebas superadas. ✔

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **8**

Correcta

Se puntúa 1,00 sobre 1,00

Escribe una clase Potencia que me permita la ejecución descrita en el ejemplo.

Para calcular el número real asociado a una potencia deberás basarte en multiplicaciones sucesivas, no utilices en este caso el método pow(). El exponente siempre lo vamos a utilizar entero o el algoritmo para calcular la potencia se complica muchísimo. La base puede ser real (double en java).

La suma de potencias sólo caso fácil, ambas de igual base.

Para imprimir $base^{exponente}$ utilizamos la notación base^exponente

Por ejemplo:

Test	Resultado

`Potencia p= new Potencia(5,2);`
`System.out.println(p);`
`System.out.println(p.toReal());`
`p= new Potencia(2.5,2);`
`System.out.println(p);`
`System.out.println(p.toReal());`
	`5.0^2`
	`25.0`
	`2.5^2`
	`6.25`

`Potencia p= new Potencia(2,3);`
`System.out.println(p);`
`System.out.println(p+"="+p.toReal());`
`p= new Potencia(2,-3);`
`System.out.println(p+"="+p.toReal());`
`p= new Potencia(-2,-3);`
`System.out.println(p+"="+p.toReal());`
`p= new Potencia(2.5,2);`
`System.out.println(p+"="+p.toReal());`
`Potencia p1= new Potencia(2,3);`
`Potencia p2= new Potencia(2,2);`
`System.out.println("si multiplico con distinta base:`
`"+p1.multiplicarConIgualBase(p));`
`System.out.println(p1.multiplicarConIgualBase(p2));`
	`2.0^3`
	`2.0^3=8.0`
	`2.0^-3=0.125`
	`-2.0^-3=-0.125`
	`2.5^2=6.25`
	`si multiplico con distinta base:`
	`null`
	`2.0^5`

Respuesta: (sistema de penalización: 0 %)

```
1  public class Potencia {
2      double base;
3      int exponente;
4
5      public Potencia(double base, int exponente) {
6          this.base = base;
7          this.exponente = exponente;
8      }
9
10     public String toString() {
11         return this.base + "^" + this.exponente;
12     }
13
14     public double toReal() {
15         double result = 1;
16         double expPos = Math.abs(this.exponente);
17         for (int i = 1; i <= expPos; i++) {
18             result *= this.base;
19         }
20         if (this.exponente == 0) {
21             return 1;
22         } else if (this.exponente < 0) {
```

	Test	Esperado	Se obtuvo	
✔	Potencia p= new Potencia(5,2); System.out.println(p); System.out.println(p.toReal()); p= new Potencia(2.5,2); System.out.println(p); System.out.println(p.toReal());	5.0^2 25.0 2.5^2 6.25	5.0^2 25.0 2.5^2 6.25	✔

	Test	Esperado	Se obtuvo	

✔	`Potencia p= new Potencia(2,3);`
	`System.out.println(p);`
	`System.out.println(p+"="+p.toReal());`
	`p= new Potencia(2,-3);`
	`System.out.println(p+"="+p.toReal());`
	`p= new Potencia(-2,-3);`
	`System.out.println(p+"="+p.toReal());`
	`p= new Potencia(2.5,2);`
	`System.out.println(p+"="+p.toReal());`
	`Potencia p1= new Potencia(2,3);`
	`Potencia p2= new Potencia(2,2);`
	`System.out.println("si multiplico con distinta base: "+p1.multiplicarConIgualBase(p));`
	`System.out.println(p1.multiplicarConIgualBase(p2));`
		`2.0^3`
		`2.0^3=8.0`
		`2.0^-3=0.125`
		`-2.0^-3=-0.125`
		`2.5^2=6.25`
		`si multiplico con distinta base: null`
		`2.0^5`
			`2.0^3`
			`2.0^3=8.0`
			`2.0^-3=0.125`
			`-2.0^-3=-0.125`
			`2.5^2=6.25`
			`si multiplico con distinta base: null`
			`2.0^5`
				✔

✔	`Potencia p= new Potencia(7,0);` `System.out.println(p+"="+p.toReal());`	`7.0^0=1.0`	`7.0^0=1.0`	✔

✔	`Potencia p1= new Potencia(3,3);`
	`Potencia p2= new Potencia(3,1);`
	`System.out.println(p1.multiplicarConIgualBase(p2));`
		`3.0^4`	`3.0^4`	✔

Todas las pruebas superadas. ✔

Correcta

Puntos para este envío: 1,00/1,00.

◀ COPIA NO PUNTUABLE DE Tarea 1: MyLittleDecimal

Ir a...

1. Registrarse en acepta el reto ▶