Programa-Me 2011
Problemas

Ejercicios realizados por

Universidad Complutense
de Madrid

I.E.S. Antonio de Nebrija
(Móstoles)

Realizado en

IES Antonio de Nebrija. Móstoles

Índice

Autores de los problemas:

- Marco Antonio Gómez Martín (Universidad Complutense de Madrid)
- Pedro Pablo Gómez Martín (Universidad Complutense de Madrid)
- Patricia Díaz García (I.E.S. Antonio de Nebrija - Móstoles)

A
Códigos de barras

En el lejano 1952, tres norteamericanos patentaron lo que terminó llamándose *código de barras*. Consiste en una técnica para representar números (y, en menos ocasiones, letras) mediante una serie de líneas verticales paralelas, con diferentes grosores y separaciones entre ellas. Si bien el primer uso sirvió para identificar de manera automática los vagones de un ferrocarril, hoy los códigos de barras se utilizan en infinidad de lugares, siendo la catalogación de productos la más habitual.

La manera concreta de codificar mediante barras los números y las letras puede ser muy variada, lo que ha llevado a la aparición de diferentes estándares. De todos ellos, el EAN (*European Article Number*) resulta ser el más extendido. De éste, hay principalmente dos formatos, que se diferencian en el ancho. Existe así el llamado EAN-8, que codifica 8 números, y el EAN-13, que, naturalmente, codifica 13.

(a) EAN-8 (b) EAN-13

Figura 1: Códigos de barras EAN

El último dígito del código se utiliza para detección de errores, y se calcula a partir de los demás. Para eso:

- Empezando *por la derecha* (sin contar el dígito de control que se está calculando), se suman los dígitos individuales, multiplicados por un factor:
  - Los dígitos en posiciones impares (empezando a contar por la derecha saltándonos el de control) se multiplican por 3.
  - Los dígitos en posiciones pares se multiplican por 1.

  Por ejemplo, para el código EAN-8 de la figura la operación a realizar es:

$$2 \cdot 3 + 5 \cdot 1 + 9 \cdot 3 + 3 \cdot 1 + 8 \cdot 3 + 5 \cdot 1 + 6 \cdot 3 = 88$$

- El dígito de comprobación es el número que hay que sumar al resultado anterior para llegar a un valor múltiplo de 10. En el ejemplo de EAN-8, para llegar al múltiplo de 10 más cercano por encima del número 88 hay que sumar 2 (y llegar al 90). Ten en cuenta que si la suma resulta ser ya múltiplo de 10, el dígito de control será 0.

En EAN-13, los primeros dígitos se usan además para identificar al país. A continuación se indica una tabla (parcial) de esos códigos de país.

Código	País
0	EEUU
380	Bulgaria
50	Inglaterra

Código	País
539	Irlanda
560	Portugal
70	Noruega

Código	País
759	Venezuela
850	Cuba
890	India

Entrada

La entrada estará formada por una serie de casos de prueba. Cada uno contendrá una sucesión de números pertenecientes a un código de barras EAN-8 o EAN-13, incluyendo el dígito de control. Si el número de dígitos es inferior a 8, se asumirá que es un código EAN-8; si es superior a 8 pero inferior a 13, se asumirá EAN-13. En ambos casos, se completarán el resto de dígitos colocando ceros a la izquierda.

La entrada finalizará con el código especial 0.

Salida

Para cada caso de prueba, el programa indicará si el dígito de control es correcto o no. Si lo es, escribirá `SI`. En otro caso, escribirá `NO`.

Si el código de barras es EAN-13 y correcto, el programa escribirá además el país al que pertenece utilizando la tabla anterior (separado por un espacio). Si el código no aparece en la tabla, el programa mostrará `Desconocido`. Ten cuidado al escribir los países; deberás respetar el uso de mayúsculas y minúsculas de la tabla.

Entrada de ejemplo

```
65839522
65839521
8414533043847
5029365779425
5129365779425
0
```

Salida de ejemplo

```
SI
NO
SI Desconocido
SI Inglaterra
NO
```

Fuente

http://en.wikipedia.org/wiki/European_Article_Number

B
Aproximación de Gauss

Si hay un tipo de números importante y que es base de las matemáticas ese es el de los *números primos*.

Se dice que un número es *primo* cuando sólo es divisible por él mismo y la unidad[1]. Es decir, no puede descomponerse en producto de otros números.

Estos números han interesado a los matemáticos desde el inicio de los tiempos, habiendo pruebas de que se conocía su existencia antes del año 1000 a.C. En la antigua Grecia se crearon las primeras tablas de números primos.

Cuando Gauss era joven, recibió como regalo un libro que contenía una lista de números primos. Pero algo en la lista los hacía desconcertantes: no había una manera de, dado un número primo, encontrar el siguiente de la serie. Parecía que habían sido elegidos al azar, así que se decidió a buscar un modelo que pudieran cumplir. En un mundo sin ordenadores, donde las cuentas se tenían que hacer de forma manual, era evidente la ventaja de encontrar ese modelo. Cuando Gauss llegó a la conclusión de que no podía encontrar la respuesta que buscaba, pensó en cambiar la pregunta... y su nueva cuestión fue:

> "Si no puedo predecir cuál será el siguiente número primo, quizá sí pueda contar cuántos hay antes de un número natural dado."

Una vez que se planteó esto, llegó a realizar una aproximación que aún hoy, con las herramientas que tenemos, sigue considerándose buena. Esta aproximación dice que el número de primos entre 1 y N es de $\frac{N}{log(N)}$, donde $log()$ es el *logaritmo natural*.

Esto se concreta en el ***Teorema de los Números Primos***, que dice lo siguiente:

$$\frac{\pi(n)}{n} \to \frac{1}{ln(n)} \quad \text{para n suficientemente grandes}$$

donde $\pi(n)$ representa el número de primos entre 1 y n, y el "$\to$" significa "tiende a". De esta manera consideraremos el error producido por esta aproximación como:

$$error = \frac{\pi(n)}{n} - \frac{1}{ln(n)}$$

Entrada

El programa recibirá una serie de casos de prueba.

Cada caso de prueba se especificará en una línea con dos enteros positivos. El primero, n, será un número natural positivo, menor que 100.000, para el que se quiere poner a prueba la aproximación de Gauss. El segundo, m será un valor entre 0 y 5 que servirá para calcular el máximo error permitido mediante la fórmula:

$$error = \frac{1}{10^m} \quad \text{siendo } m \text{ un entero}$$

El caso de prueba `0 0` será especial y marcará el final de la entrada.

Salida

El programa indicará `Mayor` si el error (en valor absoluto) de la aproximación de Gauss es mayor que el máximo permitido, `Igual` si es el mismo, y `Menor` si es menor.

Como ayuda, recuerda que en C/C++ dispones del fichero de cabecera `math.h`, donde están definidas las siguientes funciones para calcular el logaritmo natural:

[1] Ten en cuenta que el 1 *no* se considera primo.

```
float log(float x);
double log(double x);
```

En Java, puedes usar el método `java.lang.Math.log(double)`.

Atención: Si al enviar tu ejercicio recibes como respuesta "time limit", no significa necesariamente que estés dando respuestas equivocadas, sino que el método que has utilizado es demasiado lento.

Entrada de ejemplo

```
10 3
750 2
65535 2
65535 3
10000 2
99999 1
0 0
```

Salida de ejemplo

```
Mayor
Mayor
Menor
Mayor
Mayor
Menor
```

Fuente

- **Libro:** La música de los números primos (Marcus du Sautoy).
- http://thales.cica.es/rd/Recursos/rd97/UnidadesDidacticas/16-2-o-primos.html

C
De nuevo en el bar de Javier

Tras las medidas tomadas, Javier ha visto que las ventas de su bar han mejorado bastante, así que ha decidido seguir adelante con su estudio. Ahora le gustaría investigar con qué productos gana más dinero y con cuáles gana menos. Además, también le gustaría saber si las ventas en comidas superan la media. Para ello ha establecido varias categorías:

Código	Categoría
D	Desayuno
A	Comidas
M	Meriendas
I	Cenas
C	Copas

Javier encuadra cada venta que realiza dentro de una de esas categorías. Cuando tiene un momento, pasa los datos de todas las ventas al ordenador, y le gustaría que le devolviese los siguientes valores: la categoría que más dinero ha recaudado, la que menos, y si el dinero conseguido con las *comidas* supera la media. No es demasiado constante registrando datos, pero nunca deja un día a medias de introducir.

Realiza un programa que ayude a Javier en su cometido.

Entrada

El programa recibirá una lista de ventas realizadas. Cada una constará de una categoría (`D, A, M, I` o `C`) y un valor (real). Cuando el día termina, Javier introduce una categoría inexistente (`N`) con valor cero (es decir, `N 0`), excepto el último día de la entrada, para el que utiliza `E 0`.

Salida

Para cada día, el programa generará una línea que contendrá tres valores separados por la almohadilla ("`#`"). Los dos primeros indicarán el nombre de las categorías que han supuesto más y menos beneficios respectivamente (ten en cuenta que si de una categoría no se ha vendido nada, su beneficio es cero); las categorías se indicarán con sus nombres, `DESAYUNOS`, `COMIDAS`, `MERIENDAS`, `CENAS` o `COPAS`. El tercer valor de la línea indicará "`SI`" si la media gastada por los clientes en las comidas superó a la media de ventas del día, y "`NO`" en caso contrario.

En caso de que existan varias categorías que hayan conseguido el máximo o mínimo de ventas, se especificará "`EMPATE`".

Entrada de ejemplo

```
D 2.80
C 48.00
A 8.00
N 0
D 15.33
A 60.00
M 12.00
I 25.00
E 0
```

Salida de ejemplo

```
COPAS#EMPATE#NO
COMIDAS#COPAS#SI
```

D
Liga de pádel

Los organizadores de las ligas de pádel de Hill Valley no conocen los ordenadores, de manera que siguen anotando los resultados de cada enfrentamiento en un cuaderno, algo increíble teniendo en cuenta que las ligas que manejan pueden tener hasta 2000 parejas distintas.

Al final de la temporada, terminan teniendo tanto lío, que no saben qué pareja es la ganadora de cada categoría. Por si eso fuera poco, durante el invierno, bien debido a las inclemencias meteorológicas o a lesiones de los participantes, algunos de los partidos de cada jornada no se disputan. El problema es que los jugadores no lo avisan, por lo que los organizadores no apuntan nada en el cuaderno. Afortunadamente, se sabe que todas las parejas han llegado a jugar algún partido.

Haz un programa que ayude a aclarar la situación al final de la temporada.

Entrada

Como entrada, recibirá el nombre de la categoría, seguido de todos los resultados anotados sobre ella. Un resultado se compondrá del nombre de la pareja que juega "en casa", el número de sets que ha ganado, seguido del nombre de la pareja visitante, y el número de sets ganados, separados por espacio. Tanto los nombres de las categorías como de las parejas estarán compuestos de una única palabra de un máximo de 16 letras.

Cada categoría acabará con la palabra `FIN`.

Por su parte, la entrada terminará con una categoría de nombre `FIN`.

Salida

La salida del programa indicará, para cada categoría, el nombre de la pareja ganadora. En caso de empate, se mostrará `EMPATE`. Por cada victoria, la pareja se llevará 2 puntos, por cada derrota se llevará 1, y la no asistencia no sumará ningún punto. Recuerda que en pádel no hay posibilidad de que un partido acabe empatado.

Además de la pareja ganadora (si la hay), también indicará el *número* de partidos no jugados al final de la liga. Ten en cuenta que las ligas tienen ida y vuelta.

Entrada de ejemplo

```
Junior
Buenisimos 3 Malisimos 0
Buenillos 2 Malillos 1
Buenillos 3 Malisimos 0
Buenisimos 3 Malillos 0
Buenisimos 2 Buenillos 1
Malisimos 0 Buenisimos 3
Malillos 1 Buenillos 2
Malisimos 0 Buenillos 3
Malillos 0 Buenisimos 3
Buenillos 1 Buenisimos 2
FIN
Senior
Abuelos 3 Abueletes 0
Abueletes 2 Abuelos 1
FIN
FIN
```

Salida de ejemplo

```
Buenisimos 2
EMPATE 0
```

E

Estrofas

```
Historial ayer borrado,
anteayer hubo pecado.
```

El texto anterior es un *pareado*: una estrofa con dos versos que riman entre sí con rima consonante. ¿Sabrías hacer un programa que identifique distintos tipos de estrofa?

En concreto, nos bastará con identificar las rimas (no tendremos en cuenta el número de sílabas de cada verso), existiendo dos rimas distintas:

- Rima *consonante*: se dice que entre dos versos hay rima consonante cuando todos los sonidos, tanto vocales como consonantes, riman. Para las comparaciones se tienen en cuenta todos los sonidos a partir de la última vocal acentuada.
- Rima *asonante*: similar a la anterior pero únicamente riman las vocales.

Por ejemplo, el siguiente *cuarteto* de Diego de Silva y Mendoza:

```
Una, dos, tres estrellas, veinte, ci ento, (A)
mil, un millón, millares de mill ares,     (B)
¡válgame Dios, que tienen mis pes ares     (B)
su retrato en el alto firmam ento !.       (A)
```

tiene esquema ABBA consonante, pues coinciden las vocales y consontantes del primer y último verso, así como las del segundo y tercero.

Nos piden ser capaces de identificar los siguientes tipos de estrofa:

De dos versos

- **Pareado**: rima consonante AA.

De tres versos

- **Terceto**: rima consonante en el primer y último verso (`A-A`). Ten en cuenta que `AAA` *no* se considerará terceto.

De cuatro versos

- **Cuarteto:** rima consonante ABBA.
- **Cuarteta**: rima consonante ABAB.
- **Seguidilla**: rima asonante en los pares (`-a-a`). Ten en cuenta que otras combinaciones con más rimas o con rima consonante en lugar de asonante (por ejemplo `-aaa` o `-A-A`) *no* se consideran seguidillas.
- **Cuaderna via**: rima consonante igual en todos los versos (`AAAA`).

Entrada

La entrada estará formada por un número indeterminado de casos de prueba. Cada caso de prueba comienza con una línea que contiene un único entero con el número de versos del siguiente poema. A continuación aparecen tantas líneas como versos contiene la estrofa a analizar. Podemos asumir que la última palabra de todos los versos es llana (la vocal acentuada está en la penúltima sílaba). La entrada *no* contendrá tildes para facilitar la programación, aunque esto signifique cometer errores ortográficos. Tampoco tendremos en cuenta que distintos elementos gráficos pueden tener el mismo sonido. Es decir, un verso terminado en *-aba*, no rimará de forma consonante con un verso terminado en *-ava*.

La entrada termina cuando el siguiente caso de prueba contiene `0` versos. Para ese caso de prueba *no* se generará ninguna salida.

Salida

Para cada caso de prueba el programa indicará el nombre de la estrofa, utilizando mayúsculas (`PAREADO`, `TERCETO`, `CUARTETO`, `CUARTETA`, `SEGUIDILLA`, `CUADERNA VIA`) o la palabra `DESCONOCIDO` si no conoce la estrofa dada.

Entrada de ejemplo

```
2
Historial ayer borrado
anteayer hubo pecado
2
Esto no pega
ni con cola.
4
Era un simple clerigo, pobre de clerecia,
dicie cutiano missa de la sancta Maria;
non sabie decir otra, diciela cada dia,
mas la sabie por uso qe por sabiduria.
3
Yo quiero ser llorando el hortelano
de la tierra que ocupas y estercolas,
compañero del alma, tan temprano.
0
```

Salida de ejemplo

```
PAREADO
DESCONOCIDO
CUADERNA VIA
TERCETO
```

Notas

El enunciado ha hecho simplificaciones en las definiciones de las estrofas encaminadas a hacer el ejercicio más sencillo; ejemplos de esto son no considerar el número de sílabas, manejar sólo palabras llanas, tener faltas de ortografía, etc. El resultado ha sido unas definiciones que poco tienen que ver con las aceptadas en la literatura. Por favor, no utilices el programa final delante de un experto en poesía.

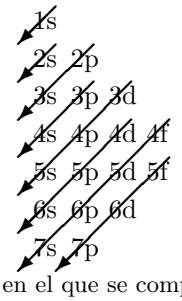
F
Aprobar Química

A Jonathan le han mandado en el instituto que realice un problema de configuración electrónica. La verdad es que con química está bastante perdido, y ha decidido que no va a ser ésta la asignatura que le deje fuera de la universidad. Teniendo en cuenta que en el examen no es crucial para la nota la realización de un ejercicio de este tipo, pero que no se les permite aprobar si no entregan bien realizados todas las actividades propuestas en clase, ha decidido pedirle ayuda a su hermano que está estudiando un Ciclo Formativo de Grado Superior de Informática.

Lo primero que hace Jonathan es explicarle a su hermano el ejercicio, que consiste en indicar de qué forma se distribuyen los electrones de los átomos o *elementos químicos*. Para eso, le cuenta, se utiliza el diagrama de Moëller que determina en qué orden se van completando los subniveles de cada orbital. La idea intuitiva es que para saber cómo se distribuyen los N electrones de un átomo concreto debemos pensar dónde se coloca el primer electrón, después dónde se coloca el segundo, etc., hasta llegar al último. La tabla 1.a muestra todos los "huecos" posibles donde se pueden colocar. Sus nombres son una combinación del número nivel ($1 \ldots 7$) y del orbital (`s`, `p`, `d` o `f`).

	s	p	d	f
n=1	1s			
n=2	2s	2p		
n=3	3s	3p	3d	
n=4	4s	4p	4d	4f
n=5	5s	5p	5d	5f
n=6	6s	6p	6d	
n=7	7s	7p		

(a) Tabla los nombres de los subniveles



(b) Orden en el que se completan

Figura 1: Diagrama de Moëller

La tabla 1.b muestra *en qué orden* se van completando los subniveles. Como se ve, consiste en recorrer los subniveles de forma diagonal de arriba a abajo y de derecha a izquierda:

`1s 2s 2p 3s 3p 4s 3d 4p 5s 4d 5p 6s 4f 5d 6p 7s 5f 6d 7p`

Lo último que hay que tener en cuenta es el número de electrones que entran en cada subnivel: el *subnivel s* puede llenarse con 1 ó 2 electrones. El *subnivel p* puede contener de 1 a 6 electrones, el *d* de 1 a 10 electrones y el *subnivel f* de 1 a 14 electrones:

Orbital	s	p	d	f
Electrones	2	6	10	14

Para que todo le quede más claro, Jonathan le enseña a su hermano un ejercicio que han realizado en clase: la configuración electrónica del Rubidio que tiene 37 electrones (en la terminología química se dice que *el número atómico* del Rubidio es $Z = 37$).

Siguiendo el diagrama de Moëller, los dos primeros electrones irán en el subnivel `1s` y los dos siguientes en `2s`. A continuación se completará el `2p` con seis electrones más (son los que entran en los orbitales `p`). Los dos siguientes irán a parar a `3s` y los siguientes seis a `3p`. En este momento hemos colocado ya 18 electrones. Si continuamos con el proceso hasta colocar los 37 veremos que los 36 primeros electrones completan todos los subniveles hasta el `4p` y por tanto que el último electrón termina en el subnivel `5s`. El número de electrones que quedan en cada subnivel es:

	s	p	d
n=1	2		
n=2	2	6	
n=3	2	6	10
n=4	2	6	
n=5	1		

La forma de indicar la configuración electrónica es mostrar uno tras otro todos los subniveles que tienen electrones utilizando el orden en el que se han ido rellenando. Además, para cada subnivel se indica el número de electrones que han caído en él. Para nuestro ejemplo será:

`1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s1`

El problema consiste en obtener la configuración electrónica de los elementos que nos vaya diciendo Jonathan.

Entrada

La entrada consistirá en una secuencia de casos de prueba, donde cada caso de prueba está formado por dos líneas: el nombre del elemento químico y su número atómico Z (el número atómico estará entre cero y 118).

El programa terminará de recibir valores cuando el nombre del elemento sea "`Exit`".

Salida

Para cada caso de prueba, el programa indicará la configuración electrónica del elemento introducido. La configuración electrónica será la lista de los subniveles en el orden en el que se van rellenando seguido del número de electrones que hay en ese subnivel. Cada subnivel se separará por un espacio en blanco.

Si por un casual nos preguntan por el isótopo del Hidrógeno que no tiene ningún electrón ($Z = 0$), escribiremos `1s0`.

Nota: ten cuidado con los espacios, no pongas de más. Las líneas de salida *no* deben terminar nunca con un espacio.

Entrada de ejemplo

```
Cloro
17
Calcio
20
Rubidio
37
Hierro
26
Exit
```

Salida de ejemplo

```
1s2 2s2 2p6 3s2 3p5
1s2 2s2 2p6 3s2 3p6 4s2
1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s1
1s2 2s2 2p6 3s2 3p6 4s2 3d6
```

Fuente

http://es.wikipedia.org/wiki/Configuracion_electronica
http://jpfisicasemiconductores.blogspot.com/2010_09_01_archive.html

G
Radares de tramo

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), y viene a decir algo así como que, en algún punto de un intervalo cerrado, una función contínua y derivable en ese intervalo tendrá derivada instantánea igual a la derivada media en el intervalo.

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 estará formada por un número indeterminado de casos de prueba. 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 (por ejemplo, indicando que el tiempo que ha tardado el coche es negativo). En esos casos, el sistema mostrará la cadena "`ERROR`".

Entrada de ejemplo

```
9165 110 300
9165 110 299
12000 100 433
12000 100 431
12000 100 359
-1000 -50 -100
0 0 0
```

Salida de ejemplo

```
OK
MULTA
OK
MULTA
PUNTOS
ERROR
```

Fuente

http://curiosoperoinutil.com/2007/04/18/consultorio-cpi-multas-y-teoremas/

H
Semáforos sin parar

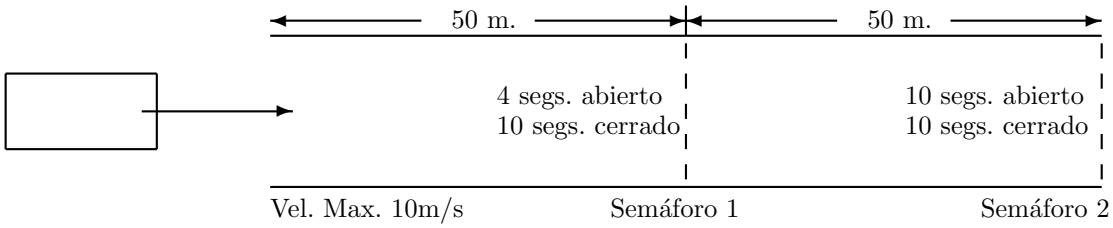
Adolfo ya no aguanta más. No le gusta parar en los semáforos; tanto es así que prefiere ir más lento por las avenidas para que los semáforos se vayan abriendo a su paso en vez de ir a más velocidad y tener que parar en un semáforo cerrado. Bueno, en realidad tiene un límite: no puede ir a menos de 0.1 metros por segundo (m/s) porque si no el coche se le cala.

Ha estado pensando y ha tomado una decisión: cuando se encuentre en una gran avenida llena de semáforos va a intentar ir a una velocidad constante elegida estratégicamente para que todos los semáforos estén abiertos en el momento de pasar por debajo de ellos. Además, quiere que cuando llegue al último semáforo de la avenida, éste se esté justo abriendo o cerrando (en ambas situaciones seguirá su camino).

El cuñado de Adolfo, que trabaja en el ayuntamiento, le ha dado información privilegiada de todas las calles con semáforos de su ciudad (en ninguna de ella hay más de 100 semáforos). Gracias a él, conoce la velocidad máxima de cada avenida, la distancia en metros que hay entre cada semáforo y cuántos segundos está cada uno de ellos abierto y cerrado. El cuñado, al ver su obsesión, también le ha confesado que la policía no tiene la precisión suficiente en sus métodos de detección para multar si pasa hasta una centésima de segundo después de que el semáforo se haya cerrado.

Con todos estos datos, Adolfo se propone averiguar la velocidad a la que tiene que pasar por cada calle para no parar, asumiendo que *cuando comienza el recorrido de la calle todos sus semáforos acaban de cerrarse*.

Consideremos, por ejemplo, una avenida cuya velocidad máxima son 10m/s con dos semáforos. El primer semáforo está situado a 50 metros del principio de la avenida y el último a 100 metros. Los dos tardan 10 segundos en abrirse pero el primero sólo permanece 4 segundos abierto, mientras que el segundo está 10 segundos abierto.



En esta avenida, Adolfo hace sus cálculos y se da cuenta de que lo más rápido que puede ir es a 5m/s. Efectivamente a esa velocidad pasará por el primer semáforo 10 segundos después de haber empezado el recorrido, justo en el momento de abrirse (recuérdese que cuando empieza el recorrido ambos semáforos acaban de cerrarse), y llega al final de la avenida 10 segundos después, justo cuando el último semáforo se cierra.

Adolfo también se da cuenta de que si la velocidad máxima en vez de 10m/s fuera 4m/s otro gallo cantaría. Entonces tendría que decidir ir a 2m/s. Así, pasaría debajo del primer semáforo a los 25 segundos, justo un segundo después de que se abra por segunda vez, y por debajo del último semáforo justo cuando se abre por tercera vez.

Entrada

La entrada estará formada por un conjunto de casos de prueba consecutivos.

Cada caso de prueba constará de dos líneas. La primera línea contendrá dos números enteros, el primero indica el número de semáforos de la avenida y el segundo la velocidad máxima en m/s. La segunda línea tiene la información de todos los semáforos separada por espacios. Cada semáforo consta de tres números

enteros también separados por espacios: la distancia (en metros) con el semáforo anterior (o con el principio de la avenida si se trata del primer semáforo), el número de segundos que permanece cerrado y el número de segundos que está abierto.

Los casos de prueba terminan con `0 0` (es decir, con un caso de prueba sin semáforos y cuya velocidad máxima es 0 m/s).

Ten en cuenta que todos los semáforos *se cierran alguna vez*, pero hay algunos que *no se abren nunca* (lo que se traduce en que el tiempo que permanecen abiertos es `0`).

Salida

Para cada caso de prueba, el programa indicará el número de segundos que Adolfo tarda en recorrer la avenida completa de tal forma que no se salte ningún semáforo (teniendo en cuenta la precisión de la policía) y el último semáforo lo pase justo en el momento de cambiar de rojo a verde o de verde a rojo.

Si con los semáforos que hay en la avenida es imposible pasarlos todos abiertos sin cambiar de velocidad, se escribirá `IMPOSIBLE`.

Entrada de ejemplo

```
2 10
50 10 4 50 10 10
2 4
50 10 4 50 10 10
1 10
10 110 100
3 10
100 31 1 1 30 1 1 31 1
0 0
```

Salida de ejemplo

```
20
50
IMPOSIBLE
IMPOSIBLE
```

Factorial

Tu primo Luis, de 12 años, está aprendiendo a usar la calculadora. Su profesor le ha dicho que calcule el factorial de varios números. Pero, para evitar que le tengan que copiar números muy largos en el cuaderno, les ha pedido únicamente el último dígito, el de más a la derecha.

Recordando que el factorial es la multiplicación de todos los números entre el número y el uno (por ejemplo, el factorial de 8, escrito 8!, es $8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$), demuestra a tu primo Luis que tú eres capaz de hacerlo mucho más rápido que él.

Entrada

El programa recibirá en la primera línea de la entrada el número de casos de prueba. A continuación, cada caso de prueba estará compuesto de una única línea que contendrá un número (positivo).

Salida

Por cada caso de prueba n, se mostrará el último dígito (el de la derecha) de su factorial, $n!$.

Atención: Es especialmente importante lo que se menciona en el enunciado de ser rápido. Si al enviar tu ejercicio recibes como respuesta "time limit", no significa necesariamente que estés dando respuestas equivocadas, sino que el método que has utilizado es demasiado lento.

Entrada de ejemplo

```
3
2
3
4
```

Salida de ejemplo

```
2
6
4
```

J

Número de Kaprekar

El matemático indio Dattaraya Ramchandra Kaprekar trabajó en *teoría de números*, realizando varios descubrimientos a lo largo de su vida. Uno de ellos fue el conjunto de los que, desde entonces, se conocen como *números de Kaprekar*, que son aquellos números enteros positivos que, al ser elevados al cuadrado, pueden descomponerse (para una base dada, que asumiremos ser base 10) en dos enteros positivos cuya suma es igual al número original.

Por ejemplo, el número 703 es un *número de Kaprekar*, dado que 703^2 es 494209 que puede descomponerse en 494 y 209 cuya suma da, de nuevo, 703. Otro ejemplo es el 9 ($9^2 = 81$ y $8 + 1 = 9$).

Hay que tener presente que ambos números en la descomposición *no* tienen por qué tener el mismo número de dígitos. Por ejemplo en el caso del número 2728 tenemos que $2728^2 = 7441984$ que es número de Kaprekar porque $744 + 1984 = 2728$. También puede darse el caso de que el número al cuadrado contenga algún cero. Por ejemplo, con el 4879 tenemos que $4879^2 = 23804641$, que es un número de Kaprekar porque $238 + 04641 = 4879$.

Si bien se permite que *el primero* de los valores de la descomposición sea 0 (y así por ejemplo 1 es número de Kaprekar), el segundo no puede serlo. Debido a ello, el 100 *no* es un número de Kaprekar. Fijate que 100^2 es 10000, que podría descomponerse en 100 y 00 cuya suma es 100. Sin embargo, el segundo número debería ser 0, que no se considera válido.

Entrada

La entrada del programa consistirá en varios casos de prueba. Cada caso de prueba será un número mayor o igual que 1 y menor que 65536. Los casos de prueba terminarán con un 0 que marcará el final de la entrada y que *no* hay que procesar.

Salida

Para cada caso de prueba el programa mostrará `SI` si es un número de Kaprekar, y `NO` en otro caso.

Entrada de ejemplo

```
22222
75
99
100
504
0
```

Salida de ejemplo

```
SI
NO
SI
NO
NO
```

Fuente

http://es.wikipedia.org/wiki/N%C3%BAmero_de_Kaprekar