"First, solve the problem. Then, write the code." John Johnson

Programa-Me 2012
Final Nacional
Problemas

Ejercicios realizados por

Universidad Complutense
de Madrid

I.E.S. Antonio de Nebrija
(Móstoles)

Realizado en el IES Antonio de Nebrija (Móstoles)
5 de junio de 2012

Listado de problemas

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
Marcadores de 7 segmentos

Los paneles informativos están en todos los sitios. Cuando la tecnología se abarató, los carteles de papel se transformaron en letreros luminosos que muestran palabras que van apareciendo por un lado y saliendo por el otro, como el de la figura:

Cuando uno de estos rótulos se pone en marcha, suele empezar vacío y el texto sale desde el lateral derecho hacia el izquierdo, desplazándose hasta desaparecer.

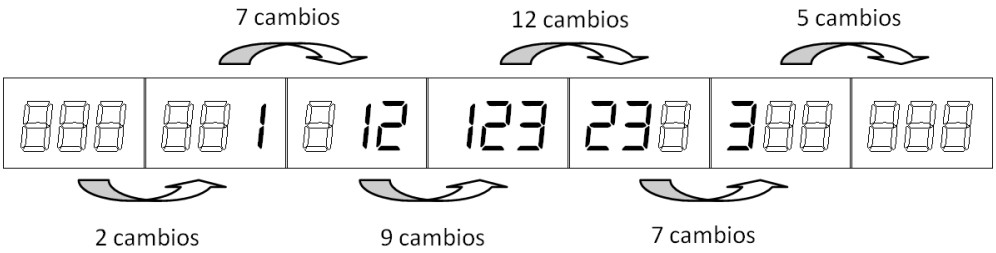
Su funcionamiento consiste en ir encendiendo y apagando las pequeñas bombillas (*leds*) para dar la sensación de movimiento.

Si en vez de texto necesitamos únicamente mostrar números, el cartel puede ser mucho menos sofisticado utilizando los llamados marcadores de siete segmentos como el siguiente:

En este caso cada número se representa gracias a la combinación encendido/apagado de únicamente 7 segmentos luminosos que permiten representar todos los números:

La pregunta que nos hacemos es ¿cuántos cambios de luces (cuántos encendidos y apagados) deberán hacerse para hacer pasar por uno de estos carteles un número determinado?

Por ejemplo, para mostrar el 123 en un cartel de 3 dígitos que comienza con todos los *leds* apagados tendremos:



1. Inicialmente todos los segmentos están apagados.
2. Cuanto entra por la izquierda el `1`, se encienden dos segmentos.
3. Al desplazarse el `1` y entrar el `2` se encienden seis segmentos y se apaga otro, lo que hacen un total de siete cambios.

4. En el siguiente desplazamiento, al entrar el `3`, se encienden siete segmentos y se apagan dos.

5. Al desaparecer el `1`, el dígito de más a la derecha queda por completo apagado; hay cinco segmentos que se encienden y siete que se apagan.

6. La desaparición del `2` provoca otro dígito más apagado por completo y un segmento que se enciende y seis que se apagan.

7. En el último paso se apagan cinco segmentos quedándose el marcador completo apagado.

Eso hace un total de 42 cambios de luces.

Entrada

La entrada contendrá una línea por cada caso de prueba. En cada línea aparecerá una secuencia de enteros ($0 \leq n \leq 9$) separados por espacios que tendrán el mensaje que se quiere mostrar en el marcador; el fin del mensaje se marca con un -1. El ancho del marcador varía con cada caso de prueba y coincide con el número de dígitos total del mensaje, de forma que todos los mensajes serán completamente visibles durante un instante.

La entrada termina cuando nos encontramos ante un marcador en el que no entrará ningún dígito; este marcador no debe generar ninguna salida.

Salida

Para cada caso de prueba se mostrará en una línea independiente el número de cambios de estado de las luces del marcador.

Entrada de ejemplo

```
1 2 3 -1
3 0 5 -1
1 -1
-1
```

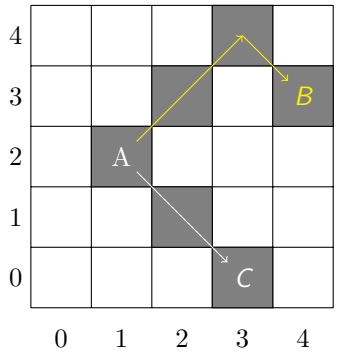
Salida de ejemplo

```
42
48
4
```

B
Mejor... imposible

En la película *Mejor... imposible*, Jack Nicholson encarna a Melvin Udall un "simpático" personaje que padece un desorden obsesivo que, entre otras cosas, le hace andar por las aceras sin pisar las líneas de unión entre las baldosas.

Perico Almodabar se está planteando hacer una película con un personaje parecido al de Nicholson pero cuyo desorden le hace andar por las aceras siempre *en diagonal*. Es decir, cuando está situado en una baldosa sólo puede ir a una de las cuatro baldosas de sus diagonales. Como ejemplo, en el siguiente diagrama se ve que para ir del punto A al punto B necesita dos movimientos, mientras que para ir de A a C necesita sólo uno. Se considera un movimiento al desplazamiento completo en linea recta, *no* al cambio de una baldosa a otra.



La pregunta que nos hacemos es, dadas dos posiciones de la acera (que supondremos cuadrada), ¿cuál es el mínimo número de movimientos que necesita nuestro personaje para llegar de una a la otra?

Entrada

La entrada estará compuesta por distintos casos de prueba, cada uno ocupando dos líneas. La primera contendrá un número ($1 \leq N \leq 10^9$) que indica el tamaño (en número de baldosas) de cada lado de la acera. La segunda línea contendrá cuatro números, $x1$, $y1$, $x2$, $y2$, que indican la posición inicial $(x1, y1)$ y la final $(x2, y2)$. Todos esos números estarán en el rango comprendido entre 0 y $N - 1$.

La entrada terminará cuando el tamaño de la acera sea 0.

Salida

Para cada caso de prueba se indicará en una línea el número mínimo de movimientos que tiene que hacer el personaje para alcanzar su objetivo. Si es imposible llegar andando en diagonal, se escribirá `IMPOSIBLE`.

Entrada de ejemplo

```
5
1 2 4 3
5
1 2 3 0
10
0 0 0 1
0
```

Salida de ejemplo

```
2
1
IMPOSIBLE
```

Fuente

Basado en un problema del Juez on-line de la Universidad de Valladolid (código de ejercicio 10849, `http://uva.onlinejudge.org/external/108/10849.html`).

C
Llenando piscinas

Se acerca el verano y llega el momento de sacar de los armarios y trasteros las piscinas para los niños (y no tan niños), colocarlas en la terraza, patio o jardín y llenarlas de agua para que los pequeños de la casa puedan empezar a disfrutarlas.

Este año la tarea se presenta complicada porque durante el invierno la larga manguera que permitía llevar el agua desde el grifo de la cocina hasta la propia piscina se ha perdido y habrá que hacerlo con un barreño...

Para complicar aún más las cosas, también durante el invierno la piscina (a pesar de ser de fibra de vidrio) se ha pinchado y pierde un poco de agua. Aún así, como los pequeños están ansiosos por darse un chapuzón decidimos llenarla cuanto antes, con pinchazo incluido, y luego, mientras ellos disfrutan, lo arreglaremos. Dado que la piscina está perdiendo agua constantemente, estará llena únicamente durante un instante de tiempo. En ese preciso momento dejaremos de hacer viajes a la cocina y nos pondremos rápidamente a arreglarla.

Como dice el refrán "mal de mucho consuelo de tontos"; la tarea de llenado será un poco más llevadera gracias al consuelo de saber que nuestro vecino está en la misma situación. A través del seto del jardín podemos verle haciendo viajes como un loco de su cocina a su piscina, para compensar el pinchazo que también él tiene. La pregunta es ¿quién tardará menos en llenar la piscina?

Entrada

La entrada estará compuesta de múltiples casos de prueba, cada uno en una línea.

Cada uno de los casos de prueba empezará con la descripción de una "competición" entre nosotros y el vecino. Los tres primeros números indican los litros de agua de nuestra piscina ($1 \leq p \leq 10^9$), el número de litros de nuestro barreño ($1 \leq b \leq 10^9$) y por último los litros de agua que la piscina pierde durante el viaje ($0 \leq n \leq 10^9$). A continuación aparecen tres números para indicar la misma información pero de nuestro vecino. Se garantiza que al menos uno de los dos podremos terminar llenando la piscina.

La entrada termina cuando alguna de las piscinas no tiene capacidad (aparece a cero).

Salida

Para cada caso de prueba aparecerá una línea compuesta por el ganador y el número de viajes que ha realizado. Como ganador, se indicará `YO` si nosotros terminamos antes de llenar la piscina (hacemos menos viajes para conseguirlo) y `VECINO` si es el vecino. En caso de empate, se indicará `EMPATE`.

Entrada de ejemplo

```
10 5 1 15 6 1
50 5 1 50 5 0
50 5 1 50 5 6
0 0 0 0 0 0
```

Salida de ejemplo

```
EMPATE 3
VECINO 10
YO 13
```

D
Las cartas del abuelo

El abuelo lleva toda la vida escribiendo cartas a máquina. Aún recuerda con nostalgia aquella vieja máquina de escribir en la que metía el folio y lo colocaba centrándolo con mimo, para después pulsar las duras teclas que terminaban moviendo de forma mecánica los tipos que golpeaban el papel. "`En relación a tu misiva del...`" decía el abuelo. TAC, TAC, TAC, TAC... contestaba la máquina de escribir. Sus dedos han desplazado miles y miles de veces los tipos de su vieja máquina de escribir desde su posición estática hasta los folios que luego terminaban siendo leídos por familiares en distintas partes del mundo. La hoy silenciosa máquina, ha sido testigo ruidosa de alegrías y penas, nacimientos y despedidas, felicitaciones navideñas y lamentos por los tiempos que quedaron atrás.

Pero llegó el día en el que la vieja máquina de escribir no pudo trabajar más. A la creciente dificultad en encontrar cintas de tinta para ella, se unió la falta del necesario mantenimiento de sus diferentes elementos (limpieza y lubricado), que terminaron desencadenando la rotura de piezas internas descatalogadas hacía décadas.

Hoy, guiado por sus nietos, el abuelo intenta acostumbrarse a la escritura de cartas en un ordenador. Pero le cuesta mucho. Los teclados modernos son mucho más delicados y los dedos del abuelo insisten en presionar las teclas con fuerza y lentitud. Para su desesperación, el ordenador se empeña en repetir las letras, algo que su vieja amiga nunca habría hecho. El resultado es que cuando intentar escribir:

```
Querido hermano, te ...
```

lo que obtiene son cosas como:

```
Quuueeriiiiiiiido   hermaaaanooooo,,   ttte ...
```

La pregunta que nos hacemos es ¿serías capaz de saber si entre dos posiciones del texto que ha escrito todas las letras son iguales? Dicho más formalmente, dada la cadena escrita por el abuelo y dos posiciones i y j de esa cadena, queremos que nos digas si todos los caracteres entre la posición i y la j, ambas inclusive, son iguales o no.

Entrada

La entrada estará compuesta de distintos casos de prueba. Para cada caso de prueba la primera línea consistirá en la cadena (de hasta un millón de caracteres) escrita por el abuelo. Posteriormente vendrá un número n que indica el número de intervalos que vienen a continuación. Siguen los n intervalos en líneas independientes, expresados como dos enteros que siempre estarán en el intervalo $0..l-1$, siendo l la longitud de la cadena.

La entrada termina cuando el número de intervalos de un caso de prueba es 0.

Salida

Cada intervalo de números i, j genera una línea en la salida. Si todos los caracteres entre $min(i,j)$ y $max(i,j)$ (ambos incluídos) son iguales, se escribirá `SI`; si hay caracteres distintos, se escribirá `NO`. Al terminar el conjunto de intervalos de cada caso de prueba se añadirá una línea en blanco adicional.

Entrada de ejemplo

```
Hooolaa
4
0 6
1 3
3 2
1 2
Adios
1
1 2
Y con esto, Dios te de salud, y a mi no olvide. Vale.
0
```

Salida de ejemplo

Fuente

Basado en el problema "Zeros and ones" del Juez on-line de la Universidad de Valladolid (código de ejercicio 10324, `http://uva.onlinejudge.org/external/103/10324.html`).

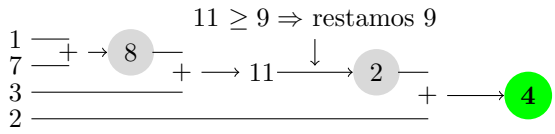
La última frase de la entrada de ejemplo es también la última de la introducción a la primera parte de *El ingenioso hidalgo Don Quijote de la Mancha*, de Miguel de Cervantes.

E

Prueba del nueve en base N

La conocida como *prueba del nueve* es una técnica que nos permite comprobar si nos hemos equivocado al calcular el resultado de una operacion aritmética. La aplicación de la prueba del nueve requiere dos pasos. El primero es *reducir* (o "resumir") cada uno de los números de la operación a comprobar (operandos y resultado) a *un sólo dígito*, y luego repetir la operación original con esos dígitos.

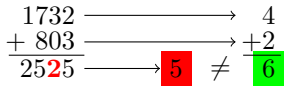
Hay muchas maneras diferentes de realizar el primer paso de "reducción", pero todas acaban con el mismo valor. Una de las más sencillas consiste en ir sumando cada uno de los dígitos del número a reducir a la suma de los anteriores. Cada vez que sumamos el valor de un dígito, comprobamos si la suma acumulada hasta el momento es mayor o igual que 9. Si lo es, antes de continuar sumando restaremos 9 a ese valor acumulado. El objetivo es conseguir que el *resumen* de cualquier número sea un dígito comprendido entre 0 y 8. Por ejemplo, para el número 1732:



Otros ejemplos de reducción son:

$$126 \to 0 \qquad 803 \to 2 \qquad 2525 \to 5 \qquad 2535 \to 6$$

El segundo paso de la prueba del nueve es repetir la operación original *usando los dígitos "resumen"*, y comprobar si el resultado obtenido es el mismo que el dígito *resumen* del resultado original. Por ejemplo, para comprobar una suma, primero se reducen a un solo dígito tanto los operandos como el resultado. Una vez hecho esto, se suman los resúmenes obtenidos y se comprueba si el resultado es igual al resumen obtenido para él. La siguiente figura muestra un caso en el que la prueba del nueve detecta un error:



En este ejemplo, la suma de los *resúmenes* ha dado directamente un único dígito (el 6). Si hubiera dado un número mayor, tendríamos que aplicar el proceso descrito anteriormente hasta reducirlo.

Se debe tener en cuenta que la prueba del nueve *no es infalible*. Si en el caso anterior al hacer la suma hubiéramos llegado a la solución (incorrecta) 2526, el dígito *resumen* habría sido 6, y habría superado la prueba del 9. No obstante, se garantiza que si la prueba del nueve detecta un error, la operación original estará mal. Dicho de otro modo, la prueba del nueve puede dar *falsos positivos* (decir que la operación está bien, cuando no lo está), pero nunca *falsos negativos* (decir que la operación está mal, cuando está bien).

Para comprobar una división, *no* se hace la división con los resúmenes, dado que el dividendo podría ser cero y la división no podría calcularse. En su lugar, nos apoyamos en que el dividendo debe ser igual a la multiplicación del divisor por el cociente más el resto ($D = d \cdot c + r$)

Por ejemplo, la división de 1732 entre 803 da 2, con resto 126, y por tanto se cumple que:

$$(803 \times 2) + 126 = 1732$$

Usando los resúmenes:

$$(2 \times 2) + 0 = 4$$

Si al calcular la división nos hubiéramos equivocado en el cociente o en el resto, probablemente la prueba del nueve lo habría detectado.

Para demostrar matemáticamente que la prueba del 9 funciona, se utiliza aritmética modular y el teorema fundamental de la numeración. De hecho, se puede demostrar que esta prueba funciona para todas las bases B de numeración cambiando únicamente el nueve de la descripcion por la base en la que estemos trabajando menos uno $(B - 1)$. En ese caso, cada vez que sumemos mientras calculamos el dígito *resumen*, tendremos que comprobar si el número alcanzado es mayor o igual que $B - 1$ y restar $B - 1$ cuando eso ocurra.

Entrada

La entrada estará compuesta por un numero que indicará el número de casos de prueba que aparecerán a continuación. Para cada caso de prueba se proporcionarán cinco números. El primero de ellos será la base en la que se encuentran los números que aparecen a continuación, y será un entero mayor que 2 y menor o igual que 36. Los cuatro siguientes serán el dividendo, el divisor, el cociente y el resto de una división, y estarán representados en la base dada. Ninguno de esos números superará el millón de dígitos.

Cuando se utilicen bases superiores a 10, los dígitos mayores que 9 se representarán mediante letras del alfabeto inglés (en mayúsculas), tal y como muestra la tabla siguiente.

Número:	0	1	2	3	...	9	10	11	...
Símbolo:	0	1	2	3	...	9	A	B	...

Salida

Para cada caso de prueba, se escribirá `INCORRECTO` si la prueba descrita determina que la división indicada en el caso de prueba es errónea, y se escribirá `POSIBLEMENTE CORRECTO` si no se detecta error alguno.

Entrada de ejemplo

```
4
10 1732 803 2 126
10 199 13 16 4
16 AF 12 9 D
3 2 1 1 0
```

Salida de ejemplo

```
POSIBLEMENTE CORRECTO
INCORRECTO
POSIBLEMENTE CORRECTO
INCORRECTO
```

F
Escalera de color

La llamada *baraja inglesa* es una modificación menor de la *baraja francesa*. Sus similitudes son tan grandes, que es habitual considerarlas la misma. Contiene 52 cartas, distribuídas en 4 palos diferentes. Los palos se conocen con el nombre de *picas* (♠), *diamantes* (♢), *tréboles* (♣) y *corazones* (♡). De cada uno, hay trece cartas, con valores del 1 (al que se conoce como *As*) hasta el 10, más las tres figuras, *Jack* (*J*), *Queen* (*Q*) y *King* (*K*), que, numéricamente, serían los valores 11, 12 y 13. Las diferencias más notables entre la baraja francesa y la inglesa están en el nombre de la carta *Jack* (conocida en la francesa como *Valet*, *V*), y el As, nombre específico de la baraja inglesa que, además, desplaza su valor en muchos juegos del 1 al 14, convirtiéndola en una carta más poderosa que la *K*.

La baraja inglesa se utiliza en juegos mundialmente conocidos, como el *bridge*, la *canasta* o el *póquer*.

En este último, la jugada más valiosa es una *escalera de color*, que se forma cuando un mismo jugador consigue una mano de 5 cartas del mismo palo con valores consecutivos.

Entrada

La entrada estará compuesta por sucesivos casos de prueba. Cada uno ocupará una línea, y estará compuesto por una mano *de cuatro cartas* del juego del póquer.

Cada carta se representará indicando primero su *número* (`2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `J`, `Q`, `K`, `A`) y luego su palo (`P` para picas, `D` para diamantes, `T` para tréboles y `C` para corazones) separados por un espacio. Las cuatro cartas de cada caso de prueba estarán también separadas por un espacio.

La entrada finalizará cuando se reciba un `0` en lugar del valor de la primera carta de la mano.

Salida

Para cada caso de prueba el programa escribirá la carta necesaria que habría que añadir a las cuatro recibidas para obtener la escalera de color más alta posible. La salida vendrá dada por un valor y un palo, con el mismo formato que en la entrada. Si con las cartas del caso de prueba resultase imposible crear una escalera de color, se escribirá `NADA`.

Aunque en el desarrollo de una partida de póquer para crear una escalera el As puede utilizarse tanto al principio de los *números* (valor 1) como al final (valor 14), el programa considerará que sólo puede colocarse después de una *K*.

Entrada de ejemplo

```
2 C 3 C 4 C 5 C
Q P 9 P 7 P 6 P
Q P K P 9 P 10 P
0
```

Salida de ejemplo

```
6 C
NADA
J P
```

G
Viaje en el tiempo

Con motivo del centenario del nacimiento de Alan Turing, Marty McFly Jr. ha pedido al científico amigo de su padre, Doc Emmett Brown, que le ayude a realizar un viaje en el tiempo que le permita conocerle[1]. Para eso, se trasladan a Cambridge, introducen en el teclado del De Lorean la fecha 5 de Junio de 1934 y viajan hasta ese día. Tras una búsqueda intensiva por las aulas del King's College, se encuentran al genio en su despacho intentando desarrollar una respuesta al *Entscheidungsproblem* o, en español, "problema de decisión". Alan tardaría aún dos años más en publicar sus conclusiones, basadas en la que hoy es conocida como *Máquina de Turing*. Esta gran aportación del genial matemático le haría figurar como el padre de la informática moderna.


King's College, Cambridge. 1934

En 1934, sin embargo, Marty y Doc se encuentran a Alan en una etapa temprana de su trabajo, desarrollando una máquina muy sencilla en la que, además, parece haber algún problema de funcionamiento. Animado por el interés que muestran los recién llegados, Turing les cuenta su problema. Está tratando de crear una máquina que genere una serie numérica en la que cada número sea la suma de una constante entera al número inmediatamente anterior. Por desgracia, la máquina no está totalmente depurada, y de vez en cuando parece modificar esa constante, usando el nuevo valor a partir de ese momento para el resto de números, hasta el próximo fallo. El resultado es que las secuencias generadas por la máquina no son series numéricas correctas.

Alan muestra a los viajantes temporales el resultado de la última serie que ha calculado su máquina. Los primeros valores eran `1, 5`, por lo que el resultado debería ser `1, 5, 9, 13,...` Pero en realidad la respuesta de la máquina ha sido `1, 5, 7, 9, 11`, es decir, a partir del valor `5`, la máquina ha pasado de calcular una sucesión de término 4 a una de término 2.

La máquina no está definida para valores negativos, ni superiores a `999.999`. Para la máquina, el espacio de números es circular, es decir por debajo de `0` se encuentra el `999.999, 999.998, ...` y por encima de `999.999` está el `0, 1, 2, 3`, etc.

En su búsqueda de la causa del problema, Turing está intentando encontrar alguna relación entre las secuencias y el número de veces que la máquina introduce un cambio en el término de la sucesión. También quiere saber cuál sería el siguiente resultado que le daría la máquina, teniendo en cuenta el último cambio de término. Esto resulta ser una ardua labor, pues las secuencias generadas por la máquina son arbitrariamente largas, y una comprobación manual es muy tediosa.

Marty y Doc le ofrecen su ayuda, sabiendo que, regresando al futuro, esa labor podrán hacerla de manera automática gracias a los ordenadores, los tataranietos de la máquina de Turing. Planean añadir así su granito de arena al desarrollo de la informática, al igual que hizo Marty en su primera visita a 1955 con el Rock and Roll[2]. Rodeados de un montón de cintas de Turing con secuencias de números erróneas, adelantan 78 años el temporizador del De Lorean y te buscan para que les ayudes.

Entrada

La entrada estará compuesta por un número indicando la cantidad de casos de prueba que vendrán a continuación. Cada caso de prueba consistirá en una serie de números enteros no negativos, con una de las

[1] El padre de Marty Jr. es conocido por las dificultades que tuvo en su *Regreso al Futuro* (desde 1955 a 1985) con la máquina inventada por Doc. Ambos saltaron a la fama cuando Robert Zemeckis y Steven Spielberg llevaron sus vidas a la gran pantalla.

[2] Nunca se ha esclarecido quién fue realmente el autor original de Johnny B. Goode.

secuencias que Turing nos ha pedido que analicemos. Todas las secuencias tendrán al menos dos números, y terminarán con un `-1` adicional, que no deberá ser procesado.

Salida

Para cada caso de prueba, el programa mostrará el número de veces que la máquina ha cambiado el término de la sucesión y el siguiente número de la misma.

Entrada de ejemplo

```
4
1 5 7 9 11 -1
1 2 3 4 5 6 7 8 10 -1
999996 999998 0 2 4 -1
10 9 8 7 3 2 -1
```

Salida de ejemplo

```
1 13
1 12
0 6
2 1
```

H
Encadenando trolls

"Y antes que amanezca cambiará al este — dijo Legolas — Pero descansad, si tenéis que hacerlo. Mas no abandonéis toda esperanza. Del día de mañana nada sabemos aún. La solución se encuentra a menudo a la salida del sol."[3]

Tras la caída de Sauron, algunos de los *hobbits* más jóvenes de La Comarca han decidido establecer rutas de comunicación con las demás razas. En algunas de las ocasiones que han tenido que atravesar el *Bosque de los Trolls* en su camino a las *Montañas Nubladas*, se han visto obligados a huir de esta desagradable especie. Por ello han tenido la idea de encadenarlos a todos, pero no han reunido el valor suficiente para hacerlo.

Sin embargo, y gracias a la amistad que une a determinadas familias con los enanos, han conseguido que éstos les ayuden. El primer paso es construir las pesadas cadenas que usarán para inmovilizar a los indeseables *trolls*. Los herreros de *Aglarond* son conocidos por su gran destreza, por lo que se encargarán de fabricarlas. Naturalmente, cada eslabón tiene que ser lo suficiente grande y robusto como para que soporte las embestidas de los enormes *trolls*. El problema es que los *hobbits* no son precisamente una raza que se caracterice por su fuerza, por lo que cada uno puede sólo transportar unos cuantos eslabones. Los enanos son mucho más fuertes, pero tendrán que transportar sus pesadas herramientas hasta el lugar donde se encadenará a los *trolls*, por lo que, en la práctica, cada enano sólo podrá cargar con el doble de eslabones que un *hobbit*.

Cada vez que se fabrica una cadena, deben decidir cómo se transporta. Si la cadena es lo suficientemente ligera como para que pueda ser transportada por un único individuo (ya sea *hobbit* o enano), se traslada tal cual. En caso contrario, habrá que partirla.

Sorprendentemente, han llegado a la conclusión de que trocearán la cadena en la misma proporción que sus fuerzas, es decir, haciendo una parte el doble de larga que la otra. Si la división no es exacta, los eslabones que sobren quedarán en el fragmento más grande. Ocurre en ocasiones que alguno de los fragmentos resultantes (¡o ambos!) sigue siendo demasiado pesado incluso para los enanos, por lo que repiten el proceso con él. Al final, una cadena puede terminar troceada en un montón de segmentos, cada uno con un número de eslabones diferente.

A ninguna de las razas les molesta que el reparto del trabajo no sea equitativo, por lo que cualquiera de los dos segmentos de la cadena (sea el largo o el corto) será transportado por el primer individuo que pueda con ella.

Entrada

La entrada estará compuesta por una serie de casos de prueba. Cada uno consistirá en dos números enteros no negativos. El primero indicará la fuerza de los *hobbits*, dando el número de eslabones que son capaces de llevar. El segundo indicará el tamaño (en número de eslabones) de la cadena que hay que transportar.

La entrada acabará cuando el primer número sea cero.

Salida

Para cada caso de prueba, se indicará el número de eslabones que hay que romper para construir los segmentos. Naturalmente, esos eslabones que se abren deben también ser transportados con alguno de los segmentos.

[3] Capitulo II del Libro Tercero. Las Dos Torres. El Señor de los Anillos. (J.R.R. Tolkien)

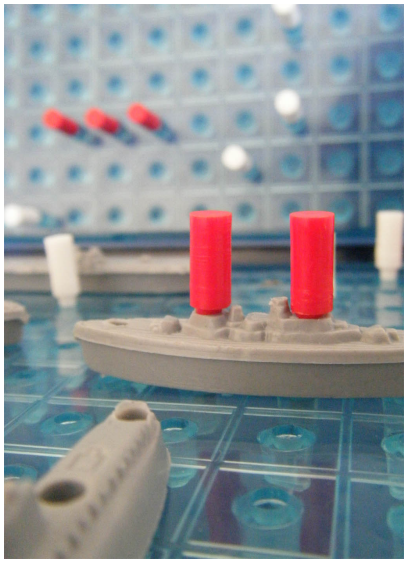
Entrada de ejemplo

```
10 8
5 10
1 3
1 4
0 0
```

Salida de ejemplo

```
0
0
1
2
```

1 Hundir la flota



El juego *Hundir la flota* o *Batalla naval* es un clásico juego infantil de lápiz y papel, del que se han comercializado multitud de versiones como juego de mesa.

Cada jugador dispone de dos tableros, formados por una cuadrícula de celdas. Cada jugador coloca, secretamente, su *flota de barcos* en uno de sus tableros, que se mantiene oculto del adversario durante toda la partida. Cada barco de la flota ocupa un número variable de celdas, y deben colocarse en el tablero vertical u horizontalmente. No se permite colocar barcos adyacentes, es decir, dos barcos *no pueden tocarse*.

En cada ronda, los jugadores se alternan para "disparar" sobre el tablero de barcos del oponente. Si en la posición escogida se encuentra una de las partes de un barco, el jugador tendrá que anunciar que su barco ha sido *tocado* o incluso *hundido* si ya han sido golpeadas todas las partes de ese barco. En otro caso, anunciará *agua*. Cuando un jugador hace un disparo, anota en su segundo tablero el resultado, para recordar las celdas que ya han sido exploradas. Gana el primer jugador que hunde completamente la flota del adversario.

Hay multitud de variedades del juego, algunas más simples que otras. El tamaño del tablero, el número, longitud y nombre de los barcos son los puntos de variación más obvios. Otras alternativas hacen, por ejemplo, uso de *salvas*, es decir múltiples disparos simultáneos en el mismo turno, cuyo número va disminuyendo al sufrir el hundimiento de buques.

Entrada

La entrada estará compuesta de múltiples casos de prueba. Cada caso de prueba contendrá la definición de una variante del juego (tamaño de tablero, y número y longitud de los barcos), y la configuración de un tablero. En concreto, estará compuesto de:

- Número de barcos. Una flota tendrá un máximo de 10 barcos.
- Tamaño de cada barco. Para cada uno se indica el número de celdas que ocupa. Este número será siempre mayor que cero.
- Tamaño del tablero ("océano"). El tablero siempre será cuadrado, por lo que aquí se proporciona un único valor, indicando el número de filas (y columnas). El tamaño del tablero será como máximo 128x128.
- Definición del tablero. Una celda con `0` indicará *agua*, y una celda con `1` indicará que hay una parte de un barco. Cada fila del tablero aparece en una línea independiente y los números están separados por espacios.

La entrada finaliza con un caso de prueba en el que el número de barcos es cero.

Salida

Para cada caso de prueba, el programa escribirá `SI` si el tablero es correcto y `NO` en otro caso. Un tablero será correcto si contiene exactamente los barcos de la variante del juego descrita en el caso de prueba (ni

más ni menos), con los tamaños indicados. Los barcos pueden colocarse en vertical o en horizontal, y no pueden tocarse (ni siquiera en diagonal).

Entrada de ejemplo

```
2
2 3
6
0 1 1 0 0 0
0 0 0 0 0 0
0 0 0 1 0 0
0 0 0 1 0 0
0 0 0 1 0 0
0 0 0 0 0 0
2
2 3
5
0 1 1 1 0
0 0 0 0 1
0 0 0 0 1
0 0 0 0 0
0 0 0 0 0
0
```

Salida de ejemplo

J
Ceros del factorial

El factorial de un número n (representado como $n!$) es el resultado de multiplicar todos los números entre 1 y el propio n:

$$n! = 1 * 2 * ... * (n-1) * n = (n-1)! * n$$

Además, por convenio, el factorial de 0 es 1 (es decir, $0! = 1$).

Es sabido que el factorial crece muy rápidamente, de forma que para n's pequeños se obtienen $n!$ con un gran número de dígitos; por ejemplo, el factorial de 24 es 620.448.401.733.239.439.360.000, un número que está muy lejos del máximo soportado en los tipos `int` de los lenguajes de programación tradicionales de 32 bits.

El escenario empeora rápidamente; 70! es el primero que rompe la barrera de los 100 dígitos, mientras que 100! tiene ya 158.

Es por esto que hoy, en vez de intentar calcular el factorial de semejantes números, nos conformaremos con saber cuántos ceros tienen al final.

Entrada

La entrada comenzará con un número natural indicando el número de casos de prueba. Cada caso de prueba aparecerá en una línea independiente con un número natural entre 0 y $2^{31} - 1$.

Salida

Para cada caso de prueba, el programa mostrará el número de dígitos a 0 consecutivos que aparecen a la derecha (dígitos menos significativos) de su factorial.

Entrada de ejemplo

```
3
5
10
24
```

Salida de ejemplo

```
1
2
4
```