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

Programa-Me 2014
Final Nacional
Problemas

Patrocinado por

Ejercicios realizados por

Universidad Complutense
de Madrid

I.E.S. El Cañaveral
(Móstoles)

Realizado en la **Facultad de Informática (U.C.M.)**
6 de junio de 2014

06 de junio de 2014
http://www.programa-me.com

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. El Cañaveral - Móstoles)

A
Números de Lychrel

Cuando se aburren, los aficionados a las matemáticas se dedican a jugar con los números. Eso les lleva, por ejemplo, a coger cualquier número, darle la vuelta y sumarlo a sí mismo, repitiendo el proceso una y otra vez hasta dar con un número capicúa. Por ejemplo, para el 91 llegamos a un capicúa en sólo dos pasos:

$$91 + 19 = 110$$
$$110 + 011 = \textit{121}$$

Algunos números se resisten a alcanzar un capicúa. El 196 es el número más pequeño para el que no se ha llegado a ninguno, por más que se ha intentado. Los matemáticos no han podido demostrar que, efectivamente, no vaya a llegarse a uno. Mientras continúan buscando una demostración, los aficionados siguen sumando y sumando con la esperanza de llegar a él. Los números con los que, se sospecha, no puede alcanzarse un capicúa se conocen como *números de Lychrel*. Curiosamente, algunos números capicúa parecen ser también números de Lychrel.

Entrada

La entrada comienza con un número indicando la cantidad de casos de prueba que vendrán a continuación. Cada caso de prueba estará compuesto de un número $1 \leq n \leq 100.000$.

Salida

Para cada caso de prueba n, el programa deberá indicar el número de iteraciones que hay que dar hasta llegar a un número capicúa, seguido del número capicúa alcanzado. Si durante el proceso se llega a un número mayor que 1.000.000.000, se deberá asumir que el capicúa no es alcanzable y escribir "`Lycrhel?`"[1].

Entrada de ejemplo

```
4
91
196
4994
5445
```

Salida de ejemplo

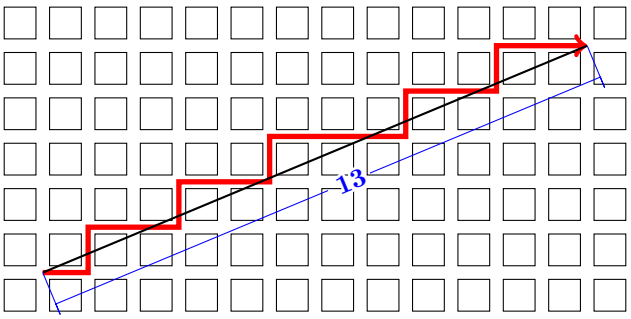
```
2 121
Lychrel?
Lychrel?
4 79497
```

[1] Esta suposición *es falsa* y se usa para simplificar el problema. A modo de ejemplo, el número 10.677 alcanza el capicúa 4.668.731.596.684.224.866.951.378.664, de 28 dígitos, tras 53 iteraciones. Sin embargo, el programa escribirá, para ese número, "`Lychrel?`".

B
En taxi por Manhattan

Debido al crecimiento de población en la isla de Manhattan (entre 1790 y 1810 se había triplicado), en 1811 se presentó el conocido como "plan de los comisionados" (*Commisioners' plan*) que planteó un plano de la futura ciudad organizándola en una cuadrícula. En ella, las avenidas cruzarían la isla de Norte a Sur, mientras que las calles lo harían de Oeste a Este, creando manzanas de unos 5 acres. Esta regularidad en el plano, unido a que las avenidas y calles se numeran en lugar de bautizarse con nombres de personajes ilustres, hace que, desde entonces, orientarse por Manhattan sea un juego de niños.

La pega para los peatones, sin embargo, es que no hay *atajos*. Cuando alguien debe desplazarse de un lugar a otro, está obligado a moverse o bien en horizontal o bien en vertical, pues no existen calles que recorran la isla en diagonal.



Pero para los turistas que cogen taxis es una bendición, porque es muy fácil darse cuenta cuándo los conductores de estos famosos coches amarillos intentan dar un rodeo para llevarlos a su destino. En Nueva York, es muy difícil ser taxista defraudador... ¿verdad?

Entrada

La entrada comienza con un número que indica la cantidad de casos de prueba que habrá que procesar.

Cada caso de prueba consiste en una linea con un número natural $1 \leq n \leq 30.000$ que representa la distancia *en diagonal* entre el punto origen y el punto destino de un turista en Nueva York. Ambos puntos estarán siempre en el centro de las intersecciones de dos calles y avenidas distintas.

Salida

Para cada caso de prueba, el programa escribirá, en una línea independiente, la distancia *mínima* que habrá recorrido el taxi para llevar al turista a su destino. La distancia entre el centro de dos cruces adyacentes es de 1.

Si para la distancia del caso de prueba existen varios posibles destinos, se asumirá que el turista ha solicitado ir a aquél que esté más cerca en taxi. Además, se garantiza que todos los casos de prueba tendrán al menos un destino válido.

Entrada de ejemplo

```
2
13
25
```

Salida de ejemplo

```
17
31
```

C
Sombras en el camping

Se acerca el verano y los aficionados a la naturaleza pasarán buena parte de él en *campings*, disfrutando del aire libre.

Un requisito imprescindible en los meses de calor es colocar la tienda de campaña bajo la sombra de un buen árbol para poder pasar frescos las horas de siesta. Pero, dependiendo de la zona, eso no siempre es fácil. En los *campings* nuevos, el número de árboles es escaso, y también lo es por tanto el número de *parcelas* aptas para tiendas.

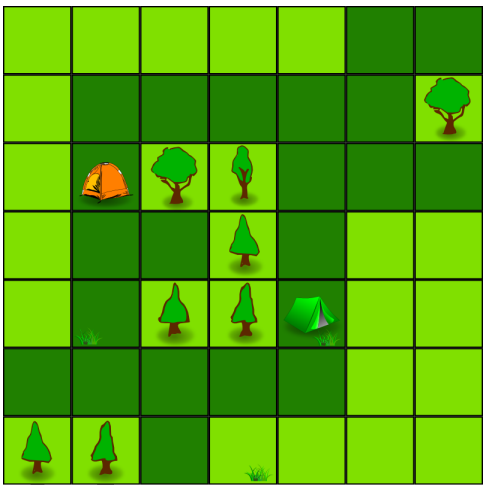


Figura 1: primer ejemplo de entrada destacando las zonas de sombra

Sabiendo que cada árbol proporciona sombra a las ocho parcelas adyacentes, ¿cuántas tiendas de campaña disfrutarán de sombra en un *camping*?

Entrada

El programa deberá procesar múltiples casos de prueba recibidos por la entrada estándar. Cada uno representa un *camping* formado por una cuadrícula de parcelas de igual tamaño en los que puede haber hueco para una tienda, o un árbol.

Cada caso de prueba comienza con dos números $1 \leq c, f \leq 50$, indicando el número de columnas y de filas de la cuadrícula de parcelas. A continuación se indica el número a de árboles del camping.

Si hay árboles, en la siguiente linea aparece la posición de todos ellos, indicando para cada uno la columna $(1 \ldots c)$ y la fila $(1 \ldots f)$ que ocupan. En total, aparecerán $2 \times a$ números.

La entrada termina con una línea con tres ceros (camping con dimensiones nulas y sin árboles), que no debe procesarse.

Salida

Por cada caso de prueba el programa indicará, en una línea, el número de parcelas que disfrutarán de sombra.

Entrada de ejemplo

```
7 7 8
7 2 3 3 4 3 4 4 3 5 4 5 1 7 2 7
5 3 1
3 2
0 0 0
```

Salida de ejemplo

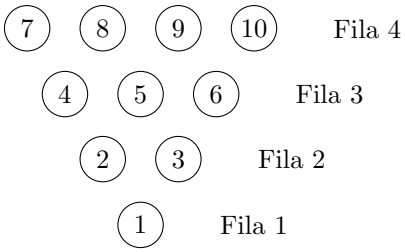
```
22
8
```

D
Tirando bolos

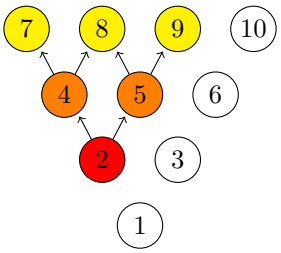
Los "*bolos*" son un juego tradicional en el que los jugadores tratan de derribar el mayor número de *bolos* posible utilizando, normalmente, una bola.

La cantidad de variantes del juego es inmensa, cada una con sus propias normas particulares. La colocación de los bolos, su número, y el objeto que se lanza varía enormemente. En ocasiones, incluso el objetivo cambia; en ciertas modalidades no sólo es importante tirar muchos bolos, sino también conseguir *desplazarlos lo más posible* desde su posición original.

La variante más conocida es el *bowling*, que se juega en lugares cerrados y en el que la bola se desliza por un suelo encerado, en lugar de lanzarla al aire. Al final de la pista, de algo más de 19 metros de largo, se colocan 10 bolos según la disposición siguiente (vista desde arriba):



Al golpear un bolo, éste cae y puede derribar a otros. En concreto, para este problema, supondremos que cuando un bolo cae, golpea (y tira) los dos bolos que tiene justo detrás, y éstos hacen lo propio con los siguientes[2]. Por ejemplo, si la bola golpea el bolo número 2, éste hará caer a los bolos 4 y 5, y éstos a su vez a los bolos 7, 8 y 9:



Dado que lo importante es no dejar bolos en pie, ¿cuántos nos dejamos al golpear un bolo?

Entrada

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

Cada caso de prueba se compone de dos números, f y b. El primero, f, indica el número de *filas* de bolos; cada fila tiene un bolo más que la fila anterior. El segundo número, b, indica el número de la fila en la que está el bolo que se golpea, que se considera siempre situado en un extremo. Se garantiza que $1 \leq b \leq f < 2^{31}$.

La entrada termina con un caso en el que no hay ninguna fila de bolos.

Salida

Para cada caso de prueba se escribirá, en una línea independiente, el número de bolos que quedan de pie tras tirar el bolo indicado, y que éste tire a todos los que tiene detrás. Se garantiza que la salida será siempre menor que 2^{31}.

[2] Cualquier parecido con el juego real es pura coincidencia.

Entrada de ejemplo

```
4 2
4 4
3 1
0 0
```

Salida de ejemplo

```
4
9
0
```

E

Pánico en el túnel

En el último Consejo de Ministros se ha decidido que se va a poner en marcha la construcción de un túnel que va a unir la península ibérica con las islas Baleares. Si se pudo unir Europa con Inglaterra el siglo pasado, el gobierno tiene claro que nosotros podremos hacer un túnel similar, como experiencia piloto para un proyecto aún más ambicioso que uniría también las islas Canarias.

Uno de los temas que más preocupan es la seguridad, por lo que se instalarán un número aún por determinar de teléfonos de emergencia (eso sí, nunca serán más de 500) de forma que ante algún accidente la gente pueda salir rápidamente de sus coches y notificarlo en el teléfono más cercano.

No obstante existe el problema de que los accidentados, a la hora de salir de sus vehículos y echar a correr hacia ese teléfono, duden sobre hacia qué lado ir pues en esos momentos de pánico se pierde la capacidad de decisión y no se tiene la agilidad mental suficiente para elegir bien.

De lo que se trata, pues, es de hacer una pequeña aplicación para móvil que se pueda usar dentro del túnel y que, dada la localización actual de la persona, le diga hacia qué lado tiene que correr, si hacia el lado que le acerca hacia la península o hacia el lado que le acerca a las islas.

Como el túnel está aún por construir, crearemos y probaremos el *núcleo* del sistema utilizando distintas descripciones de túneles. Cada descripción consiste en discretizar el túnel en distintas secciones en las que puede (o no) haber un teléfono de emergencia.

Entrada

La entrada estará compuesta por varios casos de prueba. Cada caso de prueba tendrá una primera línea con la descripción del túnel. Esa línea tendrá un carácter por cada sección del túnel (que puede ser de hasta 1.000.000 secciones) donde '`T`' indica que en esa sección hay un teléfono, y '`.`' indica ausencia de teléfono. El primer carácter de la línea se corresponde con la sección "del lado de la península".

Tras la descripción del túnel, aparecerá un número n con la cantidad de consultas que vienen a continuación. Le siguen n líneas, cada una con un entero que indica en qué sección se encuentra el accidentado (entre 1 y el número de secciones total, siendo 1 la primera sección en la ruta desde la península hacia las islas).

Salida

Por cada caso de prueba se escribirán n líneas, una por consulta, donde se indicará "`PENINSULA`" si el accidentado debe correr hacia el lado de la península o "`ISLAS`" si debe ir en la dirección contraria. Si no hace falta correr porque estamos en una sección con teléfono, se escribirá "`AQUI`".

Ten en cuenta las siguientes reglas:

- Si se está más cerca de una de las salidas del túnel que de un teléfono, se preferirá correr hacia la salida.
- Si los teléfonos más próximos de ambos lados están a la misma distancia, preferiremos ir hacia aquel que esté más cerca de una de las salidas del túnel.
- Si en el escenario anterior vuelve a haber empate, se irá hacia la península.

Entrada de ejemplo

```
.T...T.T.
4
2
3
4
7
...
1
2
```

Salida de ejemplo

```
AQUI
PENINSULA
PENINSULA
ISLAS
PENINSULA
```

F
Primo de riesgo

Para las máquinas, las palabras no son más que sucesiones de caracteres. Esos caracteres, a su vez, no son más que números que, gracias a la codificación utilizada, son interpretados como letras, signos de puntuación o cualquier otro símbolo. Una de las codificaciones más extendidas y que sigue utilizándose hoy en día (aunque sea como *subconjunto* de las codificaciones utilizadas) es la codificación ASCII. De hecho, en la mayoría de los lenguajes de programación actuales, todas las letras del alfabeto inglés y los símbolos más utilizados (espacio, punto, coma, ...) cuando están almacenadas en una variable de tipo carácter pueden convertirse a un número entero, consiguiendo así su código ASCII.

Dado, pues, que las letras de una cadena pueden convertirse a *números*, podemos hacer la *suma* de todos los caracteres para obtener el *número de la cadena completa*.

Así, la codificación de cada carácter de, por ejemplo, la palabra "`riesgo`" es

Carácter	Código ASCII
`r`	114
`i`	105
`e`	101
`s`	115
`g`	103
`o`	111

Lo que hace que la codificación completa de `riesgo` sea:

$$114 + 105 + 101 + 115 + 103 + 111 = 649$$

Lo que nos preguntamos es, dado ese número, ¿cuál es el número primo inferior más cercano?[3]

Entrada

La entrada comienza con un número que indica la cantidad de casos de prueba que siguen. Cada uno de estos casos de prueba es una palabra con letras del alfabeto inglés de no más de 50 caracteres.

Salida

Para cada caso de prueba se mostrará una única línea en la que se indicará el número primo inferior más cercano al número obtenido a partir de la palabra según la descripción anterior.

Entrada de ejemplo

```
3
A
i
riesgo
```

Salida de ejemplo

```
61
103
647
```

[3]Recuerda que un número primo es aquel que sólo puede dividirse por él mismo y por 1.

G
Aprendiendo el código Morse

Todos hemos oído hablar del código o alfabeto Morse, que antiguamente servía para transmitir mensajes de telégrafo. El código consiste en la codificación de cada letra del abecedario con una sucesión de puntos y rayas que se traducen a señales auditivas cortas (puntos) o largas (rayas), siguiendo las transformaciones que se indican en la tabla.

Letra	Código
A	`.-`
B	`-...`
C	`-.-.`
D	`-..`
E	`.`
F	`..-.`
G	`--.`
H	`....`
I	`..`
J	`.---`
K	`-.-`
L	`.-..`
M	`--`

Letra	Código
N	`-.`
O	`---`
P	`.--.`
Q	`--.-`
R	`.-.`
S	`...`
T	`-`
U	`..-`
V	`...-`
W	`.--`
X	`-..-`
Y	`-.--`
Z	`--..`

El código, no obstante, es bastante complicado de aprender y utilizar. Por una parte hay que aprenderse los códigos de cada letra. Por otra, hay que añadir pausas entre cada símbolo, al existir codificaciones de letras que son prefijos de otras, y pausas más largas entre cada palabra, pues el "espacio" no tiene ningún código asociado.

Una guía de ayuda para aprenderse la tabla de codificación consiste en tener una palabra de referencia para cada letra. Así, por ejemplo, para la letra '`A`' podemos memorizar *Arco*. La palabra elegida para cada letra debe comenzar por esa letra y ser tal que si cada vocal '`o`' se sustituye por una raya, y el resto de vocales por un punto, el resultado final sea la codificación de la letra en cuestión.

A continuación aparecen algunos ejemplos de palabras que pueden utilizarse como palabras de referencia:

Letra	Palabra de referencia	Código
A	Arco	`.-`
B	Bogavante	`-...`
C	Corazones	`-.-.`

Ahora estamos haciendo una tabla nueva y tenemos que comprobar si, dada una palabra, podemos o no utilizarla como palabra de referencia.

Entrada

La entrada consiste en una serie de palabras de no más de 20 letras, cada una en una línea independiente. Cada palabra contendrá únicamente símbolos del alfabeto inglés ya sea en mayúscula o en minúscula. Las palabras no contendrán tildes (aunque eso implique no escribir correctamente la palabra).

Salida

Para cada palabra se mostrará en una línea independiente si puede ser utilizada como palabra de referencia según la descripción dada. En caso afirmativo se escribirá la palabra leída seguida de "`OK`". En caso negativo, tras la palabra se escribira "`X`".

Entrada de ejemplo

```
Arco
Corazones
ARBOLES
```

Salida de ejemplo

```
Arco OK
Corazones OK
ARBOLES X
```

H
Operación asfalto

Las calles de Eulerandia están en un estado lamentable. El ayuntamiento ha decidido poner en marcha su famosa *operación asfalto* y arreglar todas las calles de una vez. Para eso ha alquilado una de esas grandes y pesadas máquinas asfaltadoras. En concreto, la que utilizarán es capaz de asfaltar ambas direcciones de la calle simultáneamente con solo recorrerla de principio a fin en un sentido cualquiera.

Se da la paradoja de que la máquina asfaltadora es tan grande y pesada que, una vez asfaltada una calle, si la máquina volviera a pasar por ella, destrozaría el asfalto que ella misma puso.

Esa circunstancia complica la planificación de la operación asfalto, pues hay que ver si es posible diseñar una ruta de la máquina que recorra toda la ciudad sin pasar dos veces por ninguna calle.

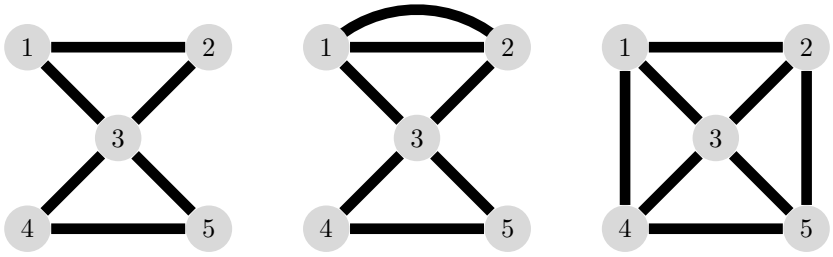


Figura 1: ejemplos de ciudades

En la figura 1 aparecen tres ejemplos de ciudades distintas. Si Eulerandia tuviera la forma de la ciudad 1, se podría llevar la asfaltadora desmontada hasta cualquiera de las cinco intersecciones y, tras el montaje, ésta podría asfaltar toda la ciudad volviendo al punto de partida. Con unas calles como las de la ciudad 2, sin embargo, sólo se podría asfaltar toda la ciudad si se empezara en la intersección 1 y se terminara en la 2 (o al contrario). Por último, la ciudad 3 no podría asfaltarse por completo sin hacer que la asfaltadora pase por alguna calle más de una vez. La figura 2 muestra posibles recorridos sobre las tres ciudades.

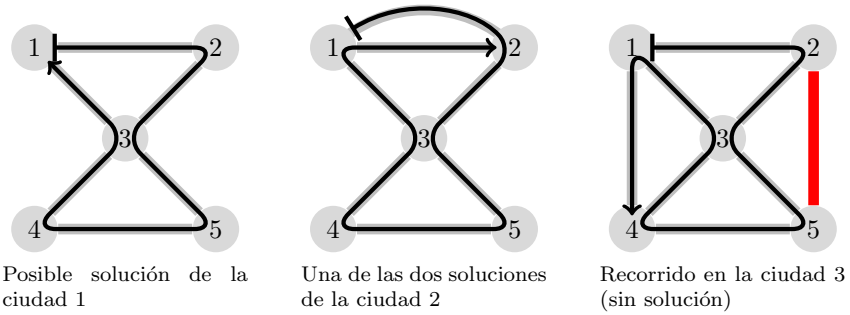


Figura 2: soluciones a las ciudades de la figura 1

Se trata, pues, de analizar el plano de la ciudad y ver si se podrán asfaltar todas sus calles con las condiciones anteriores o no (teniendo en cuenta que la asfaltadora puede empezar en un extremo de una calle y terminar su trabajo en otro de una calle distinta).

Entrada

La entrada está compuesta de varias descripciones de ciudades. Cada ciudad comienza con dos líneas, la primera con el número de calles y la segunda con el número `I` de intersecciones ($1 \leq I \leq 50$), que estarán

numeradas de 1 a I (entendemos por *intersección* a los puntos donde comienzan o terminan calles; eso incluye también los extremos de calles unidas con otras). A continuación aparece una línea por cada calle indicando las intersecciones que une. Ten presente que algunas calles pueden en realidad ser túneles o tener puentes y pasar unas por encima de otras. El último caso de prueba es seguido de una línea con un `0`.

Se garantiza que desde cualquier calle se puede ir a cualquier otra.

Salida

Se escribirá una línea por cada ciudad, con un `SI` si ésta se puede asfaltar según las condiciones descritas y `NO` en caso contrario.

Entrada de ejemplo

```
6
5
1 2
1 3
2 3
3 4
4 5
3 5
7
5
1 2
1 3
2 3
3 5
2 1
3 4
4 5
8
5
1 2
1 3
2 3
1 4
2 5
3 5
3 4
4 5
0
```

Salida de ejemplo

```
SI
SI
NO
```

● I
Abono de temporada

Con el auge de las urbanizaciones con piscina privada y pistas de paddle, la afluencia de bañistas a las piscinas públicas ha disminuido considerablemente en los últimos años.

Por esta razón, estas instalaciones se están esforzando para volver a atraer a esos usuarios que hoy por hoy parecen preferir una pequeña piscina compartida por un montón de vecinos en vez de grandes piscinas en las que poder disfrutar todo el día.

Para ello están utilizando los llamados *abonos de temporada*. Cualquier usuario que adquiera uno de estos abonos, tiene derecho a entrar y salir de las instalaciones siempre que quiera. Eso incluye, incluso, poder ir a la piscina por la mañana un rato, volver a casa, y regresar por la tarde a darse el último chapuzón, lo que permite a los usuarios mantener el mismo patrón de uso que ahora hacen en las piscinas de sus urbanizaciones.

Eso sí, para evitar morir de éxito y que la piscina termine plagada de bañistas quejándose por la falta de espacio, han decidido añadir una clausula que dice "el uso del abono está limitado a 15 días dentro de cualquier periodo de 20 días consecutivos". Es decir que al ir a entrar a la piscina comprueban antes tu registro de los últimos 20 días y si ya has ido 15, no te dejan pasar.

Como propietario de una casa sin piscina privada, te estás planteando comprar el abono de una de las piscinas que tienes cerca de casa. Ahora te toca determinar, para cada una de las ofertas, cuántos días podrías ir a esa piscina si te compras el abono justo antes de que comience la temporada.

Entrada

Cada caso de prueba contiene la información de una oferta constituida por tres enteros: el número de días que tiene la temporada (T), la longitud del periodo (P) y el número máximo de días de uso durante ese periodo (D). Se garantiza que $1 \leq D \leq P \leq T$.

Los casos de prueba terminan con una línea con los tres ceros, que no debe procesarse.

Salida

Para cada caso de prueba se mostrará una única línea con el número de días como máximo que se podrá ir a la piscina en toda la temporada.

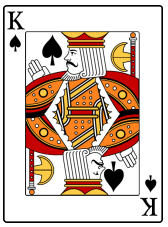
Entrada de ejemplo

```
30 20 15
25 20 15
0 0 0
```

Salida de ejemplo

```
25
20
```

J
Abdicación de un Rey



Cuando un rey abdica, su primogénito hereda el trono y debe recibir, en su coronación, un número que lo identificará para la posteridad. La numeración es importante porque, de otro modo, sería difícil diferenciar a reyes con el mismo nombre de una misma dinastía al compartir también apellido.

El resultado es que ante la abdicación de un rey, toca revisar los libros de historia para averiguar su número. ¿Eres capaz de hacerlo tú?

Entrada

El programa recibirá, por la entrada estándar, múltiples casos de prueba. Cada uno consta de una primera línea con un número indicando la cantidad de reyes de una determinada dinastía. A continuación vendrá, en otra linea, los nombres de todos sus reyes separados por espacio.

Después aparecerán dos líneas más, una con la cantidad de sucesores futuros que hay que numerar (al menos uno), y otra con sus nombres separados por espacio.

Todos los nombres estarán compuestos de una única palabra de no más de 20 letras del alfabeto inglés, y serán sensibles a mayúsculas. Además, se garantiza que en cada caso de prueba no habrá más de 20 nombres de reyes diferentes.

El último caso de prueba, que no deberá procesarse, contendrá un `0` como número de reyes de la dinastía.

Salida

Para cada sucesor de cada caso de prueba se indicará, una línea independiente, el número que le corresponderá. Aunque normalmente se utilizan números romanos, por simplicidad se indicará el número en la notación arábiga tradicional.

Después de cada caso de prueba se escribirá una línea en blanco.

Entrada de ejemplo

```
11
Felipe Carlos Felipe Felipe Felipe Carlos Felipe Carlos Alfonso Alfonso JuanCarlos
3
Felipe Leonor Felipe
12
Carlos Isabel Carlos Jorge Jorge Jorge Jorge Guillermo Victoria Jorge Jorge Isabel
3
Carlos Guillermo Jorge
0
```

Salida de ejemplo

```
6
1
7

3
2
7

```