Concurso Regional Programame Zaragoza

10 de Abril de 2025

Cuaderno de Problemas

Índice de problemas

Ejercicios realizados por:

- Antonio Pérez-Aradros
- Santiago Faci

A. ASCENSORES EN VALDEBEBAS

Después de muchos años de esfuerzo y horas de entrenamiento en La Fábrica[1], por fin me he ganado un puesto en la cantera del Real Madrid. ¡Es un sueño hecho realidad!

Para estar más cerca de la Ciudad Deportiva, me he mudado a un moderno bloque de pisos en Valdebebas, donde viven otros compañeros del equipo.

La urbanización es una chulada: tiene portero, gimnasio, piscina, pista de pádel, … y ascensores, claro. Estos funcionan de una manera muy curiosa:

- Los ascensores están numerados empezando por la izquierda del 1 al N
- Siempre hay un ascensor esperando en la planta baja (piso 0), listo para recibir a cualquiera que regrese de entrenar.
- Si llamas a un ascensor desde un piso cualquiera, el ascensor más cercano es el que acude a recogerte. En caso de que haya dos o más ascensores igual de próximos, acude el que tiene un número inferior
- Si el ascensor que responde es el de la planta baja, como siempre tiene que haber uno esperando, el ascensor más próximo acude a esa planta 0, porque aquí el trabajo en equipo es clave (¡igual que en el fútbol!).

Me pregunto cómo habrán hecho para que los ascensores sean tan “listos”

Entrada

La entrada es una serie de casos de prueba. Cada uno de ellos consta de dos líneas.

La primera son dos números: N (2 <= N <= 10) con el número de ascensores que hay en el edificio y P (-2 <= P <= 100) que indica el piso desde el que se llama al ascensor.

En la segunda aparecen N números (entre -2 y 100) indicando el piso en el que se encuentra cada uno de los ascensores antes de la llamada.

Salida

Para cada caso de prueba, el programa deberá escribir la posición en la que se encuentran los N ascensores después de la llamada que se ha realizado desde el piso P

Entrada de ejemplo

```
3 7
0 2 3
3 1
-1 0 7
5 10
3 6 0 14 90
5 1
-1 0 2 5 -1
```

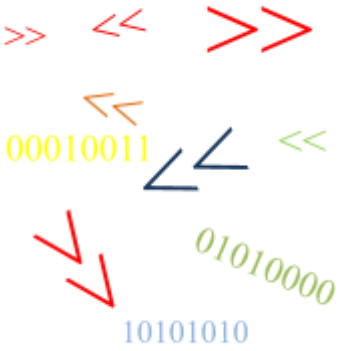
```
10 1
-2 -1 0 1 2 3 4 5 6 7
```

Salida de ejemplo

```
0 2 7
0 1 7
3 10 0 14 90
0 1 2 5 -1
-2 -1 0 1 2 3 4 5 6 7
```

1 *La Fábrica* es como se conoce popularmente a la cantera del R.Madrid

B. DESPLAZANDO BITS



Hoy, en clase de Pensamiento Computacional, nos ha explicado el profe los operadores de desplazamiento de bits (<< y >>). Me han parecido interesantísimos. Sirven, entre otras cosas, para hacer más eficientes la multiplicaciones y divisiones por potencias de 2. Así, por ejemplo, multiplicar un número por 2 es equivalente a desplazar una posición a la izquierda sus bits. Dividirlo por 4 sería desplazar los bits hacia la derecha 2 posiciones. Nos ha puesto algunos ejemplos.

En decimal: 5 * 2 = 10. En binario: 00000101 << 1 = 00001010

En decimal: 6 * 4 = 24. En binario: 00000110 << 2 = 00011000

En decimal: 24 / 4 = 6. En binario: 00011000 >> 2 = 00000110

También podemos emplearlos cuando se multiplica por otras cantidades no potencias de 2. Por ejemplo, N * 10 se puede poner como N * (8 + 2) = N * 8 + N * 2, es decir (N << 3) + (N << 1). O, por ejemplo, N * 7 se puede poner como N * (4 + 2 + 1) = N * 4 + N * 2 + N * 1, es decir (N << 2) + (N << 1) + (N << 0). Bueno, pues ésta es la tarea que nos ha puesto para mañana el profe: encontrar los desplazamientos que tenemos que hacerle al número para simular la multiplicación que se nos pide.

Entrada

La entrada comienza con un primer número que indica cuántos casos de prueba deberán ser procesados.

Cada caso de prueba es un número M (1 <= M <= 1000000) que indica el número por el que queremos multiplicar N. El resultado de la multiplicación nunca será superior a 10^9.

Salida

Para cada caso de prueba se escribirán, ordenados de mayor a menor, todos los desplazamientos que hay que hacer de forma que, una vez sumados, simulen la multiplicación N x M.

Entrada de ejemplo

```
4
3
7
8
10
```

Salida de ejemplo

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

C. DIGITÓPOLIS

En el reino de Digitópolis, el Rey Númerus está preparando una gran fiesta para celebrar su centésimo décimo primer cumpleaños. Para la ceremonia se le ha ocurrido la brillantísima idea de numerar a todos sus súbditos. Quiere que cada habitante de su reino lleve tatuado un número que lo identifique. Empezarán en el 1 y se irán asignando consecutivamente a cada uno de los habitantes. El rey llevará el 0, faltaría más (todo el mundo sabe que el 0 es el número más grande que existe)

No me preguntes cómo, pero hemos tenido acceso a la conversación del Rey con su alquimista, Mágicus. He aquí una parte de dicha conversación.

- *Rey: ¿Te ves capaz de tatuar un número a cada uno de mis súbditos?*
- *Mágicus: Naturalmente, Majestad. Pero va a resultar extremadamente costoso.*
- *Rey: Eso no es un problema. ¿De cuánto estamos hablando?*
- *Mágicus: Tengo que echar las cuentas, pero sí le puedo anticipar que el precio de la tinta que empleo depende del número que escriba.*
- *Rey: Explícate mejor, diantres.*
- *Mágicus: Quiero decir que cada dígito que escribo cuesta el valor que indica. Así, el 1 vale 1, el 2 vale 2, el 12 vale 3 (1 + 2), el 123 vale 6 (1 + 2 + 3) y así todos los demás. ¿Me entiende ahora mejor, su majestad?*
- *Rey: Malditos magos, sois todos iguales. Dime una cifra y no me cuentes patrañas.*

Anda, échale una mano a Mágicus y calcula cuánto costaría tatuar a todos los habitantes de Digitópolis.

Entrada

El programa deberá leer, de la entrada estándar, múltiples casos de prueba, cada uno ocupando una única línea.

En cada línea aparece un número N (1 <= N <= 100000) indicando el número de habitantes del reino, sin contar al Rey, que, como hemos dicho antes, lleva el número 0.

La última línea es un cero que no debe procesarse.

Salida

Para cada caso de prueba, el programa escribirá la suma total que cuesta tatuar a todos los súbditos del Rey.

Entrada de ejemplo

```
3
5
10
20
```

```
0
```

Salida de ejemplo

```
6
15
46
102
```

D. HUYENDO DEL ENEMIGO

Los soldados de la XII Legión romana han logrado escapar del asedio enemigo y se encuentran en la ribera de un ancho río. Al otro lado les espera un terreno seguro donde podrán reorganizarse y continuar la lucha. Sin embargo, el enemigo se acerca rápidamente, y la única forma de cruzar el río es utilizando una pequeña barca que han encontrado en la orilla.

La barca es frágil y solo puede transportar, en cada viaje, a un máximo de dos soldados a la vez. Además, tiene una capacidad de carga limitada: si el peso combinado de los soldados en un viaje excede cierto umbral, la barca se hundirá.

El tiempo apremia, y los soldados necesitan cruzar lo más rápido posible antes de ser alcanzados. Tu misión es calcular el número mínimo de viajes necesarios para que todos los soldados logren cruzar sanos y salvos al otro lado del río.

Entrada

La entrada consiste en varios casos de prueba. Cada uno consta de dos líneas.

La primera línea contiene dos enteros: N (1 <= N <= 1000000) indicando el número de soldados que han escapado y P, que representa el peso máximo que puede soportar la barca sin hundirse. La unidad de peso que se utilizaba en las legiones romanas se ha perdido, pero se sospecha que era muy precisa, así que P está entre 1 y 2000000000.

La segunda línea contiene N números con los pesos de cada uno de los soldados (1 <= N_i <= 2000000000). En la legión no importaba el peso del soldado, solo su bravura y su capacidad para cumplir órdenes.

El último caso de prueba es una única línea con 0 0 que no debe procesarse.

Salida

Para cada caso de prueba, el programa escribirá el mínimo número de viajes que son necesarios para cruzar a todos los soldados a la otra orilla. Ah, ningún soldado romano sabía nadar, así que…

Si no es posible salvar a todos los soldados, se escribirá IMPOSIBLE

Entrada de ejemplo

```
10 10
2 2 2 2 2 2 2 2 2 2
10 10
1 2 3 4 5 6 7 8 9 10
10 10
20 20 20 20 20 20 20 20 20 20
10 10
3 5 8 9 7 3 4 2 10 10
0 0
```

Salida de ejemplo

```
5
6
IMPOSIBLE
7
```

E. POR QUÉ PIERDO AL PÁDEL

Desde que te mudaste a tu nuevo barrio, has encontrado en el pádel una forma perfecta de hacer amigos y mantenerte en forma. Sin embargo, después de cada partido perdido (que, siendo sinceros, son muchos), te queda una sensación extraña. Algo dentro de ti te dice que la culpa no es solo tuya...

Después de un profundo análisis, has identificado los principales factores que afectan a tus resultados:

- El 54% de la culpa es de tu compañero o compañera. No importa quién sea, siempre falla en los momentos clave, deja pasar bolas fáciles o simplemente no cubre su parte de la pista.
- El 12% se debe a la pista. A veces hay viento, otras veces el bote no es el esperado, o incluso hay pequeñas irregularidades en el suelo que solo parecen afectarte a ti.
- El 25% es por culpa de la pala. O pesa demasiado, o tiene poco agarre, o su balance no es el adecuado. Da igual cuál uses, siempre le pasa algo.
- El 8% es por la dichosa pelota. Hay unas que botan una barbaridad, otras no botan nada. Así no hay quién se aclare

Y ya está, así que el 1% de las veces tienes que reconocer que es culpa tuya. A veces no das lo mejor de ti mismo, es verdad.

Entrada

La entrada comienza con un número indicando cuántos casos de prueba tendrán que procesarse. Cada uno de ellos consta de una línea con 4 números: el porcentaje de culpa que se atribuye al compañero/a, a la pista, a la pala y a la pelota. Todos ellos son números enteros entre 1 y 90 y la suma de todos ellos siempre es menor que 100.

Salida

Para cada caso de prueba, el programa deberá escribir el porcentaje que tienes tú de culpa.

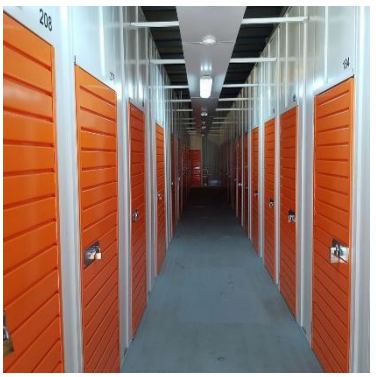
Entrada de ejemplo

```
4
54 12 25 8
10 10 10 10
25 20 20 1
50 20 20 5
```

Salida de ejemplo

```
1
60
34
5
```

F. LA HERENCIA DE LA ABUELA



Tu abuela Eusebia se ha empeñado en darte ya su herencia. Su legado, claro, no son coches ni apartamentos ni mucho menos unos cuantos bitcoins. Lo que vas a heredar es un montón de "recuerdos de valor sentimental". Tú, con la mejor intención, has aceptado, pero ahora te enfrentas a un problema: tu trastero tiene un espacio limitado, y debes decidir si puedes guardar todo lo donado o si tendrás que rechazar algo (con el consiguiente drama familiar).

La abuela ya está planeando su próxima visita, y si descubriera que no has guardado todo lo que te dio, podrías verte obligado a escuchar durante horas la historia de cada cachivache rechazado. Desde aquel viejo reloj de pared que "sobrevivió a la Guerra Civil" hasta la caja de botones que "seguro que algún día te servirán para algo", cada objeto tiene un trasfondo emocional difícil de ignorar. Si no eres lo suficientemente eficiente, prepárate para una sesión intensiva de anécdotas familiares y sopa de fideos con extra de reproches.

Así que debes actuar con estrategia. Tu trastero está dividido en huecos de distintos tamaños, y cada vez que intentas guardar una cosa, buscas el hueco más grande disponible para colocarla (si hay más de uno igual de grande, cualquiera de ellos sirve). Si el trasto cabe, reduces el tamaño del hueco en la cantidad ocupada. Si no cabe en ningún hueco, habrás fracasado en tu misión. Se trata, por tanto, de ver si eres capaz, siguiendo la estrategia elegida, de guardar todos los regalos de la abuela

Entrada

La entrada consiste en varios casos de prueba. Cada uno consta de cuatro líneas:

La primera es un número N (1 <= N <= 100000) con el número de huecos de que dispone el trastero. En la segunda línea aparecen N números con los tamaños de los huecos. Como utilizo una medida muy precisa, son números entre 1 y 1000000. La tercera línea es un número M (1 <= M <= 100000) con el número de cosas que me ha dejado la abuela. A continuación, aparecen los M tamaños de las cosas, números entre 1 y 1000000.

Salida

Para cada caso de prueba, el programa deberá escribir SI o NO dependiendo de si se han podido guardar o no todos los objetos siguiendo la estrategia descrita en el enunciado.

Entrada de ejemplo

```
5
10 20 30 40 50
2
40 40
5
10 30 10 10 50
3
20 30 30
5
10 30 10 10 50
4
20 20 20 15
```

Salida de ejemplo

```
SI
SI
NO
```

G. EL PORCENTAJE DEL PANADERO

Todo panadero que se precie conoce bien lo que es “el porcentaje del panadero”, que es la forma en la que se calculan las proporciones correctas de cada ingrediente para la preparación de cualquier pan. Cómo no, la harina es siempre la protagonista y (casi) todo gira en torno a ella.

La idea de este sistema de medida es la siguiente: todos los ingredientes se indican en porcentajes sobre el total de harina empleada, excepto el de la harina, que se supone siempre como el 100% del total. Ese total se calculará sumando los porcentajes del resto de ingredientes al de la harina. Así, si decimos que un pan tiene un 60% de agua, 1% de sal y 2% de levadura, la fórmula total del pan será de un 163% (el indicado + el 100% de harina). Y ese porcentaje será la referencia para considerar el peso total del pan y sobre el que empezaremos calculando el peso de la harina para, a partir del mismo, calcular el del resto de ingredientes en base a sus porcentajes.

La verdad es que sería súper cómodo que todo panadero tuviera una calculadora que le hiciera todos esos cálculos cómodamente.

Entrada

Como entrada se recibe una primera línea que indica cuántas recetas de pan deben procesarse.

A continuación se recibe cada receta en grupos de tres líneas donde la primera será el peso total del pan. En la segunda línea de la receta se reciben los nombres de los ingredientes necesarios para preparar el pan (excepto la harina que se considera esencial y no viene aquí indicada) y, como última línea, se reciben los porcentajes para cada uno de esos ingredientes (excepto el de la harina, que se considera siempre del 100%).

Salida

Como salida se devolverá, cada para cada receta, la cantidad necesaria en gramos de cada uno de los ingredientes (incluida la harina). Se indicará la harina en primer lugar y el resto de ingredientes en el mismo orden en que fueron proporcionados en la entrada, ocupando una sola línea por receta.

(las cantidades en gramos de todos los ingredientes se tomarán siempre como valores enteros, tanto para los cálculos como para la salida, truncando el número cuando sea necesario, y teniendo en cuenta que la cantidad mínima de un ingrediente será siempre de 1 gramo)*

Entrada de ejemplo

```
3
100
sal levadura agua semillas
1 2 50 1
1200
sal levadura agua
1 1 48
2000
agua sal levadura semillas
40 2 2 1
```

Salida de ejemplo

```
harina 64 sal 1 levadura 1 agua 32 semillas 1
harina 800 sal 8 levadura 8 agua 384
harina 1379 agua 551 sal 27 levadura 27 semillas 13
```

H. MUCHO POR LEER

En este año se han propuesto potenciar la lectura en el colegio donde estudia Diana. Les han preparado una lista de forma que tengan que leer un libro cada semana. Cada estudiante puede organizarse como quiera siempre y cuando el domingo de cada semana haya terminado el libro que corresponda.

Diana es una estudiante muy organizada y curiosa y ha ido apuntando, para cada libro, donde empezaba y dejaba de leer cada uno de los días de la semana. Y ahora quiere saber si hay algún día de la semana en el que suele leer más. Así que quiere revisar todas esas anotaciones para ver si algún día se repite más veces.

Como son muchos los libros que ha leído este año, ¿qué tal si le ayudamos con un pequeño programa que nos indique en qué día de la semana leyó más páginas para cada uno de los libros?

Entrada

Como entrada se recibirá, en cada línea, cómo ha llevado Diana su lectura. Se recibe una secuencia de números que indican en qué página empezó y dónde dejó de leer Diana para cada día de la semana, teniendo en cuenta que nunca lee esa última página de cada día porque será la que empezará leyendo al día siguiente (excepto para el último día, que será la última página que lea). El día límite es el domingo, pero hay semanas en las que puede terminar de leer antes.

La entrada termina con un 0

Salida

Como salida se mostrará, para cada uno de los casos de entrada, el día de la semana en el que Diana ha leído más páginas. En caso de empate, se tomará como referencia el primer día en que leyó dicha cantidad.

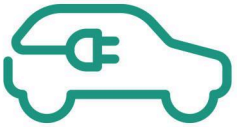
Entrada de ejemplo

```
1 4 9 25 50 54 54 100
1 100
1 10 20 30 100 120 130 150
1 10 20 30 40 50 60 70
1 50 60 70 80 90 100 108
0
```

Salida de ejemplo

```
domingo
lunes
jueves
domingo
lunes
```

I. VIAJANDO EN COCHE ELÉCTRICO



Últimamente se están poniendo de moda los coches eléctricos y Antonio no se quiere quedar atrás. Se ha comprado el último modelo y ahora, cuando se dispone a viajar de verdad, se ha dado cuenta de que necesita planear los viajes más que antes porque estos coches eléctricos no son tan fáciles de recargar. No hay muchos puntos de recarga y conviene estudiar la ruta antes de salir para saber bien dónde y cuándo recargar.

Antonio tiene planeado varios viajes y ha conseguido, para cada uno de ellos, trazar la ruta de forma que sabe en qué punto kilométrico hay un punto de carga. Antonio, que es muy organizado, se ha propuesto planear en qué puntos de recarga debería parar con el objetivo de hacerlo el menor número posible de veces. Hay que tener en cuenta que la autonomía de su coche es solamente de 300 kilómetros. Esa será la distancia máxima que puede recorrer entre dos puntos de conexión.

Algo muy importante para Antonio es detectar cuando alguna ruta sea imposible porque dos puntos de carga estén demasiado separados entre sí. No quiere acabar tirado en mitad de la nada con su nuevo coche.

Entrada

Como entrada se recibe una primera línea que contiene el número de rutas a analizar. A partir de ahí, cada ruta se define en 2 líneas: La primera indica la duración total del viaje. La segunda empieza con un valor que indica el número de puntos de recarga de la ruta (que puede ser igual a 0) y, a continuación, los puntos kilométricos de cada uno de ellos, si los hay.

Se asume siempre que el origen de la ruta es la casa de Antonio, donde se ha instalado un punto de recarga, por lo que se puede suponer que el coche está completamente cargado cuando se pone en marcha. Esa recarga, además, no debe figurar en la salida del programa.

Salida

La salida mostrará, para cada caso, una línea con todos los puntos kilométricos en los que habrá que recargar para llegar al final del viaje. Si no fuera necesario ninguna parada durante el viaje, se devolverá `SIN PARADAS`. Si la ruta es inviable porque en algún momento de la ruta dos puntos de conexión están demasiado separados, se devolverá solamente `RUTA IMPOSIBLE` para ese caso.

Entrada de ejemplo

```
5
2000
10 100 400 700 800 950 1100 1300 1500 1600 1900
1000
5 100 200 300 400 900
1300
10 100 200 300 400 500 600 700 800 900 1000
400
0
200
2 100 150
```

Salida de ejemplo

```
100 400 700 950 1100 1300 1600 1900
RUTA IMPOSIBLE
300 600 900 1000
RUTA IMPOSIBLE
SIN PARADAS
```

J. LA MEJOR PANTALLA

Diana quiere comprar una de esas retro-consolas que permiten ejecutar múltiples emuladores con cientos de juegos cada uno.

Ahora está toda preocupada por si todos esos videojuegos de diferentes emuladores podrán verse bien en una pantalla determinada. Cada emulador tiene su relación de aspecto, pero ella tiene que decidirse por una máquina concreta con una relación determinada y jugar con ella a todos esos videojuegos.

La relación de aspecto de una pantalla, independientemente de la resolución que tenga, nos indica la relación entre el ancho y el alto de la misma. Por ejemplo, si decimos que la relación de aspecto es 4:3, estamos indicando cuánto de ancho será la pantalla (4) con respecto a su altura (3).

Así que Diana no para de investigar por Internet cuáles son esas resoluciones y se encuentra con foros donde los usuarios expresan la misma resolución utilizando diferentes escalas. Por ejemplo, usan 3:3, 4:4 o 2:2 para referirse a una misma relación de aspecto, en este caso 1:1. En este caso resulta fácil, pero imagínate cuando le dicen que la relación es 80:45. Tiene que andar haciendo cálculos para finalmente darse cuenta de que es la que todos conocemos como 16:9. Es por eso que ha decidido preparar un pequeño programa al que le pueda pasar todas esas resoluciones "extrañas" que encuentra por ahí para poder tener la relación equivalente por las que se les conoce normalmente, reduciendo la escala lo máximo posible para mantener siempre dos números enteros.

Entrada

Como entrada se comienza recibiendo una línea que indicará el número de casos de prueba que se deberán procesar. Cada una de las siguientes líneas representa un caso de prueba, que será una resolución expresada como `ANCHO:ALTO` donde `1 <= ANCHO <= 10000` y `1 <= ALTO <= 10000`

Salida

Como salida se recibirá, para cada uno de los casos de entrada, la resolución equivalente de forma que Diana pueda compararlas todas sin andar haciendo cálculos.

Entrada de ejemplo

```
9
2366:273
16:9
80:45
21:9
2:3
3:2
4:4
96:27
8:6
```

Salida de ejemplo

```
26:3
```

```
16:9
16:9
7:3
2:3
3:2
1:1
32:9
4:3
```

K. REGAR LAS PLANTAS

Ya estamos en plena primavera y, aunque está lloviendo bastante, todo el mundo está muy atento de cuidar su jardín y regar las plantas cuando toca. Pero uno no siempre está en casa para poder hacerlo. Y al final, entre eso y lo ocupados que estamos siempre por una cosa o por otra, te olvidas de regar cuando debes.

Diana es muy maja y le gusta siempre ayudar. Ha decidido echar una mano a su familia y encargarse de regar, por temporadas, las plantas de todos ellos. Les ha propuesto que, de vez en cuando en períodos de 2 semanas, será ella la encargada de regar en todas las casas. Le gusta mucho la informática también, así que quiere preparar un programa que le indique cuándo estará libre o tendrá que regar alguna planta. Así puede encargarse de cumplir con su promesa y también saber rápidamente cuándo tiene un día libre para despreocuparse de todo y dedicarse a sus cosas. Ella solamente pide que al comienzo de su período todo el mundo deje las plantas regadas para que ella pueda planificarse. Y también hay que tener en cuenta que Diana irá a regar cuando sea estrictamente necesario y algunas de las plantas de cualquier casa lo necesite. Y, para ahorrar, solamente regará las plantas que en ese momento no puedan pasar más días sin agua.

Entrada

Como entrada se reciben varios casos de prueba. Cada caso de prueba, que es la casa de un familiar de Diana, está compuesto de 2 líneas. En la primera se muestra el número de plantas que hay en esa casa. La segunda línea indica, para cada una de esas plantas, cuántos días puede resistir sin que se riegue. La entrada termina con una línea que contiene un 0 que no se tiene que procesar.

Salida

Como salida se obtendrá, para cada uno de los casos, una línea que reflejará qué tiene que hacer Diana cada día durante las dos próximas semanas. Una L significará que tiene el día libre y una R que tiene que regar al menos una de las plantas de esa casa.

Entrada de ejemplo

```
5
3 5 6 5 4
3
4 4 4
3
2 0 2
1
2
9
1 1 1 1 1 1 1 1 1
0
```

Salida de ejemplo

```
L L L R R R R R L R L R L R
L L L L R L L L L R L L L L
R R R R R R R R R R R R R R
```

```
L L R L L R L L R L L R L L
L R L R L R L R L R L R L R
```

L. CARRERAS DE RESISTENCIA

En los últimos años se han puesto de moda las carreras de resistencia. Cada vez son más exigentes y es más difícil llegar en forma. Si no entrenas lo suficiente no llegarás con la forma adecuada. Pero también puede pasar que te pases de entrenar y llegues totalmente agotado por haberte pasado de forma.

Así, lo más importante es planificar bien los entrenamientos para saber cómo y cuándo empezar. Y es precisamente en esa fase en la que se encuentra Antonio, puesto que ha decidido que va a correr la siguiente edición de la UTMB Mont-Blanc, una carrera de 174 km de distancia con unos cuantos miles de metros de desnivel acumulado.

Como Antonio no va con afán competitivo, y solamente quiere disfrutar, se ha puesto como objetivo acabarla a un ritmo y en un tiempo ambos ya determinados por él mismo. En su caso ha calculado que le llevará 18 horas acabarla. También se lo ha comentado a esos amigos con los que queda, pero no saben si todos podrán participar porque cada uno está en un estado de forma diferente y es posible que no les dé tiempo a llegar en la forma adecuada. Se han informado y el plan de entrenamiento correcto propone doblar el tiempo de carrera cada semana hasta conseguir correr durante 18 horas al menos una vez antes de la competición, aunque sea en la misma semana que la carrera. Solo así podrán participar en la carrera sin riesgo de lesiones. Ese mismo plan dice que no se debe llegar a esa distancia con 4 semanas o más de antelación porque se corre el riesgo de llegar agotado a la carrera debido al sobre entrenamiento.

Todos saben ya cuánto tiempo están corriendo a la semana a día de hoy así que, para saber si podrán o no participar en la carrera con el entrenamiento adecuado, Antonio se ha prestado a hacer un programa que pueda adelantar el resultado de su planificación y así saber quienes llegarán en forma a la carrera.

Entrada

Como entrada se reciben varios casos de prueba, cada uno compuesto de 2 líneas. La primer línea de cada caso contiene números que indican el número de amigos que están interesados en apuntarse a la carrera junto con el número de semanas que quedan para la carrera. En la segunda línea del caso vendrá, por cada amigo interesado, un número que representa los minutos que cada uno está corriendo semanalmente a día de hoy.

Salida

La salida mostrará, para cada caso de prueba, una línea con el resultado de la planificación de cada uno de los amigos: `NO` en el caso de que no vayan a conseguir llegar en forma a la carrera, `OK` para los que lo vayan a conseguir y `KO` para aquellos que, consiguiéndolo, lo hagan con 4 semanas o más de antelación con respecto a la semana de la competición.

Entrada de ejemplo

```
5 6
1 15 20 12 600
2 10
30 40
11 8
1 10 20 30 40 50 60 70 80 90 100
0
```

Salida de ejemplo

```
NO NO OK NO KO
KO KO
NO OK OK OK OK OK OK KO KO KO KO
```