A. Caminando por los Cerros Orientales

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Los investigadores del Tribunal de Cimas y Suelos (TCS) realizan cada año un recorrido lineal por los senderos que bordean los Cerros Orientales de Bogotá. Durante la expedición registran la altitud sobre el nivel del mar en cada uno de los n puntos del camino, siguiendo siempre la misma dirección.

Según la tradición del tribunal, basada en estudios antiguos y en la sabiduría de los guías locales, un tramo del recorrido se considera parte de un cerro cuando su altitud se mantiene por encima de un umbral crítico k, asociado a la altura mínima en la que comienza la vegetación de alta montaña. Cuando el terreno desciende estrictamente por debajo de ese nivel, se considera que se ha dejado atrás un cerro antes de llegar a otro.

Dadas las n mediciones de altitud, tu tarea es determinar cuántos cerros distintos aparecen a lo largo del recorrido.

Entrada

La primera línea consiste en dos enteros n y k ($1 \leq n \leq 1000, 1 \leq k \leq 10^9$) la cantidad de puntos del camino y el umbral crítico.

La segunda línea contiene n enteros $h_1, h_2, ..., h_n$ ($1 \leq h_i \leq 10^9$), donde h_i es la altitud en el punto i.

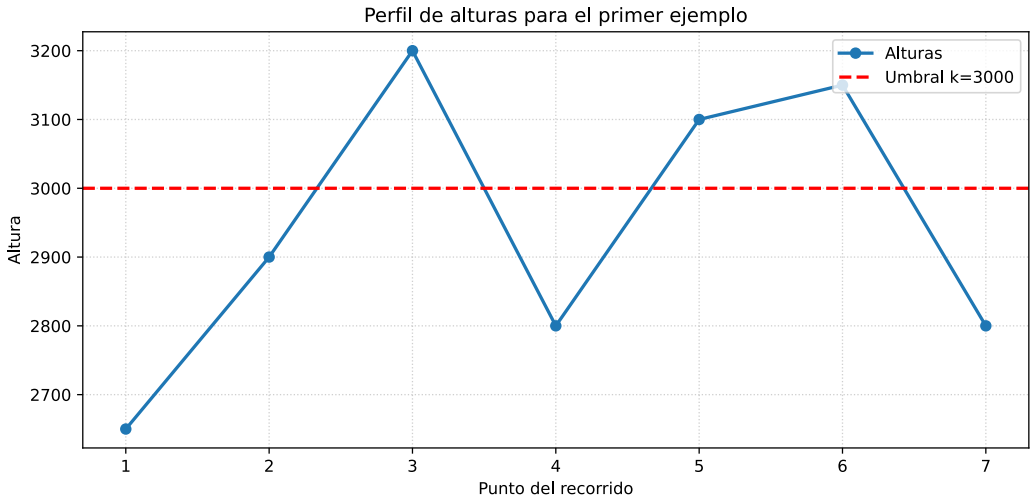
Salida

Imprime un único entero: el número de cerros identificados en el recorrido.

Ejemplos

Entrada 1	Salida 1
7 3000 2650 2900 3200 2800 3100 3150 2800	2

Entrada 2	Salida 2
5 3 6 5 2 3 3	2



B. Buscando el tesoro en Playa Blanca

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Este problema es interactivo.

Nacho y Paz están en la famosa Playa Blanca, en Colombia, aprovechando de descansar después de preparar la final de Codificadas. Mientras recuperan energías, Ximena, de TCS, les cuenta que en esa playa existe un tesoro escondido. Para ayudar a encontrarlo, Ximena ha desarrollado un instrumento especial.

Este instrumento funciona de la siguiente manera: desde cualquier posición, permite consultar si uno se acerca al tesoro al moverse un paso hacia la izquierda, derecha, arriba o abajo.

Para facilitar la búsqueda, Nacho y Paz representan una grilla de tamaño $n \times n$ sobre la playa. Emocionados, se ubican exactamente en el centro de la playa y comienzan la búsqueda. Sin embargo, el cansancio por la organización de Codificadas los supera, así que te piden ayuda para encontrar el tesoro utilizando el instrumento.

Pero tienes que tener cuidado, el instrumento se sobrecalentará y explotará si lo usas demasiadas veces.

Entrada

La primera línea consiste de un entero n ($1 \leq n \leq 1000$), correspondiente al tamaño de la grilla.

n siempre será impar. Por lo tanto la posición inicial de Nacho y Paz será $(\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor)$. Los índices de la grilla van de 0 a $n-1$. La posición $(0, 0)$ de la grilla se encuentra en la esquina inferior-izquierda.

Interacción

Después de leer el entero n, puedes realizar las siguientes acciones:

- `M d`: mueve a Nacho y Paz una posición en la dirección `d`. `d` puede ser `U` (arriba), `D` (abajo), `L` (izquierda) o `R` (derecha). Si intentas moverte a una posición inexistente* de la grilla, tu programa se considerará incorrecto.
- `? d`: consulta al instrumento si, al moverte una posición en la dirección `d`, distancia al tesoro aumenta o disminuye. El instrumento responderá `+` o `-`. `+` significa que la distancia aumentó y `-` significa que disminuyó (te estás acercando). Si preguntas al instrumento por una posición inexistente* de la grilla, tu programa se considerará incorrecto.
- `!`: indica que crees haber llegado a la posición donde está el tesoro y el programa debe terminar. Si la posición de Nacho y Paz no corresponde al tesoro, el programa se considerará incorrecto.

Se puede consultar al instrumento a lo más $4 \cdot n$ veces.

* Una posición inexistente corresponde a cualquier (i, j) tal que $i < 0$, $i \geq n$, $j < 0$ o $j \geq n$.

Recordatorio

Después de imprimir cada línea, no olvides escribir el fin de línea y vaciar (flush) el búfer de salida. Algunos lenguajes no imprimen inmediatamente, por lo que es necesario realizar una operación de flush manualmente:

- *Python: Agregar el parámetro* `flush` *al imprimir:* `print(..., flush=True)`
- *C++: Utilizar* `std::cout << std::flush;` *tras imprimir.*
- *C: Utilizar* `fflush(stdout);` *tras imprimir.*
- *Otros lenguajes: Referir a la documentación.*

Ejemplos

Entrada 1	Salida 1
7 - - + - - 	 ? D M D ? D M D ? D ? L M L ? L M L !

C. Policía internacional

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Eres un agente de inmigración que recibe personas que quieren entrar a tu país. Tienes la importante tarea de interactuar con un visitante, revisar su pasaporte y su visa, y verificar si los nombres coinciden.

Esta tarea tiene dos dificultades principales. La primera es que los nombres pueden aparecer en un orden distinto en el pasaporte y en la visa, ya sea por razones culturales o porque algunos elementos pueden ser títulos más que nombres. La segunda es que los documentos no están necesariamente diseñados para ser leídos por personas, por lo que en cada documento el nombre aparece sin espacios.

Por ejemplo, una persona llamada `Mateo Matis Mayor` puede tener en el pasaporte:

`MATEOMATISMAYOR`

y en la visa:

`MATISMAYORMATEO`

pero también:

`MAYORMATEOMATIS`

`MAYORMATISMATEO`

entre otras combinaciones. Siempre que el par de documentos sea válido, el nombre del pasaporte se podrá descomponer en exactamente tres partes, y existirá una forma de reordenar esas tres partes para obtener el nombre de la visa (la descomposición o el orden pueden no ser únicos).

Sin embargo, a veces aparecerá una persona intentando entrar al país con documentos inválidos. Por ejemplo, puede tener en el pasaporte:

`JOSEPEDROPASCAL`

y en la visa:

`PEDROJOSECALPAS`

Este par no es válido porque no existe una forma de separar el nombre del pasaporte en tres partes, reordenarlas y obtener el nombre de la visa.

Tu labor será recibir este par de cadenas, decidir si son válidas y, si lo son, entregar una posible descomposición.

Entrada

La primera línea contiene un número n, el largo de ambos nombres que vas a recibir ($3 \leq n \leq 50$).

La segunda línea contiene un string con letras mayúsculas del alfabeto inglés, de largo n, que corresponde al nombre tal como aparece en el pasaporte.

La tercera línea contiene un string con letras mayúsculas del alfabeto inglés, de largo n, que corresponde al nombre tal como aparece en la visa.

Salida

Si el par de nombres es válido, entrega una línea que diga `YES` y luego dos líneas más:

- La primera debe contener una descomposición del nombre del pasaporte en tres partes, separadas por espacios.
- La segunda debe contener una descomposición del nombre de la visa en tres partes. Estas tres cadenas deben ser exactamente las mismas de la línea anterior, pero posiblemente en otro orden.

Si el par de nombres no es válido, entrega solo una línea que diga `NO`.

Ejemplos

Entrada 1	Salida 1
15 MATEOMATISMAYOR MATISMAYORMATEO	YES MATEO MATIS MAYOR MATIS MAYOR MATEO

Entrada 2	Salida 2
15 JOSEPEDROPASCAL PEDROJOSECALPAS	NO

Entrada 3	Salida 3
22 PEDROPEDROPEDROPEDROPE DROPEDROPEDROPEDROPEPE	YES PE DROPEDROPE DROPEDROPE DROPEDROPE DROPEDROPE PE

D. Constelación colmillos de león

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

En lo alto del cielo nocturno se encuentra la constelación Diente de León, la cual está compuesta por n estrellas conectadas por $n-1$ hilos de luz, formando un árbol.

A los astrónomos les interesa otro tipo de constelación llamada Constelación Colmillos de León. Esta consiste en dos estrellas centrales conectadas por un hilo de luz, y todas las demás estrellas están conectadas directamente a una de estas dos estrellas centrales por un hilo de luz.

Los astrónomos están tan fascinados con la Constelación Colmillos de León que buscan modificar la Constelación Diente de León para transformarla en este tipo de constelación. Para lograr esto, pueden realizar la siguiente operación que consta de dos pasos en orden:

1. **Elegir** un hilo de luz existente que conecta dos estrellas, digamos u y v
2. **Desconectar** el hilo de uno de sus extremos y **conectarlo** a cualquier otra estrella z tal que los hilos de luz sigan formando un árbol.

Para obtener la aclamada Constelación Colmillos de León lo más rápido posible, los astrónomos te piden ayuda para calcular la **mínima cantidad de operaciones** que deben realizar para lograr la transformación.

Entrada

La primera línea de la entrada contiene un entero n ($2 \leq n \leq 10^5$), la cantidad de estrellas en la Constelación Diente de León.

Luego siguen $n-1$ líneas, cada una con dos enteros a y b ($1 \leq a, b \leq n$), describiendo un hilo de luz que conecta las estrellas a y b.

Salida

La salida debe contener un único entero: la mínima cantidad de operaciones que se deben utilizar para conseguir la Constelación Colmillos de León.

Ejemplos

Entrada 1	Salida 1
9 1 2 2 3 3 4 4 5 4 6 4 7 1 8 1 9	2

Entrada 2	Salida 2
4 1 2 2 3 3 4	0

Entrada 3	Salida 3
5 1 2 2 3 3 4 4 5	1

Nota

En el primer caso de prueba, la siguiente es una posible secuencia de 2 operaciones para formar una constelación colmillos de león:

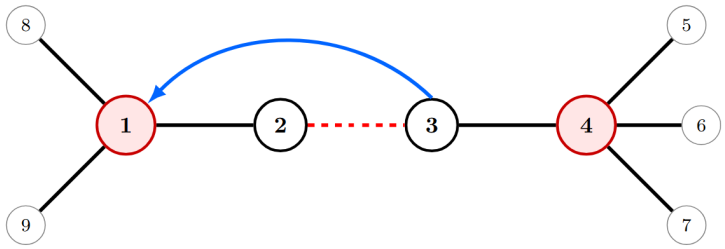


Figure 2: En la primera operación se elige el hilo de luz que conecta las estrellas (2, 3) y se desconecta de 2 para crear el hilo de luzque conecta (1, 3).

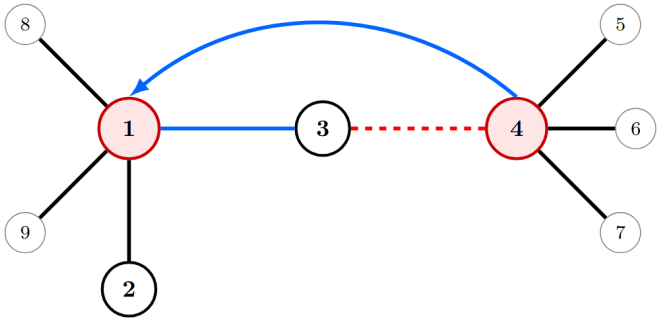


Figure 3: En la segunda operación se elige el hilo de luz que conecta las estrellas (3, 4) y se desconecta de 3 para crear el hilo de luz que conecta (1, 4).

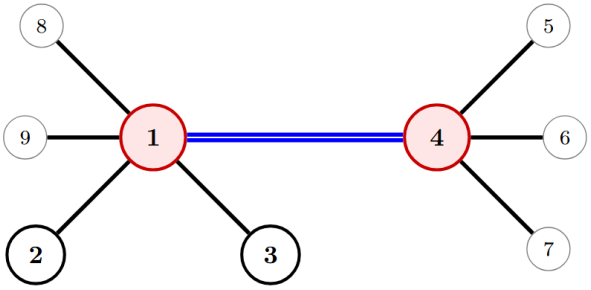


Figure 4: Luego de dos operaciones se consigue una constelación colmillos de león cuyas estrellas centrales son 1 y 4.

Se puede demostrar que no se puede conseguir una constelación colmillos de león en menos de 2 operaciones.

E. Robot perdido

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Este problema es interactivo.

Hay un robot en una grilla de $n \times m$ que consiste en celdas que están vacías (`.`) u ocupadas (`#`). Este pobre robot está perdido, por lo que no conoce su posición inicial. Por suerte tienes un mapa de la grilla y puedes darle instrucciones al robot, con las cuales debes intentar encontrar su posición.

En cada consulta puedes escoger una dirección (arriba, derecha, abajo, izquierda), y el robot intentará desplazarse una celda en la dirección escogida. El movimiento será válido y ocurrirá solo si el robot se intenta mover a una celda vacía.

Luego debes leer la respuesta, que consistirá en 0 si el movimiento es inválido y 1 si es válido. Si el movimiento es válido, **la posición del robot cambiará** según la dirección escogida.

Tu tarea es descubrir la posición del robot usando no más de $2(n \cdot m + n + m)$ consultas. Se garantiza que es posible moverse entre cualquier par de celdas vacías usando alguna secuencia de movimientos.

Entrada

La primera línea consiste en dos enteros n y m ($1 \le n \cdot m \le 1000$) — la cantidad de filas y columnas de la grilla.

La i-ésima de las siguientes n líneas describen a la i-ésima fila de la grilla, conteniendo m caracteres. El j-ésimo de estos indica el caracter de la posición (i, j) de la grilla, el cual puede ser '`.`' (celda vacía) o '`#`' (celda ocupada).

Al terminar de leer la grilla, la interacción comienza con tu primera consulta.

Interacción

Para realizar una consulta, debes imprimir una línea en alguno de estos formatos:

- `"? U"` — intentar moverse hacia arriba;
- `"? R"` — intentar moverse hacia la derecha;
- `"? D"` — intentar moverse hacia abajo;
- `"? L"` — intentar moverse hacia la izquierda.

Después de cada intento de movimiento, el juez imprimirá un entero que será `0` si el movimiento es inválido (ya sea porque la celda estaba ocupada o porque estaba fuera de la grilla) y un `1` si el movimiento fue válido (y la posición del robot cambió).

Una vez encuentres la posición actual del robot, debes reportarla imprimiendo una línea en este formato:

- `"! x y"` — indicando que la posición **actual** del robot es `(x, y)` (`1 ≤ x ≤ n, 1 ≤ y ≤ m`).

Después de esto el programa debe terminar.

Si en cualquier paso de la interacción lees `-1` en vez de datos válidos, el programa debe terminar inmediatamente, recibiendo `Wrong answer` por haber hecho una consulta de formato inválido. No terminar el programa en este caso puede resultar en un veredicto arbitrario ya que el programa seguirá leyendo de un **stream** cerrado.

La interacción en este problema es **adaptativa**, lo que significa que el juez puede cambiar las coordenadas iniciales del robot en cualquier momento, siempre y cuando la nueva posición sea consistente con todas las consultas y respuestas realizadas en la interacción.

Recordatorio

Después de imprimir cada línea, no olvides escribir el fin de línea y vaciar (flush) el búfer de salida. Algunos lenguajes no imprimen inmediatamente, por lo que es necesario realizar una operación de flush manualmente:

- *Python: Agregar el parámetro* `flush` *al imprimir:* `print(..., flush=True)`
- *C++: Utilizar* `std::cout << std::flush;` *tras imprimir.*
- *C: Utilizar* `fflush(stdout);` *tras imprimir.*
- *Otros lenguajes: Referir a la documentación.*

Ejemplos

Entrada 1	Salida 1
4 4 .#.# .#.# .#.# 1 1 0 1 0	 ? D ? D ? D ? R ? R ! 4 4

Nota

En este ejemplo, la posición inicial del robot es $(2, 3)$, conocida solo por el juez.

Para la primera consulta ("`? D`") el juez retorna 1 y el robot se mueve una posición hacia abajo. Su nueva posición es $(3, 3)$.

Para la segunda consulta ("`? D`") el juez retorna 1 y el robot se mueve una posición hacia abajo. Su nueva posición es $(4, 3)$.

Para la tercera consulta ("`? D`") el juez retorna 0 ya que el robot no puede moverse hacia abajo porque se saldría de la grilla, y el robot se mantiene en la posición $(4, 3)$.

Para la cuarta consulta ("`? R`") el juez retorna 1 y el robot se mueve una posición hacia la derecha. Su nueva posición es $(4, 4)$.

Para la quinta consulta (“`? R`”) el juez retorna 0 ya que el robot no puede moverse más a la derecha. El robot se mantiene en $(4, 4)$.

Finalmente se determina que el robot está en la posición $(4, 4)$ y se imprime esa posición.

Nota que este es solo un ejemplo de funcionamiento de la interacción, y pueden haber otras soluciones válidas

F. Sanar o no sanar

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

En un mundo donde existen M enfermedades numeradas del 1 al M, y N remedios mágicos numerados del 1 al N, cada remedio tiene dos efectos:

- Cura ciertos tipos de enfermedades.
- Causa otros tipos de enfermedades.

Además, comienzas ya enfermo con un conjunto inicial de K enfermedades.

Si tomas un conjunto de remedios S, su efecto total es:

- Contraer todas las enfermedades causadas por cualquier remedio en S.
- Curar todas las enfermedades curadas por cualquier remedio en S.

Finalmente quedarás con:

(enfermedades iniciales $\cup$ enfermedades causadas) $\setminus$ enfermedades curadas.

Tu objetivo es encontrar si existe algún subconjunto de remedios tal que quedes completamente sano (sin ninguna enfermedad). De ser posible, debes reportar cuántos remedios necesitas y cuáles son (sus índices).

Entrada

La primera línea contiene dos enteros N y M ($1 \leq N, M \leq 2 \cdot 10^5$), el número de remedios y el número de enfermedades, respectivamente.

La segunda línea contiene un entero K ($0 \leq K \leq M$), seguido de K enteros $d_1, d_2, ..., d_K$ ($1 \leq d_i \leq M$), que representan las enfermedades iniciales que padeces. Si $K = 0$, la línea contendrá únicamente el cero.

Cada una de las siguientes N líneas describe un remedio. Cada línea contiene primero un entero a_i ($0 \leq a_i \leq M$), seguido de a_i enteros distintos $c_{i,1}, c_{i,2}, ..., c_{i,a_i}$ ($1 \leq c_{i,j} \leq M$), que representan las enfermedades que cura el remedio i, y luego un entero b_i ($0 \leq b_i \leq M$), seguido de b_i enteros distintos $p_{i,1}, p_{i,2}, ..., p_{i,b_i}$ ($1 \leq p_{i,j} \leq M$), que representan las enfermedades que provoca el remedio i.

Se garantiza que todos los enteros cumplen los rangos indicados, y que dentro de cada lista de enfermedades curadas o provocadas no hay repeticiones. Finalmente, se garantiza que la suma de a_i sobre todos los remedios i y la suma de b_i sobre todos los remedios i no supera 10^6.

Salida

Si es posible sanar las enfermedades iniciales usando algún subconjunto de remedios:

- En la primera línea imprima la palabra “YES”.
- En la segunda línea imprima un entero L, el largo del subconjunto de remedios utilizado.
- En la tercera línea imprima L enteros separados por espacios, los índices distintos de los remedios que conforman la solución (numerados del 1 al N). En caso de que existan multiples soluciones, imprima cualquiera de ellas.

Si no es posible quedar completamente sano imprima únicamente la palabra “NO”.

Ejemplos

Entrada 1	Salida 1
3 5 1 5 1 2 0 1 5 1 3 1 3 1 2	YES 3 1 2 3

Entrada 2	Salida 2
4 5 2 3 1 3 5 1 2 0 4 5 2 1 3 5 4 2 1 5 3 1 5 5 1 5 3 2 4 0 0	NO

G. Fernanda vs Leticia

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Fernanda y Leticia son dos buenas amigas. Todos los días compiten para ver quién puede ganarle a la otra. Cada día eligen un juego distinto; para hoy deciden jugar uno llamado "*Achicando-ando*".

El juego consiste en un tablero de largo n, donde cada celda contiene la letra `F` o `L`, y se empieza con un rango inicial $l = 1$ y $r = n$.

Se juega por turnos y siempre comienza Fernanda. En cada turno, la jugadora debe elegir un subrango* de largo $\lfloor \frac{r-l+1}{2} \rfloor$ en el que al menos la mitad de las letras coincidan con la letra que le corresponde: `F` para Fernanda y `L` para Leticia. Al elegir un subrango definido por i y j ($i \leq j$), ese subrango pasa a ser el nuevo rango para el siguiente turno, es decir, ahora $l = i$ y $r = j$. La primera jugadora que no pueda elegir un subrango válido pierde.

Paz, que está observando a Fernanda y Leticia jugar, se interesa por el juego y quiere poder determinar si existe una forma en la que Fernanda le gane a Leticia, asumiendo que ellas juegan de manera óptima. Paz está muy ocupada organizando Codificadas, por lo que te pide ayuda a ti para determinar la ganadora.

** Un subrango dentro del intervalo $[l, r]$ se define eligiendo dos índices i y j tales que $l \leq i \leq j \leq r$. Este subrango corresponde a los valores $a_i, a_{i+1}, ..., a_j$.*

Entrada

La primera línea de la entrada consiste en un entero n ($2 \leq n \leq 10^5$), correspondiente al largo del tablero.

La segunda línea consiste en una secuencia de letras sin espacios, correspondiente al tablero del juego. Las letras solo pueden ser `F` o `L`.

Salida

Si Fernanda será la ganadora, la salida debe consistir únicamente de la palabra `Fernanda`, en caso contrario debe consistir de la palabra `Leticia`.

Puntaje

Este problema se evalúa con subtareas. Para obtener el puntaje total, tu solución debe funcionar correctamente en todos los casos de prueba de cada subtarea.

- **Subtarea 1 (60 puntos):** $n \leq 1000$. Tu solución obtendrá estos puntos si pasa **todos** los casos donde n es a lo más 1000.
- **Subtarea 2 (40 puntos):** Sin restricciones adicionales. Esta subtarea incluye casos con tamaños de hasta $n \leq 10^5$. Tu solución obtendrá estos puntos al resolver correctamente estos casos.

Ejemplos

Entrada 1	Salida 1
11 FLFFFFFLLLF	Fernanda

Entrada 2	Salida 2
32 FFFLLLFLLLLLFLFLFLLLFLLLFLFLFL	Leticia

Nota

Para el primer caso, este es un ejemplo de los posibles turnos:

- El rango inicial es $l = 1$ y $r = 11$. Fernanda deberá elegir un subrango de largo 5. Fernanda elige el subrango $l = 2$ y $r = 6$. Este subrango contiene 4 `F`, cumpliendo la condición que hayan más de la mitad de `F`.
- El rango actual es $l = 2$ y $r = 6$. Leticia deberá elegir un subrango de largo 2. Leticia solo puede elegir el subrango $l = 2$ y $r = 3$. Esto debido a que es el único subrango que contiene al menos 1 letra `L`.
- El rango actual es $l = 2$ y $r = 3$. Fernanda elige el subrango $l = 3$ y $r = 3$. Dado que es el único subrango posible.
- Leticia no puede elegir un subrango de largo 0, por lo tanto Fernanda gana.

H. ¿Múltiplos de 3?

Límite de tiempo: 1 segundos

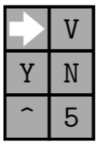
Límite de memoria: 256 megabytes

Este problema tiene un visualizador asociado. Haz click aquí para verlo.

Carla está jugando un juego de mesa. Este juego se juega usando un *tablero*, que es una grilla rectangular similar al tablero de ajedrez, pero en lugar de 8 filas y 8 columnas, este tablero tiene n filas y m columnas.

Cada celda del tablero tiene impreso un símbolo, que puede ser cualquiera de los siguientes: `<`, `>`, `V`, `^`, `N`, `Y`, `.`, o un dígito del `0` al `9`.

Antes de comenzar a jugar, Carla elige un entero positivo x y coloca una figurina sobre la celda de la primera fila y primera columna, apuntando hacia la derecha.



$x = 148$

Figure 5: Configuración inicial de un tablero de ejemplo y un número x elegido por Carla. Notar la figurina sobre la celda superior izquierda.

Luego, Carla sigue el manual de instrucciones, que detalla los pasos a seguir:

1. Mover la figurina una celda hacia la dirección que apunta. Si la figurina escapa del tablero, Carla **pierde** y termina el juego.
2. Según el símbolo escrito sobre la celda a la que se movió la figurina, realizar una acción:

   `<`

   Rotar la figurina tal que apunte hacia la izquierda.

   `>`

   Rotar la figurina tal que apunte hacia la derecha.

   `V`

   Rotar la figurina tal que apunte hacia abajo.

   `^`

   Rotar la figurina tal que apunte hacia arriba.

   `N`

   Borrar el último dígito de x. Si tras borrar no quedan dígitos, Carla **pierde** y termina el juego.

   `Y`

   Borrar el último dígito de x. Si tras borrar no quedan dígitos, Carla **gana** y termina el juego.

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

   Consideremos t como el dígito escrito en la celda ($0 \leq t \leq 9$), y d como el último dígito de x ($0 \leq d \leq 9$):

3. $t < d$: Girar la figurina 90° en sentido del reloj.

4. $t > d$: Girar la figurina 90° en contra del sentido del reloj.
5. $t = d$: No realizar ninguna acción.

   No realizar ninguna acción.
6. Volver al paso 1 y repetir. Si el juego nunca termina entonces se considera que Carla **pierde**.

Para ejemplificar, veamos cómo jugaría Carla con el tablero anterior y con $x = 34$:

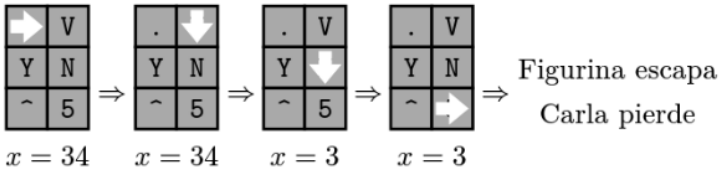


Figure 6: Configuración inicial de un tablero de ejemplo y un número x elegido por Carla. Notar la figurina sobre la celda superior izquierda.

Sin embargo, si Carla hubiera elegido $x = 87$, hubiera ganado:

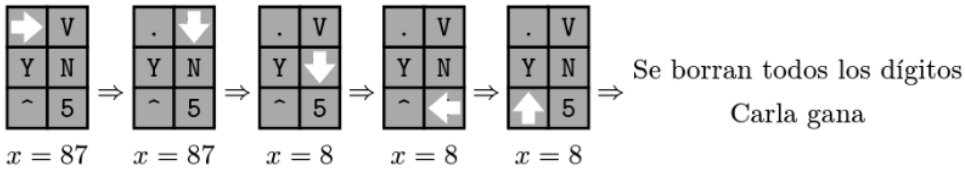


Figure 7: Configuración inicial de un tablero de ejemplo y un número x elegido por Carla. Notar la figurina sobre la celda superior izquierda.

Es decir, para cada tablero hay algunas elecciones de x que son ganadoras y otras que son perdedoras.

Hace algunos meses, Carla quiso construir un tablero *interesante*. Un tablero interesante es un tablero tal que las elecciones ganadoras de x son exactamente los múltiplos de 3. Ella cree tener la solución, pero no está segura. ¿Puedes ayudarla escribiendo un programa que determina si un tablero dado es interesante o no?

Entrada

La primera línea contiene dos enteros n y m, la cantidad de filas y columnas respectivamente ($1 \leq n, m \leq 100$).

Luego, habrán n líneas, cada una con m caracteres del alfabeto "`<>V^NY.0123456789}`". El j-ésimo caracter de la i-ésima de estas líneas representa el símbolo en la celda de la fila i (de arriba a abajo) y columna j (de izquierda a derecha) del tablero.

Salida

Si el tablero es interesante, imprime "`INTERESANTE`". En caso contrario, imprime "`FOME`".

Ejemplos

Entrada 1	Salida 1
3 2 .V YN ^5	FOME

Entrada 2	Salida 2
6 36 >VVVVVVVVVVV.VVVVVVVVVVV.VVVVVVVVVVV .>876543210V.>876543210V.>876543210V .Y..........N..........N.......... .^<..<..<..<...<..<..<......<..<..<. ....>..>..>..^<..<..<..<...<..<..<..<.. ...>..>..>......>..>..>..^<..<..<..<	INTERESANTE

Nota

El primer tablero de ejemplo no es interesante, dado que existen elecciones de x tal que Carla pierde incluso si x es múltiplo de 3, por ejemplo $x = 9$. Además, también existen elecciones de x no múltiplo de 3 tales que Carla gana, por ejemplo $x = 61$.

El segundo tablero de ejemplo es interesante, dado que Carla gana para todo x positivo y múltiplo de 3, y pierde para todo x positivo que no es múltiplo de 3.