A. Caminhando pelos Cerros Orientales

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

Os pesquisadores do Tribunal de Cimas e Solos (TCS) realizam todos os anos um percurso linear pelos caminhos que contornam os Cerros Orientales de Bogotá. Durante a expedição, registram a altitude em relação ao nível do mar em cada um dos n pontos do trajeto, sempre seguindo a mesma direção.

Segundo a tradição do tribunal, baseada em estudos antigos e na sabedoria dos guias locais, um trecho do percurso é considerado parte de um cerro quando a sua altitude se mantém maior ou igual a um limiar crítico k, associado à altura mínima em que começa a vegetação de alta montanha. Quando o terreno desce estritamente abaixo desse nível, considera-se que aquele cerro ficou para trás antes de chegar a outro.

Dadas as n medições de altitude, sua tarefa é determinar quantos cerros distintos aparecem ao longo do percurso.

Entrada

A primeira linha consiste em dois inteiros n e k ($1 \leq n \leq 1000, 1 \leq k \leq 10^9$) a quantidade de pontos do caminho e o limiar crítico.

A segunda linha contém n inteiros $h_1, h_2, ..., h_n$ ($1 \leq h_i \leq 10^9$), onde h_i é a altitude no ponto i.

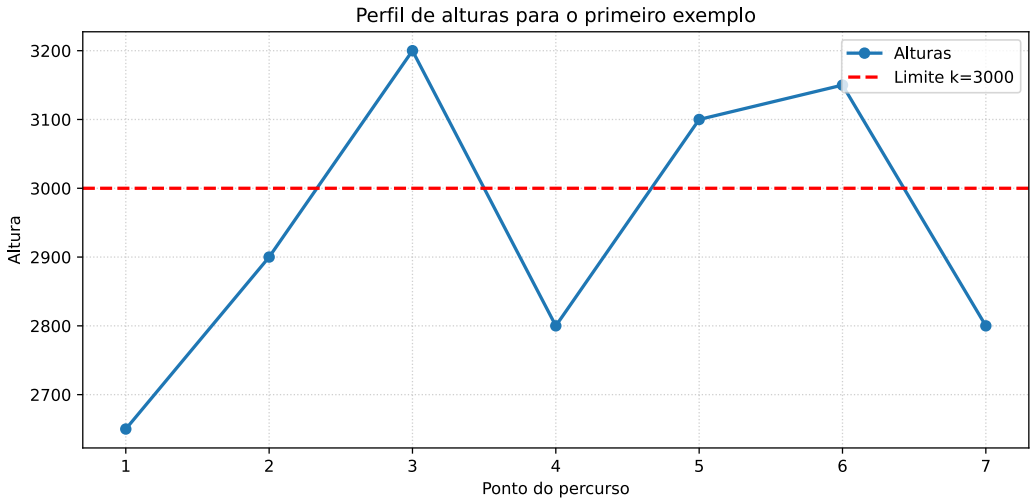
Saída

Imprima um único inteiro: o número de cerros identificados ao longo do percurso.

Exemplos

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

Entrada 2	Saída 2
5 3 6 5 2 3 3	2



B. Buscando o tesouro em Playa Blanca

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

Este problema é interativo.

Nacho e Paz estão na famosa Playa Blanca, na Colômbia, aproveitando para descansar depois de organizarem a final de Codificadas. Enquanto recuperam as energias, Ximena, da TCS, conta a eles que existe um tesouro escondido naquela praia. Para ajudar a encontrá-lo, Ximena desenvolveu um instrumento especial.

Esse instrumento funciona da seguinte maneira: a partir de qualquer posição, ele permite consultar se você se aproximaria ou se afastaria do tesouro ao mover-se um passo para a esquerda, direita, para cima ou para baixo.

Para facilitar a busca, Nacho e Paz representam uma grade de tamanho $n \times n$ sobre a praia. Animados, eles se posicionam exatamente no centro da grade e começam a procurar. No entanto, o cansaço da organização de Codificadas os vence, então eles pedem sua ajuda para encontrar o tesouro usando o instrumento.

Mas tome cuidado: o instrumento pode superaquecer e explodirá se você usá-lo vezes demais.

Entrada

A primeira linha contém um inteiro n ($1 \leq n \leq 1000$), correspondente ao tamanho da grade.

n será sempre ímpar. Portanto, a posição inicial de Nacho e Paz será $(\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor)$. Os índices da grade vão de 0 a $n-1$. A posição $(0, 0)$ encontra-se no canto inferior esquerdo.

Interação

Depois de ler o inteiro n, você poderá realizar as seguintes ações:

- `M d`: move Nacho e Paz uma posição na direção `d`. `d` pode ser `U` (cima), `D` (baixo), `L` (esquerda) ou `R` (direita). Se você tentar mover-se para uma posição inexistente* da grade, seu programa será considerado incorreto.
- `? d`: consulta ao instrumento se, ao mover-se uma posição na direção `d`, a distância até o tesouro aumenta ou diminui. O instrumento responderá `+` ou `-`. `+` significa que a distância aumentou, e `-` significa que diminuiu (você está se aproximando). Se você consultar o instrumento sobre uma posição inexistente* da grade, seu programa será considerado incorreto.
- `!`: indica que você acredita ter chegado à posição onde está o tesouro, e o programa deve terminar. Se a posição atual de Nacho e Paz não corresponder ao tesouro, o programa será considerado incorreto.

Você pode consultar o instrumento no máximo $4 \cdot n$ vezes.

* Uma posição inexistente corresponde a qualquer (i, j) tal que $i < 0$, $i \geq n$, $j < 0$ ou $j \geq n$.

Recordatório

Depois de imprimir cada linha, não se esqueça de escrever o fim de linha e esvaziar (flush) o buffer de saída. Algumas linguagens não imprimem imediatamente, portanto é necessário realizar uma operação de flush manualmente:

- *Python: Adicionar o parâmetro* `flush` *ao imprimir:* `print(..., flush=True)`
- *C++: Usar* `std::cout << std::flush;` *após imprimir.*
- *C: Usar* `fflush(stdout);` *após imprimir.*
- *Outras linguagens: Consultar a documentação.*

Exemplos

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

C. Polícia internacional

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

Você é um agente de imigração que recebe pessoas que desejam entrar no seu país. Sua tarefa é interagir com o visitante, revisar seu passaporte e sua permissão de entrada, e verificar se os nomes coincidem.

Esse trabalho apresenta duas dificuldades principais. A primeira é que os nomes podem aparecer em uma ordem diferente no passaporte e na permissão, seja por motivos culturais ou porque alguns elementos são títulos e não nomes. A segunda é que os documentos não são necessariamente feitos para leitura humana, então em cada documento o nome aparece sem espaços.

Por exemplo, uma pessoa chamada `Mateo Matis Mayor` pode ter no passaporte:

`MATEOMATISMAYOR`

e na permissão:

`MATISMAYORMATEO`

mas também:

`MAYORMATEOMATIS`

`MAYORMATISMATEO`

entre outras combinações. Sempre que o par de documentos for válido, o nome do passaporte poderá ser dividido em exatamente três partes, e existirá uma forma de reorganizar essas três partes para obter o nome da permissão (a decomposição ou a ordem podem não ser únicas).

No entanto, às vezes alguém tentará entrar no país com documentos inválidos. Por exemplo, pode ter no passaporte:

`JOSEPEDROPASCAL`

e na permissão:

`PEDROJOSECALPAS`

Esse par não é válido, pois não existe uma forma de dividir o nome do passaporte em três partes, reordená-las e obter o nome da permissão.

Sua tarefa será receber esse par de cadeias, decidir se é válido e, caso seja, fornecer uma possível decomposição.

Entrada

A primeira linha contém um número n, o comprimento de ambos os nomes que você receberá ($3 \leq n \leq 50$).

A segunda linha contém uma string com letras maiúsculas do alfabeto inglês, de comprimento n, correspondente ao nome tal como aparece no passaporte.

A terceira linha contém uma string com letras maiúsculas do alfabeto inglês, de comprimento n, correspondente ao nome tal como aparece na permissão.

Saída

Se o par de nomes for válido, imprima uma linha com `YES` e depois mais duas linhas:

- A primeira deve conter uma decomposição do nome do passaporte em três partes, separadas por espaços.

- A segunda deve conter uma decomposição do nome da permissão em três partes. Essas três cadeias devem ser exatamente as mesmas da linha anterior, mas possivelmente em outra ordem.

Se o par de nomes não for válido, imprima apenas uma linha com `NO`.

Exemplos

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

Entrada 2	Saída 2
15 JOSEPEDROPASCAL PEDROJOSECALPAS	NO

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

D. Constelação Presas de Leão

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

No alto do céu noturno encontra-se a constelação Dente de Leão, composta por n estrelas conectadas por $n-1$ fios de luz, formando uma árvore.

Os astrônomos se interessam por outro tipo de constelação chamada Constelação Presas de Leão. Ela consiste em duas estrelas centrais conectadas por um fio de luz, e todas as demais estrelas estão conectadas diretamente a uma dessas duas estrelas centrais por um fio de luz.

Os astrônomos estão tão fascinados pela Constelação Presas de Leão que desejam modificar a Constelação Dente de Leão para transformá-la nesse tipo de constelação. Para isso, eles podem realizar a seguinte operação, composta de dois passos, nesta ordem:

1. **Escolher** um fio de luz existente que conecta duas estrelas, digamos u e v
2. **Desconectar** o fio de um de seus extremos e **conectá-lo** a qualquer outra estrela z, de modo que os fios de luz continuem formando uma árvore.

Para obter a tão desejada Constelação Presas de Leão o mais rápido possível, os astrônomos pedem sua ajuda para calcular a **quantidade mínima de operações** necessárias para realizar a transformação.

Entrada

A primeira linha da entrada contém um inteiro n ($2 \leq n \leq 10^5$), o número de estrelas da Constelação Dente de Leão.

Em seguida, seguem $n-1$ linhas, cada uma com dois inteiros a e b ($1 \leq a, b \leq n$), descrevendo um fio de luz que conecta as estrelas a e b.

Saída

A saída deve conter um único inteiro: o número mínimo de operações necessárias para obter a Constelação Presas de Leão.

Exemplos

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

Entrada 2	Saída 2
4 1 2 2 3 3 4	0

Entrada 3	Saída 3
5 1 2 2 3 3 4 4 5	1

Nota

No primeiro caso de teste, a seguinte é uma possível sequência de 2 operações para formar uma constelação Presas de Leão:

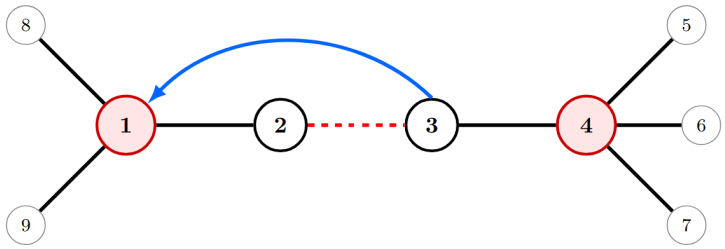


Figure 2: Na primeira operação, escolhe-se o fio de luz que conecta as estrelas (2, 3) e desconecta-se do 2 para criar o fio de luz que conecta (1, 3).

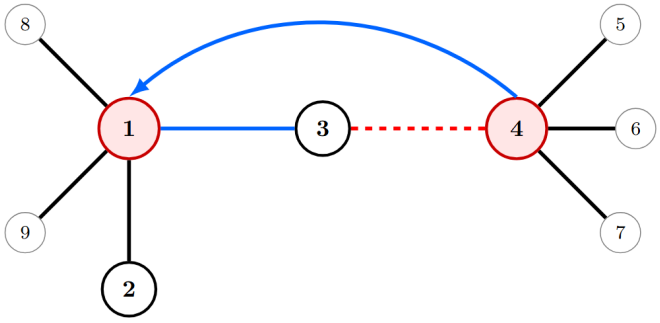


Figure 3: Na segunda operação, escolhe-se o fio de luz que conecta as estrelas (3, 4) e desconecta-se do 3 para criar o fio de luz que conecta (1, 4).

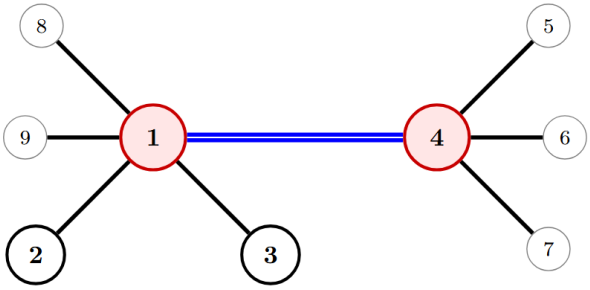


Figure 4: Após duas operações, obtém-se uma constelação Presas de Leão cujas estrelas centrais são 1 e 4. Pode-se demonstrar que não é possível obter uma constelação Presas de Leão em menos de 2 operações.

E. Robô perdido

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

Este problema é interativo.

Há um robô em uma grade de $n \times m$ composta por células que estão vazias (`.`) ou ocupadas (`#`). Esse pobre robô está perdido e não sabe sua posição inicial. Por sorte, você possui um mapa da grade e pode enviar instruções ao robô, com as quais deve tentar descobrir sua posição.

Em cada consulta, você pode escolher uma direção (cima, direita, baixo, esquerda), e o robô tentará se deslocar uma célula na direção escolhida. O movimento será válido e ocorrerá apenas se o robô tentar mover-se para uma célula vazia.

Depois de cada tentativa de movimento, você deve ler a resposta, que será `0` se o movimento for inválido e `1` se for válido. Se o movimento for válido, **a posição do robô mudará** conforme a direção escolhida.

Sua tarefa é descobrir a posição do robô usando no máximo $2(n \cdot m + n + m)$ consultas. Garante-se que é possível mover-se entre qualquer par de células vazias usando alguma sequência de movimentos.

Entrada

A primeira linha contém dois inteiros n e m ($1 \leq n \cdot m \leq 1000$) — o número de linhas e colunas da grade.

A i-ésima das n linhas seguintes descreve a i-ésima linha da grade, contendo m caracteres. O j-ésimo desses indica o caractere da posição (i, j) da grade, podendo ser '`.`' (célula vazia) ou '`#`' (célula ocupada).

Depois de ler toda a grade, a interação começa com sua primeira consulta.

Interação

Para realizar uma consulta, você deve imprimir uma linha em um destes formatos:

- `"? U"` — tentar mover-se para cima;
- `"? R"` — tentar mover-se para a direita;
- `"? D"` — tentar mover-se para baixo;
- `"? L"` — tentar mover-se para a esquerda.

Após cada tentativa de movimento, o juiz imprimirá um inteiro que será `0` se o movimento for inválido (porque a célula era ocupada ou estava fora da grade) e `1` se o movimento foi válido (e a posição do robô mudou).

Uma vez que você descubra a posição atual do robô, deve reportá-la imprimindo:

- `"! x y"` — indicando que a posição **atual** do robô é `(x, y)` (`1 ≤ x ≤ n, 1 ≤ y ≤ m`).

Depois disso, o programa deve terminar.

Se em qualquer momento da interação você ler `-1` em vez de uma resposta válida, o programa deve terminar imediatamente, recebendo `Wrong answer` por ter feito uma consulta inválida. Não terminar imediatamente pode resultar em um veredicto arbitrário, pois o programa seguirá lendo de um **stream** fechado.

A interação neste problema é **adaptativa**, o que significa que o juiz pode alterar a posição inicial do robô a qualquer momento, desde que a nova posição seja consistente com todas as consultas e respostas anteriores.

Recordatório

Depois de imprimir cada linha, não se esqueça de escrever o fim de linha e esvaziar (flush) o buffer de saída. Algumas linguagens não imprimem imediatamente, portanto é necessário realizar uma operação de flush manualmente:

- *Python: Adicionar o parâmetro* `flush` *ao imprimir:* `print(..., flush=True)`
- *C++: Usar* `std::cout << std::flush;` *após imprimir.*
- *C: Usar* `fflush(stdout);` *após imprimir.*
- *Outras linguagens: Consultar a documentação.*

Exemplos

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

Nota

Neste exemplo, a posição inicial do robô é $(2, 3)$, conhecida apenas pelo juiz.

Na primeira consulta ("`? D`"), o juiz retorna 1 e o robô se move uma posição para baixo, indo para $(3, 3)$.

Na segunda consulta ("`? D`"), o juiz retorna 1 e o robô se move novamente para baixo, indo para $(4, 3)$.

Na terceira consulta ("`? D`"), o juiz retorna 0, pois o robô não pode continuar descendo — sairia da grade. A posição permanece $(4, 3)$.

Na quarta consulta ("`? R`"), o juiz retorna 1 e o robô se move uma posição para a direita, indo para $(4, 4)$.

Na quinta consulta ("`? R`"), o juiz retorna 0, pois o robô não pode continuar indo para a direita. A posição permanece $(4, 4)$.

Finalmente, determina-se que o robô está em $(4, 4)$ e essa posição é reportada.

Note que este é apenas um exemplo de como a interação pode ocorrer; outras soluções válidas são possíveis.

F. Curar ou não curar

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

Em um mundo onde existem M doenças numeradas de 1 a M, e N remédios mágicos numerados de 1 a N, cada remédio possui dois efeitos:

- Cura certos tipos de doenças.
- Causa outros tipos de doenças.

Além disso, você já começa doente com um conjunto inicial de K doenças.

Se você toma um conjunto de remédios S, seu efeito total é:

- Contrair todas as doenças causadas por qualquer remédio em S.
- Curar todas as doenças curadas por qualquer remédio em S.

No final, você ficará com:

(doenças iniciais $\cup$ doenças causadas) $\setminus$ doenças curadas.

Seu objetivo é determinar se existe algum subconjunto de remédios que permita que você fique completamente saudável (sem nenhuma doença). Caso seja possível, você deve informar quantos remédios são necessários e quais são (seus índices).

Entrada

A primeira linha contém dois inteiros N e M ($1 \leq N, M \leq 2 \cdot 10^5$), o número de remédios e o número de doenças, respectivamente.

A segunda linha contém um inteiro K ($0 \leq K \leq M$), seguido de K inteiros $d_1, d_2, ..., d_K$ ($1 \leq d_i \leq M$), que representam as doenças iniciais que você possui. Se $K = 0$, a linha conterá apenas o zero.

Cada uma das N linhas seguintes descreve um remédio. Cada linha contém primeiro um inteiro a_i ($0 \leq a_i \leq M$), seguido de a_i inteiros distintos $c_{i,1}, c_{i,2}, ..., c_{i,a_i}$ ($1 \leq c_{i,j} \leq M$), representando as doenças que o remédio i cura. Em seguida, contém um inteiro b_i ($0 \leq b_i \leq M$), seguido de b_i inteiros distintos $p_{i,1}, p_{i,2}, ..., p_{i,b_i}$ ($1 \leq p_{i,j} \leq M$), que representam as doenças que o remédio i provoca.

Garante-se que todos os inteiros obedecem aos intervalos indicados e que, dentro de cada lista de doenças curadas ou causadas, não há repetições. Também é garantido que a soma de todos os a_i e a soma de todos os b_i não ultrapassam 10^6.

Saída

Se for possível curar todas as doenças iniciais usando algum subconjunto de remédios:

- Na primeira linha, imprima a palavra `"YES"`.
- Na segunda linha, imprima um inteiro L, o tamanho do subconjunto utilizado.
- Na terceira linha, imprima L inteiros distintos separados por espaços, correspondendo aos índices dos remédios usados (numerados de 1 a N). Se existirem múltiplas soluções, qualquer uma delas é válida.

Caso não seja possível ficar completamente saudável, imprima apenas a palavra `"NO"`.

Exemplos

Entrada 1	Saída 1
3 5 1 5 1 2 0 1 5 1 3 1 3 1 2	YES 3 1 2 3

Entrada 2	Saída 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

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

Fernanda e Leticia são duas grandes amigas. Todos os dias competem para ver quem consegue vencer a outra. Cada dia escolhem um jogo diferente; para hoje decidiram jogar um chamado "*Achicando-ando*".

O jogo consiste em um tabuleiro de comprimento n, onde cada célula contém a letra `F` ou `L`, e começa-se com um intervalo inicial $l = 1$ e $r = n$.

Jogam por turnos e Fernanda sempre começa. Em cada turno, a jogadora deve escolher um subintervalo* de comprimento $\lfloor \frac{r-l+1}{2} \rfloor$ no qual pelo menos metade das letras coincide com a letra que corresponde à jogadora: `F` para Fernanda e `L` para Leticia. Ao escolher um subintervalo definido por i e j ($i \leq j$), esse subintervalo passa a ser o novo intervalo para o turno seguinte, isto é, agora $l = i$ e $r = j$. A primeira jogadora que não conseguir escolher um subintervalo válido perde.

Paz, que está observando Fernanda e Leticia jogarem, se interessa pelo jogo e quer determinar se existe uma forma de Fernanda vencer Leticia, assumindo que ambas jogam de maneira ótima. Paz está muito ocupada organizando Codificadas, então pede sua ajuda para descobrir a vencedora.

** Um subintervalo dentro do intervalo $[l, r]$ é definido escolhendo-se dois índices i e j tais que $l \leq i \leq j \leq r$. Esse subintervalo corresponde aos valores $a_i, a_{i+1}, ..., a_j$.*

Entrada

A primeira linha da entrada contém um inteiro n ($2 \leq n \leq 10^5$), correspondente ao comprimento do tabuleiro.

A segunda linha contém uma sequência de letras sem espaços, correspondente ao tabuleiro do jogo. As letras podem ser apenas `F` ou `L`.

Saída

Se Fernanda for a vencedora, a saída deve conter apenas a palavra `Fernanda`; caso contrário, deve conter a palavra `Leticia`.

Pontuação

Este problema é avaliado com subtarefas. Para obter a pontuação total, sua solução deve funcionar corretamente em todos os casos de cada subtarefa.

- **Subtarefa 1 (60 pontos):** $n \leq 1000$. Sua solução receberá estes pontos se resolver **todos** os casos em que n é no máximo 1000.
- **Subtarefa 2 (40 pontos):** Sem restrições adicionais. Esta subtarefa inclui casos com até $n \leq 10^5$. Sua solução receberá estes pontos se resolver corretamente esses casos.

Exemplos

Entrada 1	Saída 1
11 FLFFFFFLLLF	Fernanda

Entrada 2	Saída 2
32 FFFLLLFLLLLLFLFLFLLLFLLLFLFLFL	Leticia

Nota

Para o primeiro caso, este é um exemplo possível de turnos:

- O intervalo inicial é $l = 1$ e $r = 11$. Fernanda deve escolher um subintervalo de comprimento 5. Ela escolhe o subintervalo $l = 2$ e $r = 6$. Esse subintervalo contém 4 `F`, satisfazendo a condição de ter ao menos metade de `F`.
- O intervalo atual é $l = 2$ e $r = 6$. Leticia deve escolher um subintervalo de comprimento 2. Leticia só pode escolher o subintervalo $l = 2$ e $r = 3$, pois é o único que contém ao menos 1 letra `L`.
- O intervalo atual é $l = 2$ e $r = 3$. Fernanda escolhe o subintervalo $l = 3$ e $r = 3$, já que é o único possível.
- Leticia não pode escolher um subintervalo de comprimento 0, portanto Fernanda vence.

H. Múltiplos de 3?

Limite de tempo: 1 segundos

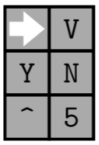
Limite de memória: 256 megabytes

Este problema possui um visualizador associado. Clique aqui para vê-lo.

Carla está jogando um jogo de tabuleiro. Esse jogo usa um *tabuleiro*, que é uma grade retangular semelhante ao tabuleiro de xadrez, mas em vez de 8 linhas e 8 colunas, este possui n linhas e m colunas.

Cada célula do tabuleiro contém um símbolo, que pode ser qualquer um dos seguintes: `<`, `>`, `V`, `^`, `N`, `Y`, `.`, ou um dígito de `0` a `9`.

Antes de começar a jogar, Carla escolhe um inteiro positivo x e coloca uma figurinha sobre a célula da primeira linha e primeira coluna, apontando para a direita.



$x = 148$

Figure 5: Configuração inicial de um tabuleiro de exemplo e um número x escolhido por Carla. Note a figurinha sobre a célula superior esquerda.

Em seguida, Carla segue o manual de instruções, que detalha os passos a seguir:

1. Mover a figurinha uma célula na direção para a qual ela aponta. Se a figurinha sair do tabuleiro, Carla **perde** e o jogo termina.
2. Dependendo do símbolo escrito na célula para a qual a figurinha se moveu, realizar uma ação:

   `<`

   Rotacionar a figurinha para que aponte para a esquerda.

   `>`

   Rotacionar a figurinha para que aponte para a direita.

   `V`

   Rotacionar a figurinha para que aponte para baixo.

   `^`

   Rotacionar a figurinha para que aponte para cima.

   `N`

   Apagar o último dígito de x. Se após apagar não restarem dígitos, Carla **perde** e o jogo termina.

   `Y`

   Apagar o último dígito de x. Se após apagar não restarem dígitos, Carla **ganha** e o jogo termina.

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

   Considere t como o dígito escrito na célula ($0 \leq t \leq 9$), e d como o último dígito de x ($0 \leq d \leq 9$):

3. Se $t < d$: girar a figurinha 90° no sentido horário.

4. Se $t > d$: girar a figurinha 90° no sentido anti-horário.
5. Se $t = d$: não realizar nenhuma ação.

   Não realizar nenhuma ação.
6. Voltar ao passo 1 e repetir. Se o jogo nunca terminar, considera-se que Carla **perde**.

Para exemplificar, vejamos como Carla jogaria com o tabuleiro anterior e com $x = 34$:

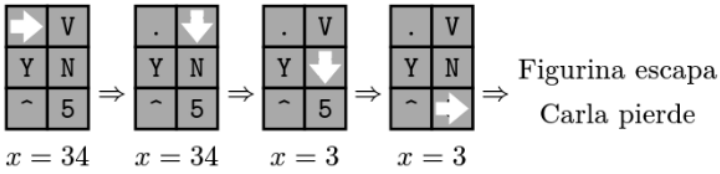


Figure 6: Configuração inicial de um tabuleiro de exemplo e um número x escolhido por Carla. Note a figurinha sobre a célula superior esquerda.

Entretanto, se Carla tivesse escolhido $x = 87$, ela teria vencido:

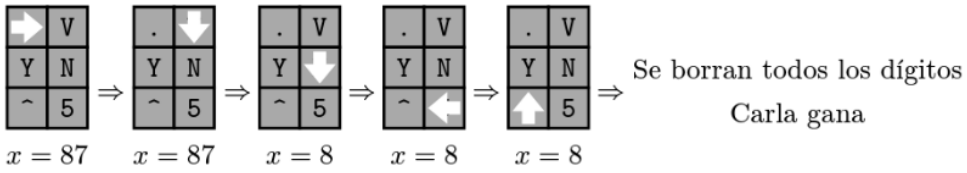


Figure 7: Configuração inicial de um tabuleiro de exemplo e um número x escolhido por Carla. Note a figurinha sobre a célula superior esquerda.

Ou seja, para cada tabuleiro existem algumas escolhas de x que são vencedoras e outras que são perdedoras.

Há alguns meses, Carla quis construir um tabuleiro *interessante*. Um tabuleiro interessante é aquele em que as escolhas vencedoras de x são exatamente os múltiplos de 3. Ela acha que encontrou a solução, mas não tem certeza. Você pode ajudá-la escrevendo um programa que determine se um tabuleiro dado é interessante ou não?

Entrada

A primeira linha contém dois inteiros n e m, o número de linhas e colunas, respectivamente ($1 \leq n, m \leq 100$).

Em seguida, haverá n linhas, cada uma com m caracteres do alfabeto "`<>V^NY.0123456789}`". O j-ésimo caractere da i-ésima linha representa o símbolo na célula da linha i (de cima para baixo) e coluna j (da esquerda para a direita).

Saída

Se o tabuleiro for interessante, imprima `"INTERESANTE"`. Caso contrário, imprima `"FOME"`.

Exemplos

Entrada 1	Saída 1
3 2 .V YN ^5	FOME

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

Nota

O primeiro tabuleiro de exemplo não é interessante, pois existem escolhas de x tais que Carla perde mesmo quando x é múltiplo de 3, por exemplo $x = 9$. Além disso, existem escolhas de x não múltiplos de 3 para as quais Carla ganha, como $x = 61$.

O segundo tabuleiro de exemplo é interessante, pois Carla ganha para todo x positivo múltiplo de 3 e perde para todo x positivo que não é múltiplo de 3.