A. Hello [nome]!

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

Javiera está começando a aprender programação e quer que seu primeiro programa seja bem simples. Uma amiga sugere que ela escreva um código que mostre na tela a mensagem `Hello [nome]!`, onde `[nome]` deve ser substituído por uma palavra informada pelo usuário.

Entrada

A primeira linha consiste em uma única palavra de no máximo 10 caracteres do alfabeto inglês, sem espaços nem acentos.

Saída

A saída deve mostrar o texto: `Hello [nome]!` onde `[nome]` é substituído pela palavra informada na entrada. Consulte os exemplos para melhor compreensão.

Exemplos

Entrada 1	Saída 1
`javiera`	`Hello javiera!`

Entrada 2	Saída 2
`ana`	`Hello ana!`

Entrada 3	Saída 3
`sofia`	`Hello sofia!`

B. T, C ou S

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

Maria da Paz, organizadora da competição *Codificadas*, está empolgada com a primeira edição. Ela é uma pessoa muito perfeccionista e gosta que tudo esteja equilibrado.

Para o conteúdo e as comunicações da competição, Maria da Paz gosta que todos os textos tenham um equilíbrio TCS. O equilíbrio TCS consiste em que, em um texto, as letras "T", "C" e "S" apareçam a mesma quantidade de vezes.

Maria da Paz não tem tempo para revisar todos os textos da competição, então ela pede a sua ajuda para determinar se um texto tem a mesma quantidade de letras "T", "C" e "S".

Entrada

A primeira linha da entrada consiste em uma sequência de caracteres do alfabeto inglês, todos em maiúsculas, sem acentos e sem espaços. Essa sequência pode ter um tamanho de até 10.000 caracteres.

Saída

A saída deve consistir na palavra "`YES`" se, na sequência de caracteres, houver a mesma quantidade de letras "T", "C" e "S". Caso contrário, a saída deve ser a palavra "`NO`".

Exemplos

Entrada 1	Saída 1
`TCS`	`YES`

Entrada 2	Saída 2
`TCTCTCTSTSTSCT`	`NO`

Entrada 3	Saída 3
`TCSCODIFICADASTST`	`YES`

Nota

No primeiro exemplo, na palavra "TCS" cada letra aparece exatamente uma vez. Portanto, satisfaz a condição de Maria da Paz.

No segundo exemplo, a letra "T" aparece 7 vezes, a letra "C" aparece 4 vezes e a letra "S" aparece 3 vezes. Portanto, não satisfaz a condição de Maria da Paz.

No terceiro exemplo, a letra "T" aparece 3 vezes, a letra "C" aparece 3 vezes e a letra "S" aparece 3 vezes. Portanto, satisfaz a condição de Maria da Paz.

C. Mais Votado

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

Após garantir o equilíbrio TCS nos textos, Maria da Paz deu o passo seguinte: construir um analisador de texto. Seu novo programa deve estudar a frequência das palavras usadas por um grupo de pessoas.

Ela recebeu muitos textos, por isso criar esse analisador é importante. No entanto, não tem tempo porque está organizando a competição *Codificadas*. Por isso, ela pede a sua ajuda para fazer um programa que indique quais palavras cada pessoa disse, determine quais foram ditas por todas e mostre essas palavras ordenadas da maior para a menor frequência.

Entrada

A primeira linha da entrada consiste em um inteiro n ($1 \leq n \leq 100$), correspondente à quantidade de pessoas no grupo a ser estudado.

Para cada pessoa, a primeira linha conterá um inteiro p ($1 \leq p \leq 100$), correspondente à quantidade de palavras usadas por essa pessoa. Em seguida vêm p linhas, onde cada uma contém uma palavra. Cada palavra é formada apenas por letras minúsculas do alfabeto inglês, sem espaços, acentos ou cedilha. Cada palavra pode ter no máximo 100 caracteres.

Saída

A saída deve mostrar **as palavras que foram usadas por todas as pessoas.** Escreva uma palavra por linha.

A ordem deve ser a seguinte:

1. **Da maior para a menor frequência**: as palavras mais usadas aparecem primeiro.
2. **Em caso de empate de frequência**, elas devem ser ordenadas em **ordem lexicográfica decrescente**, ou seja, a palavra lexicograficamente maior aparece antes. Isso é parecido com a ordem de um dicionário, mas ao contrário: em vez de "abelha" aparecer antes de "zebra" aqui "zebra" viria antes de "abelha"

Exemplos

Entrada 1	Saída 1
4 3 animal amigos casa 2 amigos casa 5 tcs codificadas amigos amigos casa 3 amigos amor casa	amigos casa

Entrada 2	Saída 2
1 5 tcs codificadas tcs come tcs	tcs come codificadas

Nota

No primeiro exemplo participam 4 pessoas:

- A primeira usou 3 palavras: “animal”, “amigos” e “casa”.
- A segunda usou 2 palavras: “amigos” e “casa”.
- A terceira usou 5 palavras: “tcs”, “codificadas”, “amigos”, “amigos” e “casa”.
- A quarta usou 3 palavras: “amigos”, “amor” e “casa”.

As palavras usadas por todas as pessoas foram “amigos” e “casa”. A palavra “amigos” apareceu 5 vezes e “casa” 4 vezes. Por isso, “amigos” é mostrada na primeira linha e “casa” na segunda.

No segundo exemplo participa apenas uma pessoa, que utilizou 5 palavras. Como não há mais integrantes no grupo, considera-se que todas essas palavras foram ditas por todas as pessoas. A palavra “tcs” apareceu 3 vezes, por isso vai na primeira linha. As palavras “codificadas” e “come” apareceram uma vez cada uma; como lexicograficamente “codificadas” é menor que “come”, “codificadas” vai na segunda linha e “come” na terceira.

Definição de ordem lexicográfica

A ordem lexicográfica é a mesma ordem em que os dicionários organizam as palavras, e **a maioria das linguagens de programação já traz implementações prontas para compará-las dessa forma.**

Dada uma string a de comprimento n e uma string b de comprimento m, diz-se que a é lexicograficamente menor que b se uma das seguintes condições for satisfeita:

- **Primeira diferença**: existe uma posição i na qual $a[i] \neq b[i]$, e o caractere em $a[i]$ é menor que o caractere em $b[i]$ segundo o alfabeto (ou, em geral, o valor do seu código de caractere).

  Exemplo: “calabaza” é menor que “calazaza” porque coincidem em “cala”, mas na posição 5 “b” é menor que “z”.

- **Prefixo**: Se todos os caracteres coincidem até que uma das strings termine, então a string mais curta é considerada lexicograficamente menor.

  Exemplo: “casa” é menor que “casamento” porque “casa” aparece inteira no início e tem menos caracteres.

D. Jardim Botânico

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

Fernanda decide visitar o majestoso Jardim Botânico de Santiago, um oásis de biodiversidade no Chile.

Enquanto passeia pelos caminhos, ela encontra uma seção especial com arranjos florais. Esses arranjos não são meramente decorativos: foram projetados por botânicos excêntricos que acreditam na simetria numérica da natureza.

Cada arranjo floral é representado por uma grade 2D, onde alguns espaços estão ocupados e outros estão vazios. As células ocupadas são marcadas com o caractere "∗" e as vazias com um ".".

A grade representa exatamente um dos dois seguintes tipos de flores raras:

- **Double Petal Flower**, representada por um formato de 2 na grade.
- **Triple Corolla Flower**, representada por um formato de 3 na grade.

```
........
...****.
......*.
......*.
..*****.
..*.....
..*****.
........
```

Exemplo de uma **Double Petal Flower**

```
........
..****..
.....*..
...***..
.....*..
.....*..
.*****..
........
```

Exemplo de uma **Triple Corolla Flower**

Essas flores são difíceis de distinguir, pois podem aparecer na grade de muitas formas diferentes: **podem aparecer em qualquer posição dentro da grade, variar no comprimento de suas linhas horizontais ou verticais,** e até mesmo estar cercadas por espaços vazios que dificultam reconhecer sua estrutura.

O que nunca muda é o padrão essencial de cada tipo de flor (o formato de um 2 ou o formato de um 3), que permite identificá-las independentemente de seu tamanho exato ou localização na grade.

Os botânicos estão procurando ajuda para classificar seus complexos arranjos florais. Ajude-os a determinar qual tipo de flor está representada em um certo arranjo floral.

Entrada

A primeira linha da entrada contém dois inteiros n e m ($5 \leq n \leq 1000$, $2 \leq m \leq 1000$), que correspondem à quantidade de linhas e colunas da grade.

Em seguida vêm n linhas, cada uma com m caracteres, que descrevem a grade:

- O caractere $*$ indica que a célula está ocupada no arranjo floral.
- O caractere $.$ indica que a célula está vazia.

Garante-se que a grade corresponde exatamente a um dos dois tipos de flores: **Double Petal Flower** ou **Triple Corolla Flower**.

Saída

A saída deve consistir em uma única linha com "**Double Petal Flower**" ou "**Triple Corolla Flower**", de acordo com o tipo da flor representada na grade.

Exemplos

Entrada 1	Saída 1
7 8 ....*****.. .....*.. .....*.. ..****.. ..*..... ..****..	Double Petal Flower

Entrada 2	Saída 2
7 8 .....*** ....* ....* ....* ......** ....* ..******	Triple Corolla Flower

Entrada 3	Saída 3
5 6 ****** .....* ..**** .....* ******	Triple Corolla Flower

Definição da forma das flores

- `Double Petal Flower`

  Para que uma grade corresponda a essa flor, deve seguir o seguinte padrão:

  1. De cima para baixo, encontra-se primeiro uma linha horizontal com pelo menos dois *.
  2. No * mais à direita dessa linha horizontal começa uma linha vertical com pelo menos três *.
  3. No * mais abaixo dessa linha vertical começa uma segunda linha horizontal com pelo menos dois * para a esquerda.
  4. No * mais à esquerda dessa segunda linha horizontal começa outra linha vertical com pelo menos três *.
  5. Finalmente, no * mais abaixo dessa segunda linha vertical começa uma última linha horizontal com pelo menos dois * para a direita.

  Para melhor compreensão, consulte o exemplo de entrada 1.

- `Triple Corolla Flower`

  Para que uma grade corresponda a essa flor, deve seguir o seguinte padrão:

  1. De cima para baixo, aparece primeiro uma linha horizontal com pelo menos dois *.
  2. No * mais à direita dessa linha horizontal começa uma linha vertical com pelo menos cinco *.
  3. Em algum ponto intermediário dessa linha vertical deve existir um * que inicie uma linha horizontal para a esquerda com pelo menos dois *.
  4. Finalmente, no * mais abaixo da linha vertical começa uma linha horizontal para a esquerda com pelo menos dois *.

  Para melhor compreensão, consulte os exemplos de entrada 2 e 3.

E. Os Esquilos de Fibonacho

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

No século XII, na república de Pisa, existiu um famoso matemático chamado Ignacio Bigollo Pisano, mas que hoje é conhecido mais popularmente como Fibonacho.

Uma de suas descobertas mais famosas foram as Sequências de Fibonacho, que ele descobriu em um verão em que ficou cuidando dos esquilos que viviam com seu tio Gabriel.

No começo do verão, Gabriel lhe entregou as chaves de sua torre, onde tinha C esquilos. Depois de uma semana a quantidade de esquilos aumentou bastante, e depois de duas semanas aumentou ainda mais. Passado algum tempo, Fibonacho percebeu que, semana após semana, a quantidade de esquilos se multiplicava por A, e além disso se juntavam mais B vindos dos arredores.

Em outras palavras, se a quantidade inicial era $C = 5$, o multiplicador era $A = 2$, e a quantidade que se juntava a cada semana era $B = 3$, então na primeira semana havia 5 esquilos, na segunda havia 13, na terceira 29, na quarta 61, e assim por diante.

Fibonacho, no futuro, encontraria um valor matemático fundamental para essa sequência de números, mas por agora está preocupado de que a torre possa colapsar com tantos esquilos.

Ele sabe que quando a quantidade de esquilos superar o valor D, a torre vai, no mínimo, inclinar-se. Fibonacho quer que você diga quantas semanas ele tem antes de precisar chamar seu tio Gabriel para vir controlar a situação.

Entrada

A primeira linha contém os quatro valores A, B, C e D separados por espaços ($1 \leq A, B, C \leq 100$, $1 \leq D \leq 10^{12}$). Garante-se que $C \leq D$.

Importante: *O valor de D pode exceder a capacidade de um `int`. Certifique-se de usar um tipo de dado que suporte números grandes (por exemplo, `long long` em C++, `long` em Java). Em Python isso não é um problema, pois os inteiros não têm limite de tamanho.*

Saída

A saída contém uma linha com um número que indica quantas semanas a torre aguentará antes de inclinar-se pelos esquilos.

Exemplos

Entrada 1	Saída 1
2 3 5 61	4

Entrada 2	Saída 2
1 2 1 10	5

Nota

O primeiro exemplo é o visto no enunciado: na quarta semana a torre terá 61 esquilos, que é o limite que a torre aguenta; se Fibonacho esperar mais uma semana, a torre já terá colapsado.

No segundo exemplo a torre terá 1, 3, 5, 7 e 9 esquilos em cada semana. Na semana 6, se se juntarem mais dois esquilos, a torre colapsa, então a resposta é 5.

F. Cruzamento Equidistante

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

Há n monumentos distribuídos em um plano, cada um localizado em uma coordenada distinta. Os organizadores querem conectar alguns desses monumentos por meio de passarelas retas, formando segmentos entre pares de monumentos.

No entanto, há uma condição artística muito particular: desejam encontrar duas passarelas, ou seja, dois segmentos formados por pares distintos de monumentos, que se cruzem exatamente no ponto médio de ambos os segmentos.

Formalmente, você deve determinar se existem quatro monumentos distintos, rotulados A, B, C, D, tais que:

- O segmento AB conecta os pontos A e B.
- O segmento CD conecta os pontos C e D.
- Os segmentos AB e CD se cruzam em um ponto P.
- P é simultaneamente o ponto médio dos segmentos AB e CD.

Entrada

A primeira linha contém um inteiro n ($4 \leq n \leq 1000$), o número de monumentos. Cada uma das n linhas seguintes contém dois inteiros x_i, y_i ($0 \leq x_i, y_i \leq 10^9$), as coordenadas do i-ésimo monumento. Garante-se que todos os pontos são distintos.

Saída

A saída deve consistir na palavra "`YES`" se existirem dois segmentos, cada um entre um par distinto de pontos, que se cruzem exatamente no ponto médio de ambos. Caso contrário, a saída deve ser a palavra "`NO`".

Exemplos

Entrada 1	Saída 1
5 3 14 5 16 11 16 12 1 13 18	YES

Entrada 2	Saída 2
5 2 14 11 13 12 20 14 5 14 15	NO

Nota

No primeiro exemplo a resposta é "`YES`". De fato, podemos escolher $A = (3, 14)$, $B = (13, 18)$, $C = (5, 16)$, $D = (11, 16)$. Dessa forma, os segmentos AB e CD intersectam em $P = (8, 16)$, que é precisamente o ponto médio de AB e CD.

No segundo exemplo é possível verificar que não há pontos que satisfaçam o pedido. A resposta é "`NO`".

G. Papel de Parede

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

É o aniversário do seu padrinho e você está procurando o que dar de presente. Um dia ele lhe diz: “olha, teci um tapete muito bonito”. O tapete se vê assim:

```
############################
#.##.##.##.##.##.##.##.##.#
############################
###...######...######...###
#.#...#.##.#...#.##.#...#.#
###...######...######...###
############################
#.##.##.##.##.##.##.##.##.#
############################
#########.........#########
#.##.##.#.........#.##.##.#
#########.........#########
###...###.........###...###
#.#...#.#.........#.#...#.#
###...###.........###...###
#########.........#########
#.##.##.#.........#.##.##.#
#########.........#########
############################
#.##.##.##.##.##.##.##.##.#
############################
###...######...######...###
#.#...#.##.#...#.##.#...#.#
###...######...######...###
############################
#.##.##.##.##.##.##.##.##.#
############################
```

Ele explica que o construiu usando a seguinte figura como base:

```
###
#.#
###
```

A essa figura ele chamou de *tapete 1*. Depois, tomou cada `#` da figura e o substituiu pela figura base, e tomou cada `.` e o substituiu por uma figura de tamanho $3x3$ que contém apenas o caractere `.` — assim formou o *tapete* 2:

```
#########
#.##.##.#
#########
###...###
#.#...#.#
###...###
#########
#.##.##.#
#########
```

Finalmente, o *tapete* 3 é obtido a partir do *tapete* 2, substituindo novamente cada `#` pela figura base e cada `.` por uma figura de $3x3$ feita de pontos. O tapete que o seu padrinho tem é justamente o *tapete* 3.

Vendo isso, você tem uma ideia — e se eu der de presente um papel de parede que use a mesma técnica? Claro que não pode começar com a mesma figura base, senão seria igual ao que ele já tem, então o que você vai fazer é começar com uma figura base qualquer de nxn, e criar o "papel de parede k".

Para isso, faz-se o mesmo procedimento do tapete, mas usando essa nova figura base. Ou seja, a figura base é o *papel de parede* 1, e o *papel de parede* k é formado a partir do *papel de parede* $k-1$, substituindo cada `#` pela figura base, e cada `.` por uma figura de nxn feita apenas de pontos.

Você pode se referir aos exemplos para maior clareza.

Entrada

A primeira linha contém os valores n e k ($2 \leq n \leq 4$, $1 \leq k \leq 5$) que indicam que a figura tem dimensões nxn e a quantidade de iterações que você vai fazer para criar o papel de parede.

As próximas n linhas são a figura. Cada linha tem tamanho n e é formada unicamente pelos caracteres `#` e `.`.

Saída

A saída é o *papel de parede* k, escrito com caracteres `#` e `.`.

Exemplos

Entrada 1	Saída 1
2 3 #. ##	#....... ##...... #.#..... ####.... #...#... ##..##.. #.#.#.#. ########

Entrada 2	Saída 2
3 2 #.. #.# .#.	#....... #.#..... .#..... #.....#.. #.#...#.# .#.....#. ...#..... ...#.#... ....#....

Entrada 3	Saída 3
4 1 .### #### #### .#.#	.### #### #### .#.#

H. Depósitos

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

Na cidade linear de Distribuilândia, há n pessoas que precisam receber suprimentos de algum dos k depósitos locais. Tanto as pessoas quanto os depósitos estão localizados ao longo da única rua principal da cidade, que pode ser modelada como uma linha reta (o eixo X).

Cada depósito tem uma capacidade limitada: o depósito i pode atender no máximo c_i pessoas. Para evitar descontentamento entre os habitantes, o governo quer que a pessoa mais distante de seu depósito designado esteja o mais perto possível.

Sua tarefa é determinar se é possível designar todas as pessoas a algum depósito (respeitando as capacidades), de forma que a distância máxima entre uma pessoa e seu depósito designado seja a menor possível. Em outras palavras, você deve encontrar a menor distância d tal que exista uma designação válida em que toda pessoa esteja a uma distância de no máximo d de seu depósito.

Entrada

A primeira linha contém dois inteiros n e k ($1 \leq n, k \leq 2 * 10^5$): o número de pessoas e o número de depósitos.

A segunda linha contém n inteiros $p_1, p_2, ..., p_n$ ($0 \leq p_i \leq 10^9$), as posições das pessoas ao longo da rua.

A terceira linha contém k inteiros $b_1, b_2, ..., b_k$ ($0 \leq b_i \leq 10^9$), as posições dos depósitos. Finalmente, a quarta linha contém k inteiros $c_1, c_2, ..., c_k$ ($1 \leq c_i \leq n$), a capacidade de cada depósito. Garante-se que todas as posições (de pessoas e depósitos) são distintas.

Saída

Imprima um único inteiro: a menor distância máxima possível d tal que todas as pessoas possam ser designadas a algum depósito a uma distância de no máximo d, sem exceder a capacidade de nenhum depósito.

Exemplos

Entrada 1	Saída 1
4 3 2 4 7 10 1 5 9 2 2 2	2

Nota

A designação ótima é obtida da seguinte forma:

- Pessoa 1 ao depósito 1: distância 1
- Pessoa 2 ao depósito 2: distância 1
- Pessoa 3 ao depósito 2: distância 2
- Pessoa 4 ao depósito 3: distância 1

Nenhuma outra designação obtém uma distância máxima menor.

I. Queijos Infectados

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

No remoto e delicioso mundo dos queijos, uma catastrófica praga de fungos começou a se espalhar.

Um único fungo, com uma voracidade insaciável, ameaça consumir todos os valiosos blocos de queijo.

Sua missão é estratégica: remover a menor quantidade possível de células com queijo para evitar que o fungo consiga alcançar pelo menos um bloco de queijo.

A zona afetada é representada como uma grade de n linhas e m colunas. Cada célula pode conter um dos seguintes elementos:

- **Queijo:** Representado pelo caractere `Q`.
- **Fungo:** Representado pelo caractere `H`. Existe exatamente uma célula desse tipo.
- **Vazia:** Representada pelo caractere `.`.

O fungo pode se expandir para células adjacentes nas quatro direções (cima, baixo, esquerda e direita), mas apenas para células que contenham queijo. Quando isso acontece, ele consome o queijo e a célula passa a estar ocupada pelo fungo.

Seu objetivo é encontrar a **mínima quantidade de células de queijo que devem ser removidas** antes que o fungo comece a se expandir, de forma que pelo menos uma célula com queijo permaneça inalcançável para ele.

Entrada

A primeira linha da entrada contém dois inteiros n e m ($1 \leq n, m \leq 1000$), que representam a quantidade de linhas e colunas da grade, respectivamente.

Em seguida, vêm n linhas, cada uma com m caracteres, descrevendo a grade. Cada caractere será `Q` (queijo), `H` (fungo) ou `.` (vazio). Garante-se que haverá exatamente uma célula com `H`.

Saída

A saída deve conter um único inteiro: a mínima quantidade de células de queijo que devem ser convertidas em células vazias para atingir o objetivo, ou -1 se for impossível garantir que sobre um queijo inalcançável para o fungo.

Exemplos

Entrada 1	Saída 1
3 5 .QQQQ .QQQQ .HQQ.	2

Entrada 2	Saída 2
3 5 Q.QQQ .H.QQ Q.QQQ	0

Entrada 3	Saída 3
3 3 .Q. QHQ .Q.	-1

J. Múltiplos de 3

Limite de tempo: 0.5 segundos

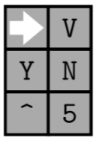
Limite de memória: 256 megabytes

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

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

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

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



$x = 148$

Figure 1: Configuração inicial de um tabuleiro de exemplo e um número x escolhido por Carla. Note a peça sobre a célula superior esquerda.

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

1. Mover a peça uma célula na direção para a qual está apontando. Se a peça sair do tabuleiro, Carla **perde** e o jogo termina.
2. De acordo com o símbolo escrito na célula para a qual a peça se moveu, realizar uma ação:

   `<`

   Rotacionar a peça para que aponte para a esquerda.

   `>`

   Rotacionar a peça para que aponte para a direita.

   `V`

   Rotacionar a peça para que aponte para baixo.

   `^`

   Rotacionar a peça 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 peça 90° no sentido horário.
4. Se $t > d$: girar a peça 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 $x = 34$:

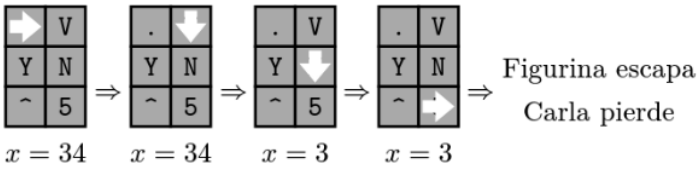


Figure 2: Configuração de um tabuleiro de exemplo e um número x escolhido por Carla. Note a peça sobre a célula superior esquerda.

No entanto, se Carla tivesse escolhido $x = 87$, teria vencido:

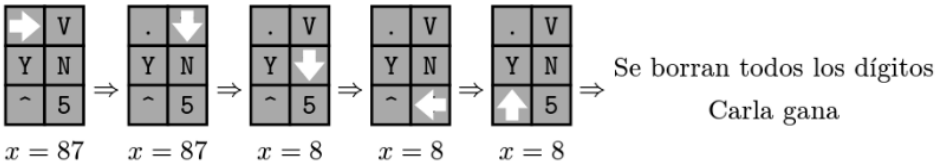


Figure 3: Configuração de um tabuleiro de exemplo e um número x escolhido por Carla. Note a peça sobre a célula superior esquerda.

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

Carla se divertiu muito com esse jogo, mas já jogou todos os tabuleiros que vêm no manual e está entediada. Por isso, ela pede para você projetar um novo tabuleiro tal que as escolhas vencedoras de x sejam exatamente os múltiplos de 3. Você consegue fazer isso?

Entrada

Seu programa não receberá entrada.

Saída

Seu programa deve imprimir um tabuleiro tal que Carla vença se escolher x múltiplo de 3, e perca caso contrário.

Formalmente, imprima n linhas ($1 \leq n \leq 100$), onde todas as linhas tenham m caracteres ($1 \leq m \leq 100$) e representem uma linha do tabuleiro cada uma. Cada caractere deve ser um dentre "`<>V^NY.0123456789`". Sua resposta será considerada correta se Carla vencer nesse tabuleiro se, e somente se, escolher um múltiplo de 3 como valor inicial de x.

Exemplos

Entrada 1	Saída 1

	`.V`
	`YN`
	`^5`

Pontuação

Para verificar se sua solução está correta, Carla jogará seu tabuleiro com diferentes escolhas de x. Dizemos que seu tabuleiro se *comporta bem* com uma escolha x múltiplo de 3 se, e somente se, Carla vencer ao jogar nesse tabuleiro com esse x. Por outro lado, para uma escolha x que não seja múltiplo de 3, dizemos que seu tabuleiro se *comporta bem* se, e somente se, Carla perder ao jogar nesse tabuleiro.

Na tabela a seguir, dizemos que seu tabuleiro *satisfaz* uma linha se se comporta bem para todo o intervalo possível de valores de x. Você receberá a pontuação máxima de todas as linhas que seu tabuleiro satisfaz, ou 0 pontos se não satisfizer nenhuma linha.

Intervalo de x	Pontuação
$1 \leq x \leq 9$	10
$1 \leq x \leq 99$	20
$1 \leq x \leq 999$	30
$1 \leq x \leq 9999$	40
$1 \leq x \leq 99999$	50
$1 \leq x < \infty$	100

Em particular, você receberá pontuação máxima apenas se seu tabuleiro se comportar bem para todo x, independentemente do seu tamanho.

Nota

O tabuleiro de exemplo anterior não é correto, é apenas um exemplo para mostrar o formato da resposta.

Você pode testar seu tabuleiro com diferentes valores de x no visualizador oficial do problema.

Você pode se basear no seguinte código em Python para imprimir o tabuleiro que projetar:

```
print("""
.V
YN
^5
""")
```