A. Hello [nombre]!

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Javiera está comenzando a aprender programación y quiere que su primer programa sea muy sencillo. Una amiga le sugiere que escriba un código que muestre en pantalla el mensaje `Hello [nombre]!`, donde `[nombre]` se reemplace por un palabra ingresada por el usuario.

Entrada

La primera línea consiste en una única palabra de a lo más 10 caracteres del alfabeto inglés sin espacios ni tildes.

Salida

La salida debe mostrar el texto: `Hello [nombre]!` donde `[nombre]` se reemplaza por la palabra ingresada en la entrada. Consulta los ejemplos para una mejor comprensión.

Ejemplos

Entrada 1	Salida 1
`javiera`	`Hello javiera!`

Entrada 2	Salida 2
`ana`	`Hello ana!`

Entrada 3	Salida 3
`sofia`	`Hello sofia!`

B. T, C o S

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Paz, organizadora de la competencia *Codificadas*, está emocionada por la primera edición. Ella es una persona muy perfeccionista y le gusta que todo esté equilibrado.

Para el contenido y las comunicaciones de la competencia, a Paz le gusta que todos los textos tengan un equilibrio TCS. El equilibrio TCS consiste en que, en un texto, las letras "T", "C" y "S" aparezcan la misma cantidad de veces.

Paz no tiene tiempo para revisar todos los textos de la competencia, así que te pide ayuda para determinar si un texto tiene la misma cantidad de letras "T", "C" y "S".

Entrada

La primera línea de la entrada consiste en una secuencia de caracteres del alfabeto inglés, todos en mayúsculas, sin acentos ni tildes y sin espacios. Esta secuencia puede tener un largo de hasta $10,000$ caracteres.

Salida

La salida debe consistir en la palabra "`YES`" si en la secuencia de caracteres hay la misma cantidad de letras "T", "C" y "S". En caso contrario, la salida debe ser la palabra "`NO`".

Ejemplos

Entrada 1	Salida 1
`TCS`	`YES`

Entrada 2	Salida 2
`TCTCTCTSTSTSCT`	`NO`

Entrada 3	Salida 3
`TCSCODIFICADASTST`	`YES`

Nota

En el primer ejemplo, en la palabra "TCS" cada letra aparece exactamente una vez. Por lo que cumple la condición de Paz.

En el segundo ejemplo, la letra "T" aparece 7 veces, la letra "C" aparece 4 veces y la letra "S" aparece 3 veces. Por lo que no cumple la condición de Paz.

En el tercer ejemplo, la letra "T" aparece 3 veces, la letra "C" aparece 3 veces y la letra "S" aparece 3 veces. Por lo que cumple la condición de Paz.

C. Más Votado

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Tras asegurar el equilibrio TCS en los textos, Paz dio el siguiente paso: construir un analizador de texto. Su nuevo programa debe estudiar la frecuencia de las palabras usadas por un grupo de personas.

Ella ha recibido muchos textos, por lo que hacer este analizador es importante. Sin embargo, no tiene tiempo porque está organizando la competencia *Codificadas*. Por eso te pide ayuda para hacer un programa que indique qué palabras dijo cada persona, determine cuáles fueron dichas por todas y muestre esas palabras ordenadas de la mayor a la menor frecuencia.

Entrada

La primera línea de la entrada consiste en un entero n ($1 \leq n \leq 100$), correspondiente a la cantidad de personas en el grupo a estudiar.

Por cada persona, la primera línea contendrá un entero p ($1 \leq p \leq 100$), correspondiente a la cantidad de palabras usadas por esa persona. Luego siguen p líneas, donde cada una contiene una palabra. Cada palabra está formada solo por letras minúsculas del alfabeto inglés, sin espacios, tildes ni acentos. Cada palabra puede tener un largo máximo de 100 caracteres.

Salida

La salida debe mostrar **las palabras que fueron usadas por todas las personas**. Escribe una palabra por línea.

El orden debe ser el siguiente:

1. **De mayor a menor frecuencia**: las palabras más usadas aparecen primero.
2. **En caso de empate de frecuencia**, se ordenan en **orden lexicográfico descendente**, es decir, la palabra lexicográficamente mayor aparece antes. Esto es parecido al orden de un diccionario, pero al revés: en lugar de que “abeja” aparezca antes que “zorro”, aquí “zorro” iría antes que “abeja”.

Ejemplos

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

Entrada 2	Salida 2
1 5 tcs codificadas tcs come tcs	tcs come codificadas

Nota

En el primer ejemplo participan 4 personas:

- La primera usó 3 palabras: “animal”, “amigos” y “casa”.
- La segunda usó 2 palabras: “amigos” y “casa”.
- La tercera usó 5 palabras: “tcs”, “codificadas”, “amigos”, “amigos” y “casa”.
- La cuarta usó 3 palabras: “amigos”, “amor” y “casa”.

Las palabras usadas por todas las personas fueron “amigos” y “casa”. La palabra “amigos” apareció 5 veces y “casa” 4 veces. Por eso, “amigos” se muestra en la primera línea y “casa” en la segunda.

En el segundo ejemplo participa solo una persona, que utilizó 5 palabras. Como no hay más integrantes en el grupo, se considera que todas esas palabras fueron dichas por todas las personas. La palabra “tcs” apareció 3 veces, por lo que va en la primera línea. Las palabras “codificadas” y “come” aparecieron una vez cada una; como lexicográficamente “codificadas” es menor que “come”, “codificadas” va en la segunda línea y “come” en la tercera.

Definición de orden lexicográfico

El orden lexicográfico es el mismo orden en que los diccionarios organizan las palabras, y **la mayoría de los lenguajes de programación ya traen implementaciones listas para compararlas de esta forma.**

Dado un string a de longitud n y un string b de longitud m, se dice que a es lexicográficamente menor que b si se cumple alguna de las siguientes condiciones:

- **Primera diferencia**: existe una posición i en la que $a[i] \neq b[i]$, y el carácter en $a[i]$ es menor que el carácter en $b[i]$ según el alfabeto (o, en general, el valor de su código de carácter).

  Ejemplo: “calabaza” es menor que “calazaza” porque coinciden en “cala”, pero en la posición 5 “b” es menor que “z”.

- **Prefijo**: Si todos los caracteres coinciden hasta que uno de los strings termina, entonces el string más corto es considerado lexicográficamente menor.

  Ejemplo: “casa” es menor que “casamiento” porque “casa” aparece completo al inicio y tiene menos caracteres.

D. Jardín Botánico

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Fernanda decide visitar el majestuoso Jardín Botánico de Santiago, un oasis de biodiversidad en Chile.

Mientras pasea por los senderos, encuentra una sección especial con arreglos florales. Estos arreglos no son meramente decorativos: han sido diseñados por botánicos excéntricos que creen en la simetría numérica de la naturaleza.

Cada arreglo floral está representado por una grilla 2D donde algunos de los espacios están ocupados y otros están vacíos. Las celdas ocupadas se marcan con el carácter "`*`" y las vacías con un "`.`".

La grilla representa exactamente uno de los dos siguientes tipos de flores raras:

- **Double Petal Flower**, representada por una forma de `2` en la grilla.
- **Triple Corolla Flower**, representada por una forma de `3` en la grilla.

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

Ejemplo de una **Double Petal Flower**

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

Ejemplo de una **Triple Corolla Flower**

Estas flores son difíciles de distinguir ya que pueden aparecer en la grilla de muchas formas distintas: **pueden aparecer en cualquier posición dentro de la grilla, variar en la longitud de sus líneas horizontales o verticales,** e incluso estar rodeadas de espacios vacíos que dificultan reconocer su estructura.

Lo que nunca cambia es el patrón esencial de cada tipo de flor (la forma de un 2 o la forma de un 3) que permite identificarlas sin importar su tamaño exacto o su ubicación dentro de la grilla.

Los botánicos están buscando ayuda para clasificar sus complejos arreglos florales. Ayúdalos a determinar qué tipo de flor representa un cierto arreglo floral.

Entrada

La primera línea de la entrada contiene dos enteros n y m ($5 \leq n \leq 1000$, $2 \leq m \leq 1000$), que corresponden a la cantidad de filas y columnas de la grilla.

Luego siguen n líneas, cada una con m caracteres, que describen la grilla:

- El carácter `*` indica que la celda está ocupada en el arreglo floral.
- El carácter `.` indica que la celda se encuentra vacía.

Se garantiza que la grilla corresponde exactamente a uno de los dos tipos de flores: **Double Petal Flower** o **Triple Corolla Flower**.

Salida

La salida debe consistir en una única línea con "**Double Petal Flower**" o "**Triple Corolla Flower**" según el tipo de la flor representada en la grilla.

Ejemplos

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

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

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

Definición de la forma de las flores

- `Double Petal Flower`

  Para que una grilla corresponda a esta flor, debe seguir el siguiente patrón:

  1. De arriba hacia abajo, se encuentra primero una línea horizontal con al menos dos *.
  2. En el * más a la derecha de esta línea horizontal comienza una línea vertical con al menos tres *.
  3. En el * más bajo de esta línea vertical comienza una segunda línea horizontal con al menos dos * hacia la izquierda.
  4. En el * más a la izquierda de esta segunda línea horizontal comienza otra línea vertical con al menos tres *.
  5. Finalmente, en el * más bajo de esta segunda línea vertical comienza una última línea horizontal con al menos dos * hacia la derecha.

  Para una mejor comprensión, consulte el ejemplo de entrada 1.

- `Triple Corolla Flower`

  Para que una grilla corresponda a esta flor, debe seguir el siguiente patrón:

  1. De arriba hacia abajo, aparece primero una línea horizontal con al menos dos *.
  2. En el * más a la derecha de esta línea horizontal comienza una línea vertical con al menos cinco *.
  3. En algún punto intermedio de esta línea vertical debe existir un * que inicie una línea horizontal hacia la izquierda con al menos dos *.
  4. Finalmente, en el * más bajo de la línea vertical comienza una línea horizontal hacia la izquierda con al menos dos *.

  Para una mejor comprensión, consulte los ejemplos de entrada 2 y 3

E. Las Ardillas de Fibonacho

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

En el siglo XII, en la república de Pisa existió un famoso matemático llamado Ignacio Bigollo Pisano, pero que hoy se conoce más popularmente como Fibonacho.

Uno de sus más famosos descubrimientos fueron las Secuencias de Fibonacho, las cuales descubrió un verano que se quedó cuidando las ardillas que vivían con su tío Gabriel.

Al comienzo del verano, Gabriel le pasó las llaves de su torre, en donde tenía C ardillas. Luego de una semana la cantidad de ardillas aumentó bastante, y luego de dos semanas aumentó aún más. Luego de un tiempo, Fibonacho notó que semana a semana, la cantidad de ardillas se multiplicaba por A, y encima de esto se unían unas B más de los alrededores.

En otras palabras, si la cantidad inicial era $C = 5$, el multiplicador era $A = 2$, y la cantidad que se unía cada semana era $B = 3$, entonces en la primera semana había 5 ardillas, en la segunda había 13, en la tercera 29, en la cuarta 61, y así.

Fibonacho, en un futuro, le encontraría un valor matemático fundamental a esta secuencia de números, pero por ahora está preocupado de que la torre puede colapsar con tantas ardillas.

Él sabe que cuando la cantidad de ardillas supere el valor D, la torre va a, como mínimo, ladearse. Fibonacho quiere que le digas cuántas semanas tiene antes de que tenga que llamar a su tío Gabriel para que venga a controlar la situación.

Entrada

La primera línea contiene los cuatro valores A, B, C y D separados por espacios ($1 \leq A, B, C \leq 100$, $1 \leq D \leq 10^{12}$). Se asegura que $C \leq D$.

Importante: *El valor de D puede exceder la capacidad de un* `int`*. Asegúrate de usar un tipo de dato que soporte números grandes (por ejemplo,* `long long` *en C++,* `long` *en Java). En Python no hay problema porque los enteros no tienen límite de tamaño.*

Salida

La salida contiene una línea con un número que indica cuántas semanas aguantará la torre antes de ladearse por las ardillas.

Ejemplos

Entrada 1	Salida 1
2 3 5 61	4

Entrada 2	Salida 2
1 2 1 10	5

Nota

El primer ejemplo es el visto en el enunciado, en la cuarta semana la torre tendrá 61 ardillas que es el límite que aguanta la torre, si Fibonacho espera una semana más la torre ya habrá colapsado.

En el segundo ejemplo la torre tendrá 1, 3, 5, 7 y 9 ardillas en cada semana. En la semana 6, si se llegan a unir dos ardillas más, la torre colapsa, así que la respuesta es 5.

F. Cruce Equidistante

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Hay n monumentos distribuidos en un plano, cada uno ubicado en una coordenada distinta. Los organizadores quieren conectar algunos de estos monumentos mediante pasarelas rectas, formando segmentos entre pares de monumentos.

Sin embargo, hay una condición artística muy particular: desean encontrar dos pasarelas, es decir, dos segmentos formados por pares distintos de monumentos, que se crucen exactamente en el punto medio de ambos segmentos.

Formalmente, debes determinar si existen cuatro monumentos distintos, etiquetados A, B, C, D, tales que:

- El segmento AB conecta los puntos A y B.
- El segmento CD conecta los puntos C y D.
- Los segmentos AB y CD se cruzan en un punto P.
- P es simultáneamente el punto medio de segmentos AB y CD.

Entrada

La primera línea contiene un entero n $(4 \leq n \leq 1000)$, el número de monumentos. Cada una de las siguientes n líneas contiene dos enteros x_i, y_i $(0 \leq x_i, y_i \leq 10^9)$, las coordenadas del i-ésimo monumento. Se garantiza que todos los puntos son distintos.

Salida

La salida debe consistir en la palabra "`YES`" si existen dos segmentos, cada uno entre un par distinto de puntos, que se crucen exactamente en el punto medio de ambos. En caso contrario, la salida debe ser la palabra "`NO`".

Ejemplos

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

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

Nota

En el primer ejemplo la respuesta es "`YES`". En efecto podemos elegir $A = (3, 14)$, $B = (13, 18)$, $C = (5, 16)$, $D = (11, 16)$. De esta manera los segmentos AB y CD intersectan en $P = (8, 16)$ que es precisamente el punto medio de AB y CD.

En el segundo ejemplo es posible verificar que no hay puntos que satisfacen lo pedido. La respuesta es "`NO`".

G. Papel Tapiz

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Es el cumpleaños de tu padrino y estás buscando qué regalarle. Un día tu padrino te dice "mira, tejí una alfombra muy bonita". La alfombra se ve así:

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

Te explica que la armó usando la siguiente figura como base:

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

A esta figura la llamó la *alfombra 1*. Luego, tomó cada `#` de la figura y lo reemplazó por la figura base, y tomó cada `.` y lo reemplazó por una figura de tamaño $3x3$ que solo contiene el caracter `.`—así formó la *alfombra* 2:

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

Finalmente, la *alfombra* 3 se obtiene a partir de la *alfombra* 2, reemplazando nuevamente cada `#` por la figura base y cada `.` por una figura de $3x3$ hecha de puntos. La alfombra que tiene tu padrino es justamente la *alfombra* 3.

Viendo esto, se te ocurre una idea—¿y si le regalo un papel tapiz que use la misma técnica? Claro que no puede partir con la misma figura base, sería lo mismo que ya tiene, así que lo que vas a hacer es comenzar con una figura base cualquiera de nxn, y crear el "papel tapiz k".

Para esto, se hace el mismo procedimiento que la alfombra, pero usando esta nueva figura base. Es decir, la figura base es el *papel tapiz* 1, y el *papel tapiz* k se forma con el *papel tapiz* $k-1$, reemplazando cada `#` por la figura base, y cada `.` por una figura de nxn hecha de puntos.

Puedes referirte a los ejemplos para mayor claridad.

Entrada

La primera línea contiene los valores n y k ($2 \leq n \leq 4$, $1 \leq k \leq 5$) que indican que la figura tiene dimensiones nxn y la cantidad de iteraciones que vas a hacer para crear el papel tapiz.

Las siguientes n líneas son la figura. Cada línea tiene tamaño n y está formada únicamente por los caracteres `#` y `.`.

Salida

La salida es el *papel tapiz* k, escrito con caracteres `#` y `.`.

Ejemplos

Entrada 1	Salida 1
2 3 #. ##	#..... ##..... #.#.... ####... #...#.. ##..##. #.#.#.#. ########

Entrada 2	Salida 2
3 2 #.. #.# .#.	#..... #.#..... .#..... #.....#.. #.#...#.# .#.....#. ...#.... ...#.#... ....#....

Entrada 3	Salida 3
4 1 .### #### #### .#.#	.### #### #### .#.#

H. Bodegas

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

En el pueblo lineal de Distribuilandia, hay n personas que necesitan recibir suministros desde alguna de las k bodegas locales. Tanto las personas como las bodegas están ubicadas a lo largo de la única calle principal del pueblo, la cual puede modelarse como una línea recta (el eje X).

Cada bodega tiene una capacidad limitada: la bodega i puede atender a lo sumo c_i personas. Para evitar descontento entre los habitantes, el gobierno quiere que la persona más alejada de su bodega asignada esté lo más cerca posible.

Tu tarea es determinar si es posible asignar todas las personas a alguna bodega (respetando las capacidades), de forma tal que la distancia máxima entre una persona y su bodega asignada sea lo más pequeña posible. En otras palabras, debes encontrar la menor distancia d tal que existe una asignación válida en que toda persona está a distancia a lo sumo d de su bodega asignada.

Entrada

La primera línea contiene dos enteros n y k ($1 \leq n, k \leq 2 * 10^5$): el número de personas y el número de bodegas.

La segunda línea contiene n enteros $p_1, p_2, ..., p_n$ ($0 \leq p_i \leq 10^9$), las posiciones de las personas a lo largo de la calle.

La tercera línea contiene k enteros $b_1, b_2, ..., b_k$ ($0 \leq b_i \leq 10^9$), las posiciones de las bodegas. Finalmente, la cuarta línea contiene k enteros $c_1, c_2, ..., c_k$ ($1 \leq c_i \leq n$), la capacidad de cada bodega. Se garantiza que todas las posiciones (de personas y bodegas) son distintas.

Salida

Imprime un solo entero: la menor distancia máxima posible d tal que todas las personas pueden ser asignadas a alguna bodega a distancia a lo sumo d, y sin exceder la capacidad de ninguna bodega.

Ejemplos

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

Nota

La asignación optima se obtiene de la siguiente forma:

- Persona 1 a bodega 1: distancia 1
- Persona 2 a bodega 2: distancia 1
- Persona 3 a bodega 2: distancia 2
- Persona 4 a bodega 3: distancia 1

Ninguna otra asignación obtiene una distancia máxima menor.

I. Quesos Infectados

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

En el remoto y delicioso mundo de los quesos, una catastrófica plaga de hongos ha comenzado a expandirse.

Un solo hongo, con una voracidad insaciable, amenaza con consumir todos los valiosos bloques de queso.

Tu misión es estratégica: quitar la mínima cantidad posible de celdas con queso para evitar que el hongo pueda alcanzar al menos un bloque de queso.

La zona afectada se representa como una grilla de n filas y m columnas. Cada celda puede contener uno de los siguientes elementos:

- **Queso:** Representado por el carácter `Q`.
- **Hongo:** Representado por el carácter `H`. Existe exactamente una celda de este tipo.
- **Vacía:** Representado por el carácter `.`.

El hongo puede expandirse a celdas adyacentes en las cuatro direcciones (arriba, abajo, izquierda y derecha), pero únicamente a celdas que contienen queso. Cuando lo hace, consume ese queso y la celda pasa a estar ocupada por el hongo.

Tu objetivo es encontrar la **mínima cantidad de celdas de queso que deben ser quitadas** antes de que el hongo comience a expandirse, de manera que al menos una celda con queso quede inalcanzable para él.

Entrada

La primera línea de la entrada contiene dos enteros n y m ($1 \leq n, m \leq 1000$), que representan la cantidad de filas y columnas de la grilla, respectivamente.

Luego siguen n líneas, cada una con m caracteres, describiendo la grilla. Cada carácter será `Q` (queso), `H` (hongo) o `.` (vacío). Se garantiza que habrá exactamente una celda con `H`.

Salida

La salida debe contener un único entero: la mínima cantidad de celdas de queso que deben ser convertidas a celdas vacias para lograr el objetivo o -1 si es imposible lograr que quede un queso inalcanzable para el hongo.

Ejemplos

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

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

Entrada 3	Salida 3
3 3 .Q. QHQ .Q.	-1

J. Múltiplos de 3

Límite de tiempo: 0.5 segundos

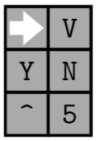
Límite de memoria: 256 megabytes

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 este tablero, 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 1: 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 $x = 34$:

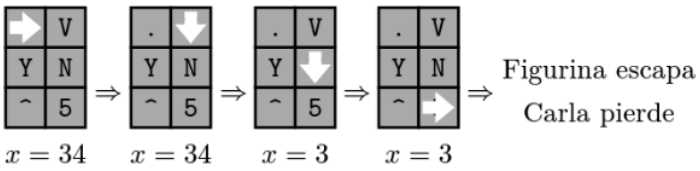


Figure 2: Configuración inicial de un tablero de ejemplo y un número x elegido por Carla. Notar la figurina sobre la celda superior izquierda.

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

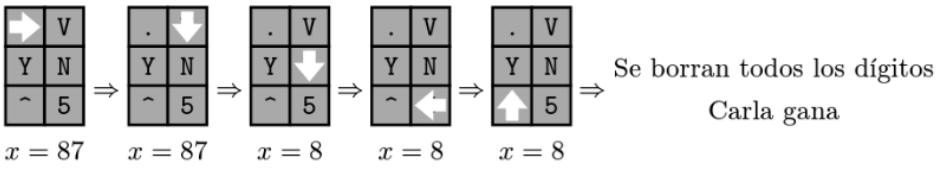


Figure 3: Configuración inicial de un tablero de ejemplo y un número x elegido por Carla. Notar la figurina sobre la celda superior izquierda.

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

Carla ha tenido mucha diversión con este juego, pero ya jugó todos los tableros que vienen en el manual y está aburrida, por lo que te pide diseñar un nuevo tablero tal que las elecciones ganadoras de x sean exactamente los múltiplos de 3. ¿Puedes hacerlo?

Entrada

Tu programa no recibirá entrada.

Salida

Tu programa debe imprimir un tablero tal que Carla gane si elige x múltiplo de 3, y pierda en caso contrario.

Formalmente, imprime n líneas ($1 \leq n \leq 100$), donde todas las líneas tengan m caracteres ($1 \leq m \leq 100$) y representen una fila del tablero cada una. Cada caracter debe ser uno de "`<>V^NY.0123456789`". Tu respuesta se considerará correcta si Carla gana sobre este tablero si y sólo si elige un múltiplo de 3 para el valor inicial de x.

Ejemplos

Entrada 1	Salida 1

	`.V`
	`YN`
	`^5`

Puntaje

Para verificar si tu solución es correcta, Carla jugará tu tablero con distintas elecciones de x. Decimos que tu tablero se *comporta bien* con una elección x múltiplo de 3 si y sólo si Carla gana al jugar tu tablero con este x. Por otro lado, para una elección x que no es múltiplo de 3, decimos que tu tablero se *comporta bien* si y sólo si Carla pierde al jugar tu tablero.

En la siguiente tabla, decimos que tu tablero *satisface* una fila si se comporta bien para todo el rango posible de valores de x. Recibirás el máximo puntaje de todas las filas que tu tablero satisface, o 0 puntos si no satisface ninguna fila.

Rango de x	Puntaje
$1 \leq x \leq 9$	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

En particular, recibirás puntaje completo sólo si tu tablero se comporta bien para todo x, sin importar su tamaño.

Nota

El tablero de ejemplo anterior no es correcto, es solo un ejemplo para mostrar el formato de la respuesta.

Puedes probar tu tablero con distintos valores de x en el visualizador oficial del problema.

Puedes basarte en el siguiente código de Python para imprimir el tablero que diseñes:

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