Problema A

Escape del Rectángulo

Tú estas actualmente en el punto (**x,y**) dentro de un rectángulo. La esquina inferior izquierda del rectángulo está en (0,0) y su esquina superior derecha está en (**w,h**). Retorna la menor distancia que tú debes viajar para llegar al borde del rectángulo.

Input

La primera línea del INPUT comienza con un número M indicando el número de casos (1<=M<=50), M líneas le siguen.

- Cada línea tendrá 4 enteros (**x y w h**) separados por un espacio.
- Cada entero estará entre 2 y 1000, inclusive.

Output

Deberá formatear su salida de la forma "Caso #N: R”.

(Comillas para claridad) donde N es el número de caso y R es el resultado.

Input	**Output**
4 1 1 5 5 6 2 10 3 653 375 1000 1000 161 181 762 375	Caso #1: 1 Caso #2: 1 Caso #3: 347 Caso #4: 161

Problema B

Número Espejo

Para este problema, un número es espejo si su representación decimal puede ser separada en dos números, de tal modo que el producto de todos los dígitos del primer número es igual al producto de todos los dígitos del segundo número. Por ejemplo, 1221 es un Número Espejo porque puede ser separado en 12 y 21, y 1*2=2*1. 1236 también es un *Número Espejo* pero 1234 no lo es. Note que Ud. Solo Puede separar un número en dos secuencias de dígitos consecutivos. Donde cada secuencia contiene al menos un digito. Así, Por ejemplo, podemos separar 12345 en cuatro diferentes formas: 1-2345, 12-345, 123-45, 1234-5. Sí el numero dado es *Numero Espejo* responda "YES" o "NO" de en caso contrario (comillas solo por claridad).

Input

La primera línea del INPUT comienza con un número M indicando el número de casos (1<=M<=100), M líneas le siguen.

- Cada línea tendrá un entero entre 1 y 2,147,483,647 inclusive.

Output

Deberá formatear su salida de la forma "Caso #N: R".

(Comillas para claridad) donde N es el número de caso y R es el resultado.

Input	**Output**
5 1 1221 1236 1234 808	Caso #1: NO Caso #2: YES Caso #3: YES Caso #4: NO Caso #5: NO

*Caso 1: Fíjese que un número de un solo digito no es Numero Espejo porque no puede ser separado en 2 números.

Problema C

Margen De Ganancia

El Margen está definido como el porcentaje de la ganancia de un ***item*** o un conjunto de ***items*** respecto a su precio de venta. Por ejemplo, si un elemento cuesta S/ 80 y es vendido a S/ 100, entonces hay una ganancia de S/ 20, o un margen de 20%.

Tú recibirás una cadena de ***items***, los cuales todos han sido vendidos en una sola transacción, cada ***item*** será formateado de la siguiente manera: “nnn.nn nnn.nn” (comillas por claridad), donde cada n es un digito entre 0 y 9 inclusive. Los ***items*** serán separados por una coma y un espacio “, “.

Cada ***item*** será de exactamente 13 caracteres, el primer número listado es el precio de venta y el segundo número es el precio de compra o costo.

Escribe un programa el cual calcule el porcentaje del margen de ganancia en la transacción y respóndelo como un entero, redondeado hacia abajo (el mayor entero que sea menor o igual que el valor actual)

Por ejemplo, digamos tú recibes los siguientes ***items***.

012.99 008.73, 099.99 050.00, 123.45 101.07

El costo total es 159.80. La venta total es 236.43. Eso significa que hubo 76.63 de ganancia. Esto resulta en un 32.41128% de margen, redondeado hacia abajo responde 32.

Input

La primera línea del INPUT comienza con un número M indicando el número de casos (1<=M<=100), M líneas le siguen.

- Cada línea tendrá entre 1 y 50 elementos de venta, inclusive.
- Los elementos están separados por una coma y un espacio “, “.
- El precio total de ventas de todos los ítems será mayor que el precio total de costos.
- El precio de venta total será mayor que 0.

Output

Deberá formatear su salida de la forma "Caso #N: R”.

(Comillas para claridad) donde N es el número de caso y R es el resultado.

Input	**Output**
3 012.99 008.73, 099.99 050.00, 123.45 101.07 000.00 049.99, 999.99 936.22, 033.99 025.64, 249.99 211.87 822.77 704.86, 829.42 355.45, 887.18 949.38	Caso #1: 32 Caso #2: 4 Caso #3: 20

Problema D

HuaH-ROT13

Existe un método de encriptado básico conocido como ROT13. Una propiedad de ROT13 es que el encriptado y des-encriptado son exactamente el mismo. Estos procesos funcionan por transformar simplemente una letra del alfabeto a otra. Las letras desde la A hasta la M se vuelven desde la N hasta la Z, tal que A->N, B->O,… M->Z. Las letras desde N hasta la Z se vuelven A hasta la M, tal que N->A, O->B,… Z->M.

Uno de los problemas con la mayoría de implementaciones es que solo cubre Mayúsculas, otro problema es que los números son ignorados completamente, dejándolos sin encriptado. Una manera de sobrellevar estas limitaciones es extendiendo el ROT13 a nuestra versión HuaH-ROT13 para cubrir minúsculas como también números. Así es como nuestro HuaH-ROT13 funcionará:

Caracteres	se vuelven
A-M	N-Z
N-Z	A-M
a-m	n-z
n-z	a-m
0-4	5-9
5-9	0-4

Por ejemplo, el mensaje “Uryyb 28” se volvería “Hello 73”:

U -> H	2 -> 7
r -> e	8 -> 3
y -> l	
y -> l	
b -> o	

Ten en cuenta que los espacios quedan como está.

Tu has interceptado un mensaje el cual tu crees está encriptado usando este proceso. Crea un programa que lea una cadena y retorna el mensaje codificado como una cadena.

Input

La primera línea del INPUT comienza con un número M indicando el número de casos (1<=M<=50), M líneas le siguen.

- Cada mensaje tendrá entre 0 y 50 caracteres (0-9, A-Z, a-z y espacios).

- El mensaje no tendrá dos o más espacios consecutivos.
- No habrán espacios adelante ni al final en el mensaje.

Output

Deberá formatear su salida de la forma "Caso #N: R”.

(Comillas para claridad) donde N es el número de caso y R es el resultado.

Input

```
6
Uryyb 28

5678901234
NnOoPpQqRr AaBbCcDdEe
Gvzr vf 54 71 CZ ba Whyl 4gu bs gur lrne 7558 NQ
Gur dhvpx oebja sbk whzcf bire n ynml qbt
```

Output

```
Caso #1: Hello 73
Caso #2:
Caso #3: 0123456789
Caso #4: AaBbCcDdEe NnOoPpQqRr
Caso #5: Time is 09 26 PM on July 9th of the year 2003 AD
Caso #6: The quick brown fox jumps over a lazy dog
```

*Caso 2: tenga cuidado que pueden venir lineas de 0 caracteres.

Problema E

RC

R y C están perdidos en un laberinto, C no se mueve y R es el que está en busca de C. Ayuda a R a encontrar a C en el menor número de pasos posibles, R puede moverse arriba, abajo, izquierda o derecha (si es que no hay un obstáculo). Retorne la cantidad mínima de pasos que le tomará a R llegar a C, y si es imposible retorne “FML” (comillas solo por claridad).

Input

La primera línea del INPUT comienza con un número M indicando el número de casos (1<=M<=50), M configuraciones de laberintos le siguen. La configuración es la siguiente

- La primera línea de cada caso esta compuesta por dos enteros “**H W**” que están entre 2 y 10, inclusive que representa el tamaño del laberinto, **H** es el Nro. de filas y **W** el Nro. de columnas. H líneas siguen:
  - Solo existirán los caracteres ‘R’ ‘C’ ‘#’ y ‘.’.
    - R es la ubicación inicial de R.
    - C es la ubicación de C y es donde R debe llegar.
    - # significa que es una pared (obstáculo) y que R no puede viajar por ahí.
    - Y ‘.’ Significa que es un espacio vacio.

Output

Deberá formatear su salida de la forma "Caso #N: R”.

(Comillas para claridad) donde N es el número de caso y R es el resultado.

Input

```
5
3 3
R..
.C.
...
3 3
R#.
#C.
...
3 3
R#C
.#.
...
6 5
R#...
.#.#.
.#.#.
.#.#.
...#C
.#.#.
8 7
R#.....
.#..#..
..#..#.
......#
#.#..#.
..#.#..
#....C#
.#.#...
```

Output

```
Caso #1: 2
Caso #2: FML
Caso #3: 6
Caso #4: 16
Caso #5: 11
```