Sai's Mini Project

Problem Description

Sai is working on a mini project where he needs to handle various tasks related to a bank account. This includes 6 operations viz., *read*, *credit*, *debit*, *abort*, *rollback* and *commit*. The operation semantics are explained below.

read - read and print the balance in the account

credit - money credited to the account

debit - money debited from the account

abort X - the Xth transaction from the beginning is aborted (cancelled). Note that you can't abort the transactions once they are committed. Only credit and debit operations are counted as transactions

For understanding abort, lets understand the example below -

credit 1000

commit

abort 1

This abort cannot be done since the transaction has already been committed.

rollback X - rollback the Xth commit. In this case, the account balance will be automatically updated to what it was, right after the Xth commit.

commit - the changes are saved permanently and cannot be undone using an abort command.

The *credit*, *debit*, *abort* and *rollback* operations will always be followed by a positive integer, whereas *read* and *commit* operations have no such operand.

Given the initial balance and a sequence of operations performed in order, determine the resulting output.

Constraints

1 <= initial amount <= 1000

1 <= Number of operations (N) <= 50

Current account balance will always be 0 or a positive value. It can never be negative.

The integer following "abort" and "rollback" will be at least 1 and will not exceed the total number of transactions and commits up to that point, respectively.

Input

The first line contains an integer denoting the initial balance of the account.

Second line consists of an integer *N* denoting the number of operations performed.

The subsequent N lines will contain operations following the above syntaxes.

Output

Output the account balance each time a read statement is encountered.

Time Limit (secs)

1

Examples

Example 1

Input

90

8

read

credit 10

debit 12

debit 8

credit 7

abort 1

commit

read

Output

90

77

Explanation

Initial balance: 90

Operation 1 - Read, prints the account balance i.e., 90

Operation 2 - Credit 10, Result: Balance increases by 10. New balance: 100

Operation 3 - Debit 12, Result: Balance decreases by 12. New balance: 88

Operation 4 - Debit 8, Result: Balance decreases by 8. New balance: 80

Operation 5 - Credit 7, Result: Balance increases by 7. New balance: 87

Operation 6 - Abort 1, this means undoing the changes made by the 1st transaction (the credit of 10). So, the account balance will become 77

Operation 7 - Commit, this commits all previous changes made. Account balance will be updated as 77

Operation 8 - Read, prints the account balance i.e., 77

Example 2

Input

43

10

credit 12

debit 10

commit

abort 1

read

credit 30

debit 4

rollback 1

commit

read

Output

45

45

Explanation

Initial balance: 43

Operation 1 - Credit 12, Result: Balance increases by 12. New balance: 55

Operation 2 - Debit 10, Result: Balance decreases by 10. New balance: 45

Operation 3 - Commit, this commits all changes made so far. Account balance will become 45.

Operation 4 - Abort 1, this means undoing the changes made by the 1st transaction. Since the changes of 1st transaction has already been committed it's impossible to abort it. Hence, the account balance remains same.

Operation 5 - Read, prints the account balance i.e., 45

Operation 6 - Credit 30, Result: Balance increases by 30. New balance: 75

Operation 7 - Debit 4, Result: Balance decreases by 4. New balance: 71

Operation 8 - Rollback 1, this means the account balance will be rolled back to what it was, after commit 1 i.e., 45. Refer Operation 3 which is the 1st commit.

Operation 9 - Commit, this commits all changes made since the last commit. Account balance will be updated as 45

Operation 10 - Read, prints the account balance i.e., 45

Frankenstein

Problem Description

"Muahahaha! No one can stop me from creating the elixir of life," -Frankenstein. Frankenstein is an alchemist who is always striving to craft the elixir of life.

The elixir of life grants immortality to whoever drinks it. In his attempts to brew the elixir of life, Frankenstein combines numerous ingredients and potions, believing he will eventually succeed. However, he has never succeeded thus far.

But each time he brews something, he meticulously notes the ingredients. For example, when he combined *snake fangs* and *wolfsbane*, he discovered the *awakening* potion. In his notes, he recorded the recipe as follows:

awakening = snakefangs + wolfbane

Similarly, when three ingredients were required, he noted the recipe in his notes as

thunderstorm = rain + wind + lightening

In general, the recipe would be

Potion 1 = Ingredient 1 + Ingredient 2

Potion 2 = Ingredient 1 + Ingredient 2 + Ingredient 3

Potion 3 = Ingredient 1 + Ingredient 2 + Ingredient 3 + Ingredient 4

and so on ...

Every brew requires magical orbs, which are mythical energy balls. The number of magical orbs needed for a recipe is equal to the number of ingredients minus one.

A recipe of a potion includes multiple ingredients. An ingredient can be an item or a potion. Items are readily available things and while potions are brewed from items. In his notes, the resultant is always a potion.

He observed that sometimes the same potion can be created using different recipes, with some requiring fewer magical orbs. Given a potion and his notes, determine the minimum number of magical orbs needed to create that potion.

Constraints

0 < N < 20

Input

First line contains an integer N representing the number of recipes he noted.

Next N lines contain string without space representing the recipes in his notes as mentioned above.

Last line contains a single string representing the potion that he needs to brew.

Output

Print single integer representing the minimum number of magical orbs required.

Time Limit (secs)

1

Examples

Example1

Input

4

awakening=snakefangs+wolfbane

veritaserum=snakefangs+awakening

dragontonic=snakefangs+velarin

dragontonic=awakening+veritaserum

dragontonic

Output

1

Explanation

Based on the input, we need to determine the minimum number of magical orbs required to brew *dragontonic*. According to the notes, there are two recipes available for brewing *dragontonic*.

The two ways of brewing *dragontonic* are, *dragontonic=awakening+veritaserum* and *dragontonic=snakefangs+velarin*

The recipe with *awakening*, *veritaserum* where *awakening* is a potion that must be brewed first. Brewing *awakening* requires 1 magical orb and brewing it with *veritaserum* requires one additional magical orb, totaling 2 magical orbs.

Since second recipe of *dragontonic* requires only 1 magical orb which is the minimum number of orbs required, hence print 1 as output.

Example 2

Input

6

oculus=thunderbrew+jellfish

felix=thunderbrew+pumpkin

wigenweld=thunderbrew+ladybug

wigenweld=oculus+felix

thunderbrew=pumpkin+firefly

maxima=pumpkin+ladybug

wigenweld

Output

```
2
```

Explanation

To brew *wigenweld* with the minimum number of orbs, he first brewed *thunderbrew*, which requires 1 orb. Then, brewing *thunderbrew* with ladybug required an additional orb, resulting in *wigenweld*. Therefore, a total of 2 orbs were needed, which is the minimum requirement, hence the output is 2.

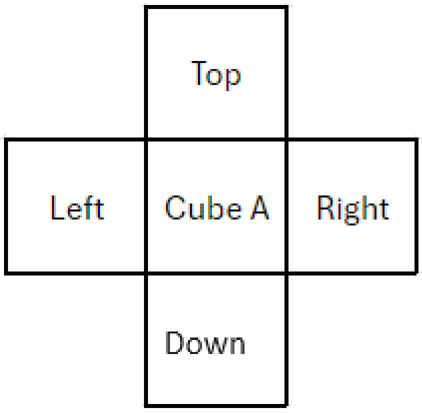
Common Cube Faces

Problem Description

Vedica got a challenging homework assignment from her teacher today, which involves arranging cubes on a desk.

There are initially Q+1 cubes, numbered from 1 to Q+1, and she will select the cubes in numerical order. She has been given Q queries, where each query is in the form "*cubeA cubeB direction*", indicating that *cubeB* is placed on the specified direction of *cubeA*. A side is considered common for two cubes if it is shared by both cubes, meaning they touch or are adjacent to each other along that entire side.

In 2D, a cube will be basically having four directions viz., top, down, left, right, as shown below.



The input queries are to be applied sequentially to form the whole structure. Only after all the queries are processed will the final structure emerge. Based on final structure, Vedica's task is to determine the number of common sides in the structure. Can you help Vedica with this?

One thing to keep in mind is that the input queries can overwrite each other. This simply means that the later query will overwrite the previous query in case they are dealing with same cubes and same direction. For example, if query is *1 3 left* - it means 3 is to the left of 1. If a later query is *1 10 left*, then it is easy to see that cube 10 has displaced cube 3 and now cube 10 is to the left of cube 1. These semantics must be factored in to compute the number of common sides.

Constraints

1 <= Q <= 50

No duplicate queries.

Input

First line consists of an integer *Q* denoting the number of queries.

The following Q lines each contain three space delimited elements, in format *cubeA cubeB direction*.

Output

Output a single integer representing the total number of common cube faces in the entire arrangement.

Time Limit (secs)

1

Examples

Example 1

Input

8

6 7 right

1 3 left

7 8 top

1 4 down

4 5 right

5 6 down

1 2 top

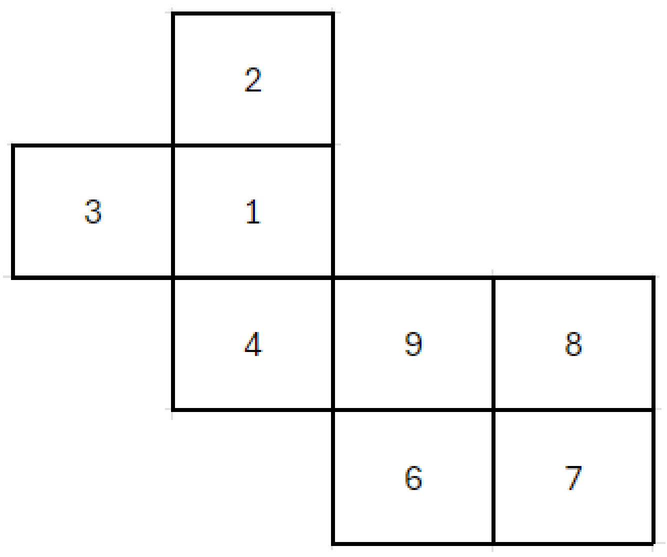
8 9 left

Output

8

Explanation

The above input when arranged looks like below.



From the image, we can find that the number of common cube faces is 8. Hence, output is 8.

Example 2

Input

3

1 2 right

3 4 left

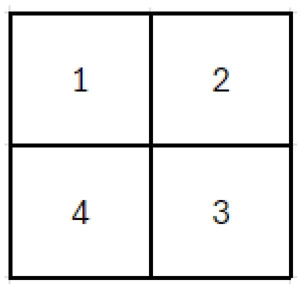
2 3 down

Output

4

Explanation

The above input when arranged looks like below.



From the image, we can find that the number of common cube faces is 4. Hence, output is 4.

Example 3

Input

5

3 4 top

1 2 right

4 5 down

1 3 top

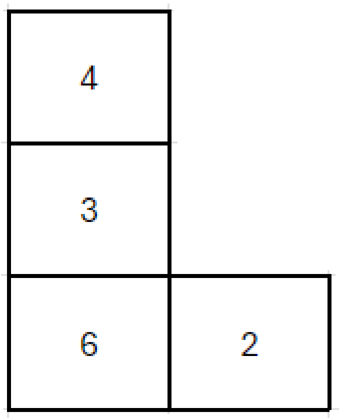
2 6 left

Output

3

Explanation

The above input when arranged looks like below.



From the image, we can find that the number of common cube faces is 3. Hence, output is 3.

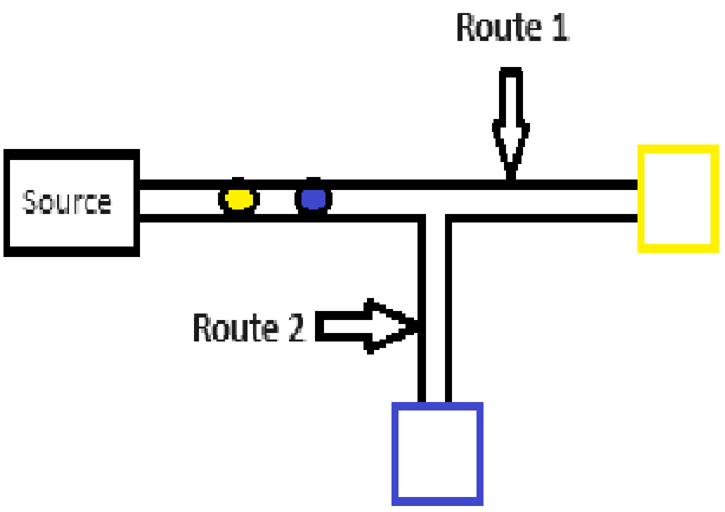
Route The Balls

Problem Description

Feeling bored at home, Lulu decided to play a mobile game called "Route The Balls". In the game, a source continuously releases different coloured balls, and the objective is to sort them into their corresponding-coloured buckets.

Between the source and the buckets, there are several junctions. The source, buckets are also considered as junctions. Initially, all the paths will be closed. At each junction, when multiple paths are present, only one path can be opened at a time. From a given junction, if a path is opened, any other paths that is currently open from the junction will be automatically closed.

For instance, if there exists a T-shaped junction, refer to the diagram below.



The initial ball is blue, and it needs to be directed towards the blue bucket positioned below. Initially, all sides of the paths are closed. Therefore, you open the downwards path (source - blue bucket) to allow the ball to flow into blue bucket. The subsequent ball is yellow, requiring it to be directed towards the yellow bucket situated to the right. To achieve this, you open the path on the right side (source - yellow bucket), automatically closing the previous downward opening. Hence, the junction was opened twice in total.

Given the network of junctions, their connections to each other, and the sequence of the balls, determine the total number of junction openings required to guide the balls into their respective buckets.

Constraints

1 <= number of balls <= 50

1 <= number of junctions <= 50

1 <= length of name of colours and junctions <= 20

There will always be a single source.

There will be only one bucket for each colour.

No balls will be used to route from the source if the corresponding-coloured bucket is not available.

Name of the junctions will consist of lower-case alphabets and digits.

Input

First line consists of *N,* denoting the number of lines representing the routes structure of the game.

In the following N lines, each line contains space-separated elements. The first element denotes a junction, while the subsequent elements indicate its connections to other junctions. Note that the paths are unidirected from source to buckets.

The last line comprises the colours of the balls originating from the source, separated by spaces.

Output

Print the total number of paths you need to open to route all the balls into respective buckets.

Time Limit (secs)

1

Examples

Example 1

Input

6

source jun1 jun3

jun1 jun2

jun2 jun4 jun6

jun4 blue

jun6 pink yellow

jun3 red

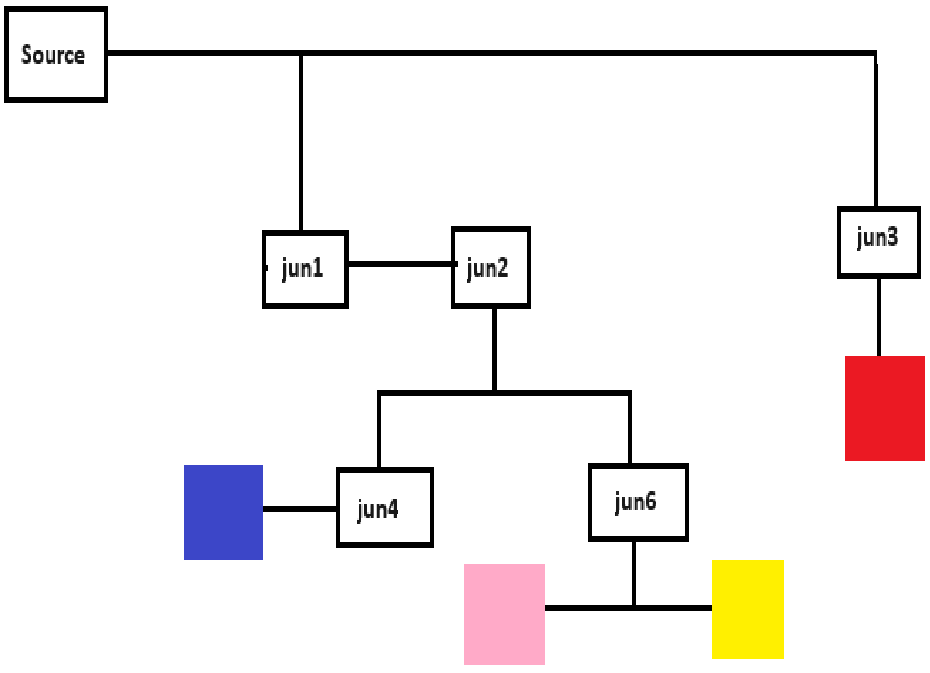
blue yellow red pink blue

Output

11

Explanation

The given input, when visualised looks below.



Initially, every path is closed.

First ball is blue coloured. For this, to land in blue bucket, open the paths, source - jun1, jun1 - jun2, jun2 - jun4, jun4 - blue. Total number of paths opened = 4

Next ball is yellow coloured. The paths from source - jun2 is already open. Open the paths jun2 - jun6 and jun6 - yellow. Total number of paths opened = 4 + 2 = 6

Next ball is red coloured. For this, open the paths source - jun3, jun3 - red. Note that the path which was open earlier from source (source - jun1) will be closed. Total number of paths opened = 6 + 2 = 8

Next ball is pink coloured. For this, open the paths source - jun1 and jun6 - pink. Note that the path which was open earlier from source (source - jun3) will be closed. The paths from jun1 - jun2, jun2 - jun6 are already open. Total number of paths opened = 8 + 2 = 10

The last ball is blue coloured. The paths source - jun1, jun1 - jun2, jun4 - blue is already open. Open the path jun2 - jun4. Total number of paths opened = 10 + 1 = 11

Hence, print 11.

Example 2

Input

4

source jun1

jun1 violet jun2 jun3

jun2 red

jun3 green yellow

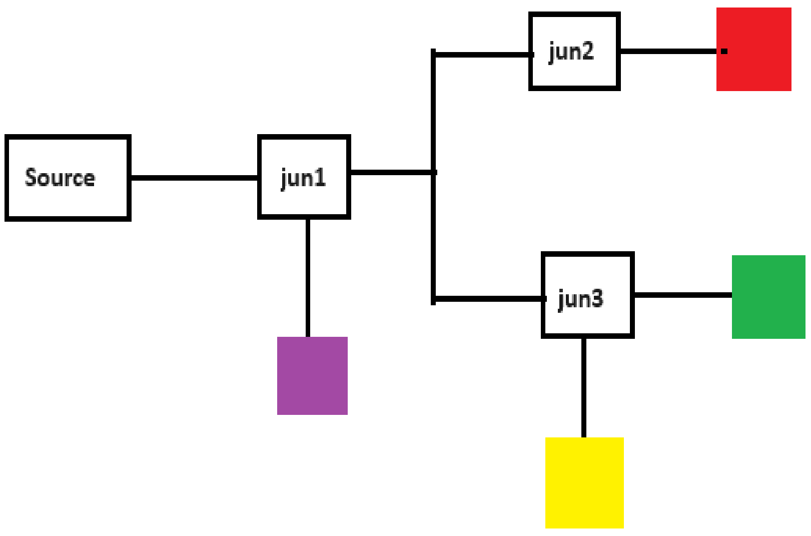
green yellow red green green violet

Output

9

Explanation

The given input, when visualised looks below.



Initially, every path is closed.

The first ball is green coloured. Open the paths, source - jun1, jun1 - jun3, jun3 - green. Total number of paths opened = 3

Next ball is yellow coloured. The paths source - jun1, jun1 - jun3 are already open. Hence open the path jun3 - yellow. jun3 - green will be closed. Total number of paths opened = 3 + 1 = 4

Next ball is red coloured. Open the paths jun1 - jun2 and jun2 - red. Other paths which are opened from the same junctions will be closed. Total number of paths opened = 4 + 2 = 6

Next ball is green coloured. Open the paths jun1 - jun3 and jun3 - green. Total number of paths opened = 6 + 2 = 8

Next ball is also green, and the respective paths are opened in the above steps. Hence the total paths opened remains same as of above step.

Next ball is violet. Open the path jun1 - violet. Total number of paths opened = 8 + 1 = 9

Hence, print 9.

Magic Stars Intensity

Problem Description

In the 1930s, King Krishnadevaraya, who had a great love for magic, kept a personal magician in his palace. Whenever he desired to witness a magical performance, he would command the magician to entertain him with his craft.

The magician constantly aimed to impress the king with new magical tricks. One day, he cast magical lines across the vast expanse of the palace floor, which was covered in tiles. Each tile is a square with sides of 1 unit length; thus, you can say the palace floor resembles a 2d plane.

Since these are magical lines, when they are drawn, they only align with the edges of the tiles or pass through their corners.

When these magical lines intersect, they create points of light called n-stars, where n ranges from 2 to 8. Each n-star forms when n lines intersect, and all these stars generate light.

For calculating the intensity of the star, there exist two cases which are explained below.

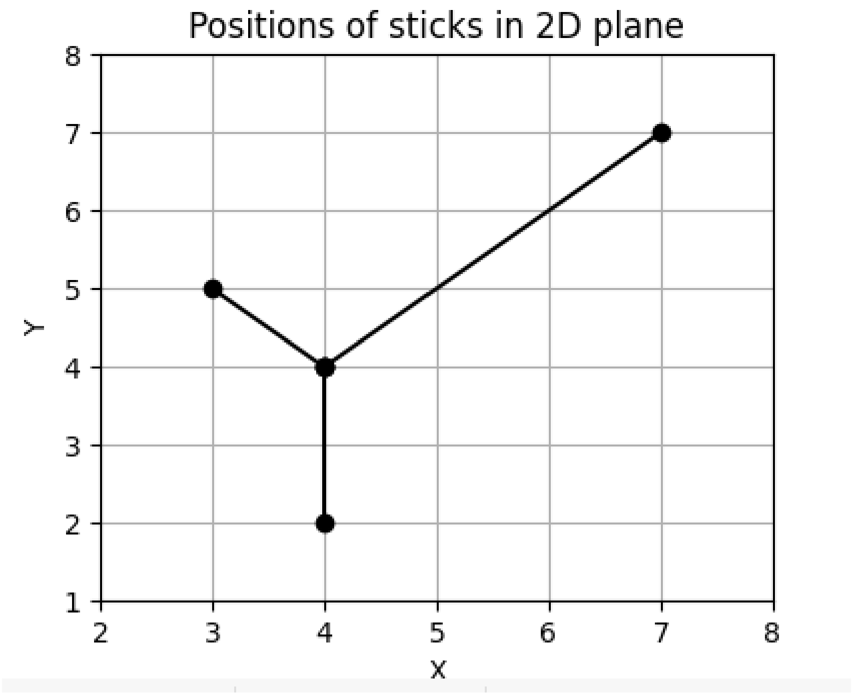
Consider below cases carefully.

Case 1 - The line is only one side to the star i.e., the star won't cut the line into two parts.

Consider the lines (4, 4, 4, 2), (4, 4, 7, 7) and (4, 4, 3, 5). These lines are intersecting at the point (4, 4). Since three lines intersect at a point, they form a star known as a 3-star.

Now, the intensity of the star = minimum (the number of cells these 3 lines are touching from the point of star formation to the last) = minimum (2, 3, 1) = 1

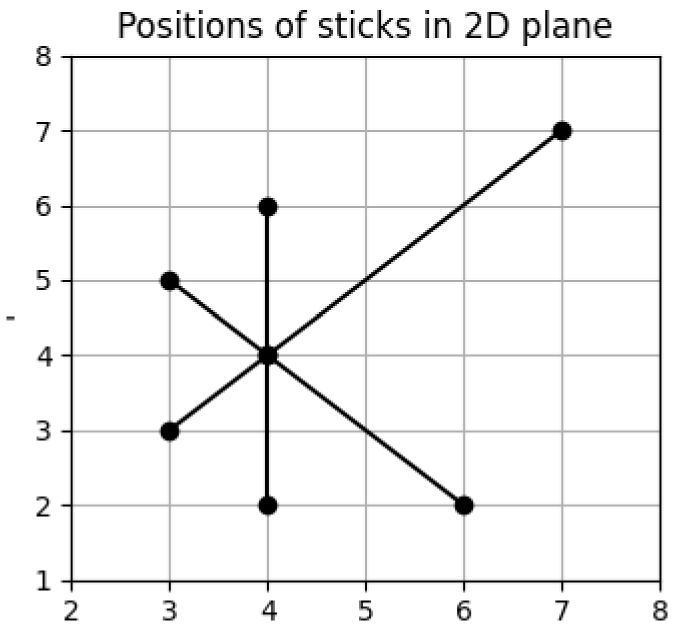
So, the intensity of this star will be 1.



Case 2 - The line is two sides to the star i.e., the star cuts the line into two parts.

Consider the lines (3, 3, 7, 7), (3, 5, 6,2) and (4, 2, 4, 6). These lines are intersecting at the point (4, 4). Since three lines intersect at a point, they form a star known as a 3-star.

In this case, the intensity of the star = minimum (the number of cells these 3 lines are touching from the point of star formation to the last on both sides) = minimum (1, 1, 2, 2, 3, 2) = 1



Given N lines and the type of star for which you need to determine the intensity, calculate the intensity for all such stars according to the cases described and print their total sum. If no stars of the specified type are present, print 0.

Constraints

1 <= N <= 50

2 <= K <= 8

0 <= x, y <= 100

Lines will not overlap either partially or completely.

Input

First line consists of an integer *N,* denoting the number of magical lines the magician casted.

The next N lines contain four space-separated integers each, representing the x and y coordinates of the starting and ending points of the magical lines.

The last line consists of an integer *K* denoting the type of star for which you need to calculate the intensity.

Output

Print a single integer representing the total intensity of all stars of the specified type given in the input. If no such stars are present, print 0.

Time Limit (secs)

1

Examples

Example 1

Input

7

4 2 4 6

6 5 6 7

1 3 3 5

3 5 4 4

3 3 7 7

2 2 2 5

4 4 5 3

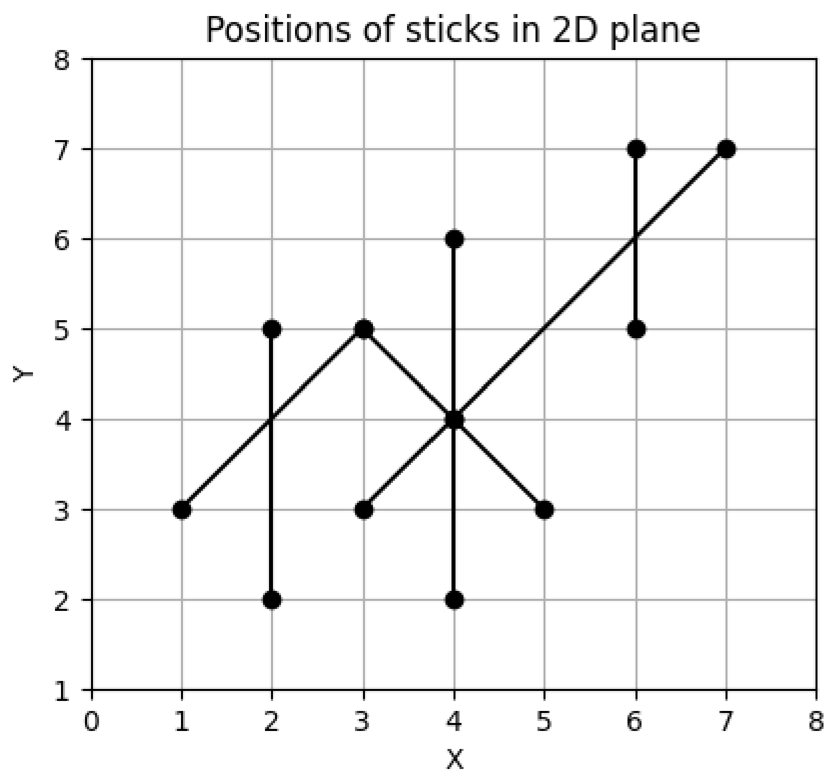
4

Output

1

Explanation

The lines given in the above input are represented in the below figure.



Here, the star formed at (4, 4) is a 4-star because it is created by 4 lines. According to given cases, the intensity of this star is minimum (2, 1, 1, 2, 1, 3), resulting in an intensity of 1. Since there is only one 4-star, the output is 1.

Example 2

Input

5

1 1 8 8

3 1 3 4

5 1 5 8

1 3 3 3

7 2 7 9

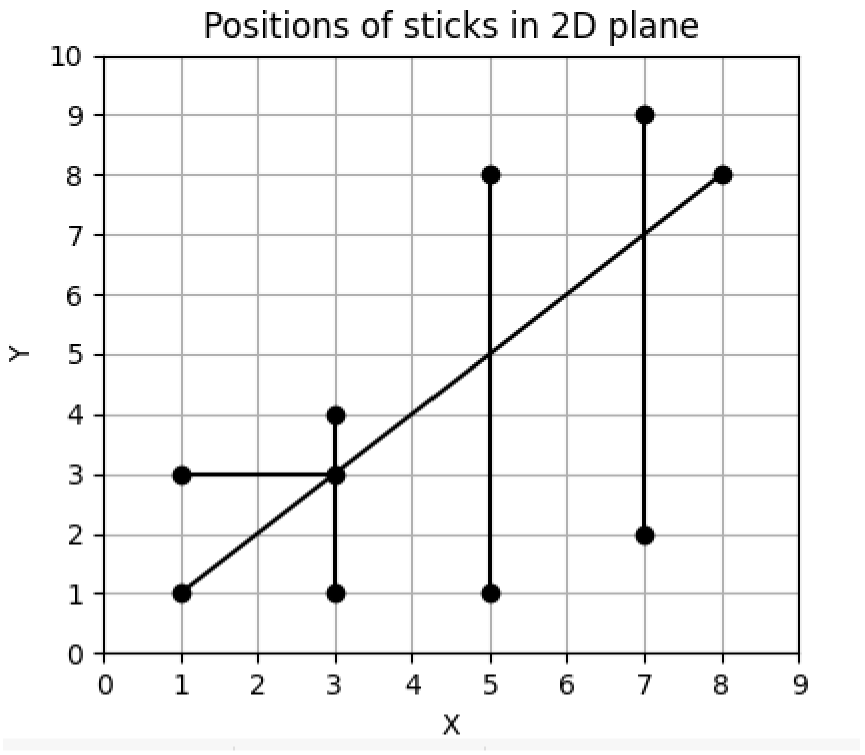
2

Output

4

Explanation

The lines given in the above input are represented in the below figure.



There are two 2-stars formed at the positions (5, 5) and (7, 7). The intensity of star at (5, 5) is minimum (4, 3, 3, 4) which is 3 and the intensity of star at (7, 7) is minimum (1, 2, 6, 5) which is 1. So, the total intensity of all 2-stars will be 3+1 = 4.

ReenuCircuit

Problem Description

Reenu is an electrical engineer who frequently designs new circuits. She constructs circuits using two types of resistors: horizontal and vertical, along with a junction. In each circuit, there are two positions: an opening and a closing. The circuit is connected to the power supply at the opening position, and the current flows through the resistors and junctions until it reaches the closing position.

In the matrix representation of the circuit, power can only flow vertically (up or down) if a vertical resistor is present, and it can only flow horizontally (left or right) if a horizontal resistor is present. The junction connects all four sides. The vertical and horizontal resistors have 1 units of resistance, while the junction considered to be having low resistance, ignorable.

Vertical resistor is represented by "|" (pipe), horizontal represented by "-"(hyphen) .a "." (period) symbol represents terminals and "+" (plus) representing junction.

The resistance for series connection is RT = R1 + R2 and for parallel connection is 1/RT = 1/R1 + 1/R2.To know about resistance in series and parallel look here.

Given the circuit, reduce and determine the equal resistance of the circuit between opening position and closing position.

To know more about reduction of resistor in series and parallel look here

Constraints

3 <= N <= 10

Input

First line consists of *N,* denoting the number of rows, columns in the matrix.

The following N lines represent the circuit as a matrix.

Output

Print the total resistance of the circuit.

Time Limit (secs)

1

Examples

Example 1

Input

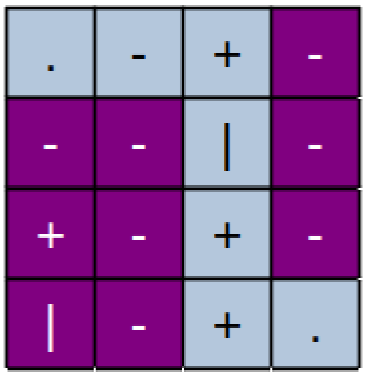
```
4
.-+-
--|-
+-+-
|-+.
```

Output

```
2
```

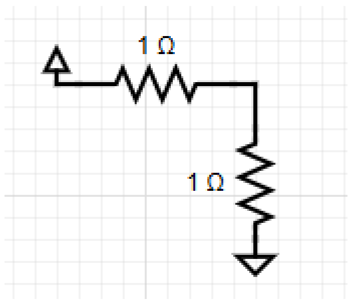
Explanation

The above input is visualized below.



.	-	+	-
-	-	|	-
+	-	+	-
|	-	+	.

The Equivalent circuit of this will be,



Two resistors in series and adding them results in 2 units of resistors which is the equivalent.

Example 2

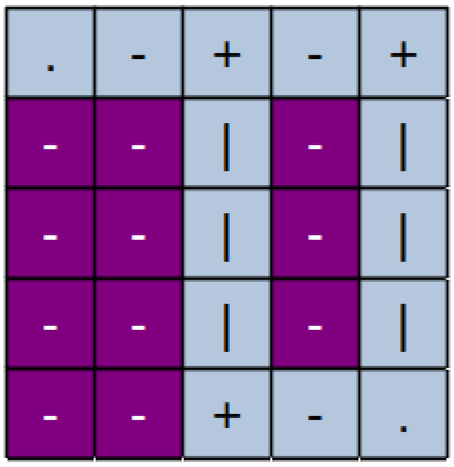
Input

```
5
.-+-+
--|-|
--|-|
--|-|
--+-.
```

Output

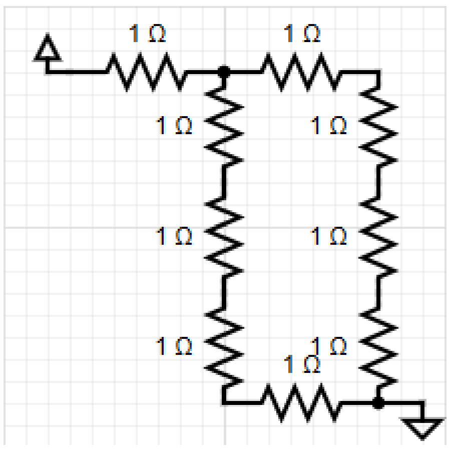
3

Explanation



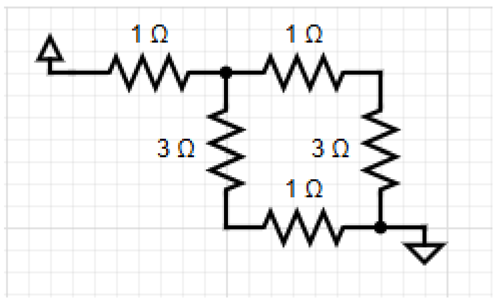
.	-	+	-	+
-	-	|	-	|
-	-	|	-	|
-	-	|	-	|
-	-	+	-	.

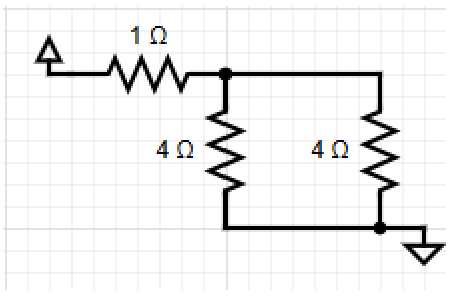
The equivalent circuits look as below.

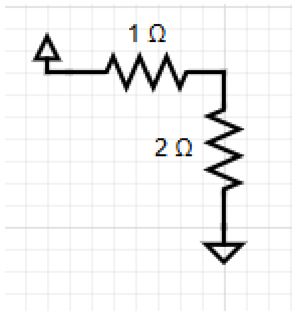


The reduction goes as below...

First the three resistors series on both the branches will combined together.Followed by 3 and 1 unit resistors on both the branches will combined together.Forming 4 unit of resitors in parallel.The 4 units of resistors are combined and become a single 2 units of resistors







The resultant 2 units of resistor is series with 1 unit of resistor equivalent to 3 units of resistors.And thus the equivalent resistor 3.