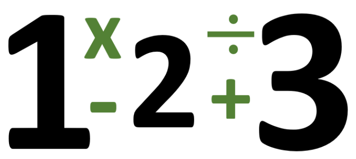
Problem A
Number Fun

Ms. Greene is trying to design a game for her third-grade class to practice their addition, subtraction, multiplication, and division. She would like for every student in her class to be able to "think mathematically" and determine if any two given numbers can be added, subtracted, multiplied, or divided in any way to produce a third given number. However, she would like to be able to check her students' work quickly without having to think about it herself.



Source: WikiMedia Commons

Help Ms. Greene by writing a program that inputs two given numbers and determines whether or not it is possible to add, subtract, multiply, or divide those two numbers in any order to produce the third number. Only one operation may be used to produce the third number.

Input

Each input file will start with a single integer N ($1 \leq N \leq 10000$) denoting the number of test cases. The following N lines will contain sets of 3 numbers a, b, c ($1 \leq a, b, c \leq 10000$).

Output

For each test case, determine whether or not it is possible to produce the third number, c, using the first two numbers, a and b, using addition, subtraction, multiplication, or division.

Sample Input 1

```
6
1 2 3
2 24 12
5 3 1
9 16 7
7 2 14
12 4 2
```

Sample Output 1

```
Possible
Possible
Impossible
Possible
Possible
Impossible
```

This page is intentionally left blank.

Problem B
Linden Mayor System

```
A B A A B   A B A
 \| | |/     |/ /
  A B A      A B
   \| |      |/
    A B      A
     \|     /
      A   B
       \ /
        A
```

Aristid is the mayor of a town called Linden. He and the townsfolk love fractals. One day, Aristid decides to genetically alter trees so that they have mathematically pleasing structures. It turns out that the people of Linden will support this idea only if the trees are sufficiently "tree-like." So Aristid came up with the following system to generate realistic looking trees. Since he's a little vain, he decided to call it the Linden Mayor System.

Start with a sequence of letters S_0. This is the seed that will be used to generate the rest of the tree. Next define some rules to model the branching behavior of the tree. A rule will have the form $x \to y$, indicating that the letter x will be replaced with the sequence y. By applying these rules to S_0, the new sequence S_1 is created. These rules can be applied over and over to produce new sequences. In general, to create S_{n+1} from S_n, replace all the letters in sequence S_n according to the rules. Some letters may not have a rule associated with them. Such *terminal* letters are not replaced.

As an example, consider the starting sequence A with rules: A $\to$ AB and B $\to$ A. The first four iterations are as follows:

S_0:	A	Starting sequence.
S_1:	AB	A is replaced with AB by rule A $\to$ AB. Note that rule B $\to$ A couldn't be applied.
S_2:	ABA	Again, A is replaced by AB but B is replaced with A (rule B $\to$ A).
S_3:	ABAAB	Keep applying rule A $\to$ AB for A's and rule B $\to$ A for B's...
S_4:	ABAABABA	This is the resulting sequence after four iterations.

Input

The first line will contain two positive integers: $0 \leq n \leq 26$ and $0 \leq m \leq 5$. Following this will be n lines defining the rules for a Linden Mayor System. Each line is of the form: x -> y, indicating that x is replaced by y. x and y will contain only uppercase letters from A to Z, and the length of y is guaranteed to be at most five. The last line will contain the starting sequence S_0 which will be no longer than 30 characters and will contain only uppercase letters from A to Z.

Output

Output the resulting sequence S_m which is produced after m iterations.

Sample Input 1

```
2 4
A -> AB
B -> A
A
```

Sample Output 1

```
ABAABABA
```

Problem C
Turtle Master

Robot Turtles is one of Theta's favorite games. In this game, kindergarteners learn how to "code" by creating programs that move a turtle from a starting field to a diamond. Robot Turtles is reportedly the most successful game funded by the Kickstarter incubator.

In the actual board game, an adult plays the role of the "turtle master," which is the person that plays the role of a CPU to execute the program. As you can imagine, this is a pretty boring task that cries out for automation: in this problem, you are asked to write a program that automates the task of the turtle master.

Robot Turtles is played on an 8×8 board. There is one turtle (marked with the letter `T`), which always starts out at the bottom-left field, facing right. The board contains empty squares (marked as `.`), castles made out of rock (`C`), and castles made out of ice (`I`). The diamond is marked with a `D`. The turtle may move only onto empty squares and the square on which the diamond is located.

A turtle program contains 4 kinds of instructions, marked by a single letter.

- `F` The turtle moves one field forward in the direction it is facing. If the turtle faces a castle or the border of the board, a program error occurs.
- `R` The turtle turns 90 degrees to the right (the turtle will just turn and stay on the same field).
- `L` The turtle turns 90 degrees to the left (the turtle will just turn and stay on the same field).
- `X` The turtle fires a laser in the direction it is facing. If the square it is facing contains an ice castle, the ice castle will melt and the square will turn into an empty square. Otherwise, a program error occurs. The turtle will not move or change direction. It is a program error to fire the laser at empty squares, rock castles or outside the board.

Input

The input consists of 9 lines. The first 8 lines contains the board, each line representing the squares in a row of the board. The turtle will always be at the bottom-left. There will be exactly 1 diamond. The 9th line contains the program the turtle master is asked to execute.

Output

Output `Diamond!` if the entire program can be executed without program error and if the turtle is on the diamond square after the program has been executed. Output `Bug!` if a program error occurred, or if the turtle does not end up on the diamond square!

Sample Input 1

```
........
........
........
...CC...
..C.DC..
.C..C...
C.IC....
T.C.....
FLFRXFLFRFLFRF
```

Sample Output 1

```
Diamond!
```

Sample Input 2

```
........
........
........
...CC...
..C.DC..
.C..C...
C.IC....
T.C.....
FLFRFLFRFLFRF
```

Sample Output 2

```
Bug!
```

Sample Input 3

```
........
........
........
...CC...
..C.DC..
.C..C...
C.IC....
T.C.....
FLFRXFLFRFLFFR
```

Sample Output 3

```
Bug!
```

Problem D
Cracking RSA

Image by Sean MacEntee (Flickr) [CC BY 2.0]

RSA is a widely used public-key cryptosystem that allows two parties (such as people or computers) to exchange secret messages without revealing information to anyone else listening on the conversation. Many websites use RSA when visited over a secure `https` connection. In RSA, each party has two different keys: a *public* key that is published and a *private* key that is kept secret. To encrypt a message intended for a specific recipient, the sender will use the recipient's public key to encrypt the message. The recipient will use their private key to decrypt the message.

An RSA public key (n, e) consists of two numbers n and e. The number n is a product of two distinct prime numbers, p and q. In real applications, n would be hundreds of decimal digits long for security.

Let $\varphi(n)$ be Euler's *totient* function, which in this case is equal to $(p-1)(q-1)$. The private key consists of (n, d), where n is the same as in the public key and d is the solution to the congruence

$$de \equiv 1 \bmod \varphi(n)$$

Formally, a congruence

$$a \equiv b \bmod c$$

holds for three integers a, b, and c, if there exists an integer k such that $a - b = kc$.

The sender will encrypt a message M (which, for simplicity, is assumed to be an integer smaller than both p and q) by computing $M^e \bmod n$ and sending it to the receiver. The recipient will calculate $(M^e)^d \equiv M^{ed} \equiv M^{k\varphi(n)+1} \equiv M^{\varphi(n)k}M \equiv M \bmod n$ since by Euler's theorem $M^{\varphi(n)} \equiv 1 \bmod n$. This will reconstruct the original message. Without the private key, no practical way has been found for a potential attacker to recover M from the knowledge of $M^e \bmod n$ and (n, e).

Your task is to crack RSA by finding the private key related to a specific public key.

Input

The first line of input has the number of test cases T, $(1 \le T \le 50)$. Each test case has one line that contains the two numbers n and e. You may assume that n is the product of two primes p, q such that $2 \le p, q < 1000$. Also, e will be chosen so that d exists and is unique, and $1 < d, e < \varphi(n)$. Note that the product de may not fit into a 32-bit integer (e.g. Java's `int` type).

Output

For each test case, output the single number d.

Sample Input 1	Sample Output 1

2
33 3
65 11
	7
	35

Problem E
Cowboy Checkers

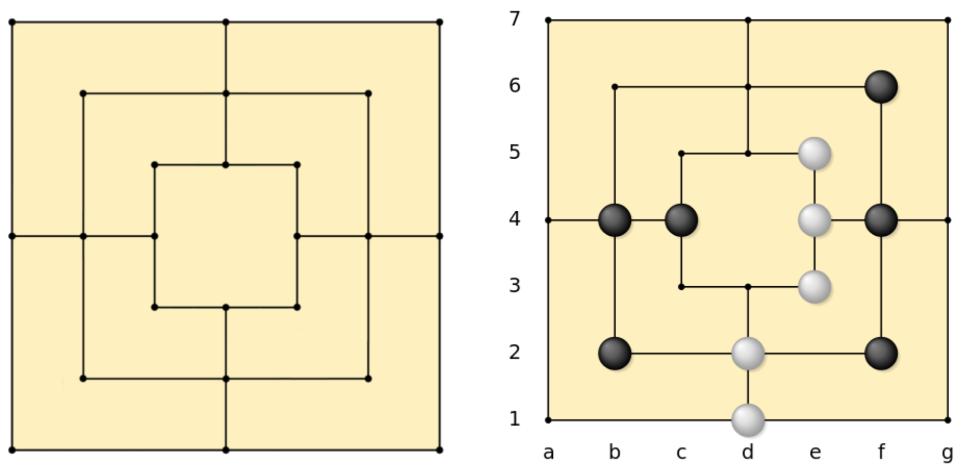


Figure E.1: Left: the layout of the board at the beginning of the game. Right: a position that results in a win for White. Source: Wikimedia Commons

Theta likes to play a board game called Cowboy Checkers, also known under the name “Nine Men’s Morris.” This strategy board game, which dates at least to the Roman Empire, is played by two players on a grid with 24 intersections, or points, as shown on the left in Figure E.1. Each player has nine pieces represented by white and black colors. During the initial phase of the game, players take turns placing pieces on the board at unoccupied intersections. Once all pieces have been placed, players take turns moving pieces. Pieces can be moved along the lines of the board from one intersection (or corner) to any connected open intersection (or corner).

The goal of the game is to take one’s opponent’s pieces, which is accomplished by forming “mills,” which consist of three pieces that are either horizontally or vertically adjacent. For instance, Figure E.1 shows two mills: white has a mill consisting of the pieces on `e3`, `e4`, and `e5`, and black has a mill consisting of the pieces on `f2`, `f4`, and `f6`.

Each time a player forms a mill they can take one piece from their opponent. Thus, a desirable position, which almost always results in a win, is a “double mill”: in this position, the player whose turn it is can move one piece from one mill such that it closes (completes) another mill. In the next move, the player can move back, closing the first mill again and reopening the second. For instance, in Figure E.1, White has created a double mill: the move `e3` to `d3` would close a mill consisting of `d1`, `d2`, and `d3` while simultaneously creating an open mill `e4`, `e5`.

Theta is pretty good at spotting double mills - can you write a program that does the same?

Input

The input consists of a single test case. This test case consists of 7 lines with 7 characters each. The letters `B` and `W` denote black and white pieces, respectively. The character `.` denotes an open corner or intersection, except in the center of the board, where it has no significance. The characters `|` and `-` are used to denote connector lines that may not contain any pieces. All inputs represents valid positions as far as which points may contain pieces, as displayed on the left in Figure E.1. There may be up to nine black and up to nine white pieces on the board.

Output

If White has at least one double mill, output `double mill`, else output `no double mill`!

Sample Input 1

```
.--.--.
|.-.-B|
||..W||
.BB.WB.
||..W||
|B-W-B|
.--W--.
```

Sample Output 1

```
double mill
```

Sample Input 2

```
.--.--.
|.-.-.|
||...||
.......
||...||
|.-.-.|
.--.--.
```

Sample Output 2

```
no double mill
```

Sample Input 3

```
B--.--.
|W-.-.|
||BBB||
......B
||WWW||
|W-.-W|
.--B--B
```

Sample Output 3

```
double mill
```

Sample Input 4

```
B--B--.
|W-.-B|
||BBB||
.W....B
||WWW||
|W-W-W|
.--B--B
```

Sample Output 4

```
no double mill
```

Problem F
Cookie Cutters

Theta likes to bake cookies for the upcoming holidays. She has a selection of cookie cutters of different shapes and sizes. She thinks it's unfair that some shapes are larger than others - because even if everyone gets the same number of cookies, some will end up with more dough!

She comes up with a plan to enlarge (or shrink) the cookie cutters' shapes — while retaining their proportions — so that all cookies will have the same weight.

Your task in this problem is to help her draw the plans the new cookie cutters will be based on!

Input

The input is a single test case, consisting of one cookie cutter outline. Cookie cutters are modeled as simple polygons. (Simple polygons do not self-intersect.) The first line of input contains a number N ($3 \leq N \leq 50$) denoting the number of corners of the polygon. This is followed by N lines containing two floating point numbers X and Y ($-500 \leq X, Y \leq 500$), each describing a set of coordinates of a polygon point. The coordinates are given in counter-clockwise order.

The next line will contain an integer number A ($0 < A \leq 10000000$) denoting the size of the desired area to which the polygon should be grown or shrunk. (Since the dough is rolled out precisely to an even height everywhere, we can use area to represent weight.) The resized polygon must be similar to the original polygon, that is, all interior angles must be congruent and all corresponding sides must have the same ratio.

Output

Output N lines with the X, Y coordinates of the expanded/shrunk polygon. Because these coordinates will be used to draw a new cookie cutter, your new polygon should lie in the north-east quadrant ($x, y \geq 0$) and it should touch the x- and y-axes in at least one point. More precisely, you must move the resized polygon horizontally and/or vertically so that $\min x_i = \min y_j = 0$. You may not, however, rotate or skew the polygon in any way.

Provide your answers as floating point numbers. Your answer will be considered correct if none of the x, y coordinates have an absolute or relative error larger than 10^{-4}.

Sample Input 1	Sample Output 1

3
0 0
1 1
0 2
4
	0.0 0.0
	2.0 2.0
	0.0 4.0

Sample Input 2

```
12
-1.5 2
0 1.5
0 0
1 .5
2 0
2 1.5
3.5 2
2 2.5
2 4
1 3.5
0 4
0 2.5
20
```

Sample Output 2

```
0.0 3.06785995539
2.30089496654 2.30089496654
2.30089496654 0.0
3.83482494424 0.766964988847
5.36875492193 0.0
5.36875492193 2.30089496654
7.66964988847 3.06785995539
5.36875492193 3.83482494424
5.36875492193 6.13571991078
3.83482494424 5.36875492193
2.30089496654 6.13571991078
2.30089496654 3.83482494424
```

Problem G
Railroad

Theta likes to play with her DUPLO railway set. The railway set she has consists of pieces of straight tracks, curved tracks, Y-shaped switches, and X-shaped level junctions, as well as bridges that allow one track to cross over another. There are also straight tracks that are railroad crossings to allow car traffic to cross.

Close-ups of some of the pieces are shown below:

To play, she picks a number of X-shaped level junctions and a number of Y-shaped switches and connects them with straight and curved pieces, using bridges as necessary.

Because the set doesn't include any bumpers, she wants to build a closed track, like all the examples shown in the manual that came with the set:

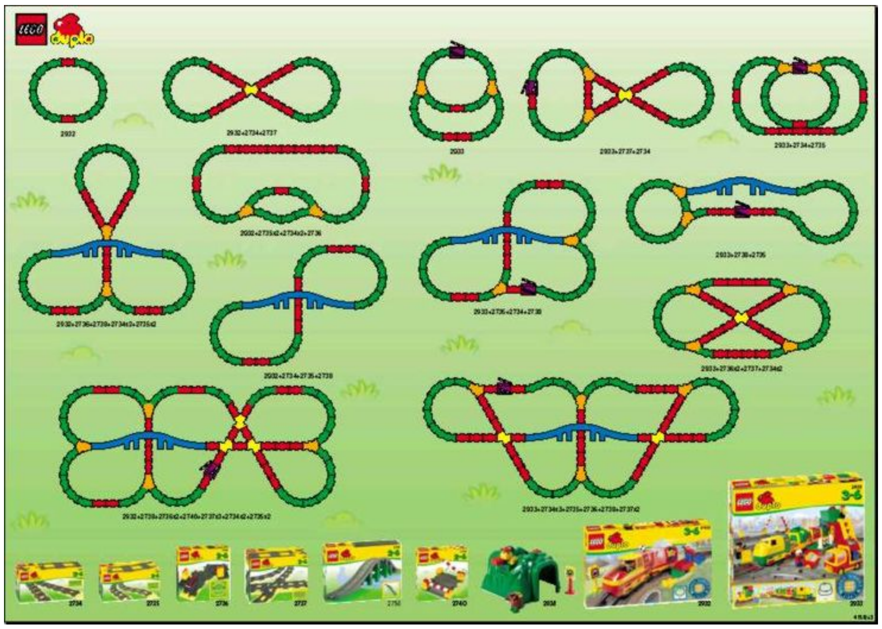


Figure G.1: Various track layouts possible with the DUPLO railway system. Curved pieces are shown in green, straights in red, switches in orange, level junctions in yellow, bridges in blue, and crossings in black. DUPLO is a trademark of the LEGO Group.

Unfortunately, sometimes, this doesn't seem to work with the number of X-shaped level junctions and Y-shaped switches she starts out with.

She quickly figures out exactly when it is possible to build a closed track - can you figure it out, too?

Write a program that outputs if it is possible to build a railroad that does not require any bumpers (i.e., which does not have any dead-end tracks).

Input

The input consists of a single test case with two integer numbers X ($0 \leq X \leq 1000$) and Y ($0 \leq Y \leq 1000$) denoting the number of level junctions and switches, respectively. You may assume that Theta has sufficiently many straight and curved pieces as well as bridges.

Output

Output `possible` if she can build a closed track using all level junctions and all switches without any dead ends, or `impossible` otherwise.

Sample Input 1

```
1 0
```

Sample Output 1

```
possible
```

Sample Input 2

```
0 2
```

Sample Output 2

```
possible
```

Sample Input 3

```
1 3
```

Sample Output 3

```
impossible
```

Problem H
Robot Turtles

Robot Turtles is one of Theta's favorite games. In this game, kindergarteners learn how to "code" by creating programs that move a turtle from a starting field to a diamond. Robot Turtles is reportedly the most successful game funded by the Kickstarter incubator.

Players must develop a program consisting of "instructions" that brings a turtle from a starting location to a goal (a diamond). An adult will then "execute" this program by moving the turtle based on the given instructions.

Robot Turtles is played on an 8×8 board. There is one turtle (marked with the letter `T`), which always starts out at the bottom-left field, facing right. The board contains empty squares (marked as `.`), castles made out of rock (`C`), and castles made out of ice (`I`). The diamond is marked with a `D`. The turtle may move only onto empty squares and the square on which the diamond is located.

A turtle program contains 4 kinds of instructions, marked by a single letter.

- `F` The turtle moves one field forward in the direction it is facing. If the turtle faces a castle or the border of the board, a program error occurs.
- `R` The turtle turns 90 degrees to the right (the turtle will just turn and stay on the same field).
- `L` The turtle turns 90 degrees to the left (the turtle will just turn and stay on the same field).
- `X` The turtle fires a laser in the direction it is facing. If the square it is facing contains an ice castle, the ice castle will melt and the square will turn into an empty square. Otherwise, a program error occurs. The turtle will not move or change direction. It is a program error to fire the laser at empty squares, rock castles or outside the board.

Input

The input consists of 8 lines, which represents the board, with each line representing one row. The turtle will always start out at the bottom-left. There will be exactly 1 diamond. There will be no more than 10 ice castles.

Output

Output the shortest valid turtle program whose execution (without program error) brings the turtle from the starting location to the diamond! If there are multiple such programs of equal length, you may output any of them!

Output `no solution` if it is not possible for the turtle to reach the diamond!

Sample Input 1

```
........
........
........
...CC...
..C.DC..
.C..C...
C.IC....
T.C.....
```

Sample Output 1

```
FLFRXFLFRFLFRF
```

Sample Input 2

```
........
........
........
...CC...
..CIDC..
.CI.C...
C.IC....
T.C.....
```

Sample Output 2

```
FLFRXFLXFRFLXFRF
```

Sample Input 3

```
........
........
........
...CCD..
..C..C..
.C..I...
C.IC....
T.C.....
```

Sample Output 3

```
FLFRXFLFRFXFFFLFFLF
```

Sample Input 4

```
........
........
........
CCCCC...
..C.DC..
..C.C...
C.IC....
T.C.....
```

Sample Output 4

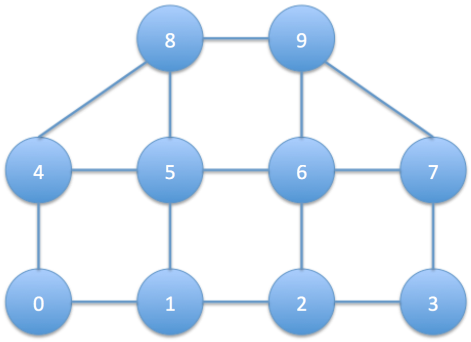
```
no solution
```

Problem I
Victory Through Synergy

Johnny has been roped into a fantasy soccer league and needs your help to set up the best team.

He has a list of 10 players that are on his fantasy team (no goalkeepers in this league). He also knows what country they're from, what league they play in, and what team they play for.

He doesn't know much about soccer, but he does know these things:



1. If two players are from the same country, a link between the two players will have a synergy score of 1.
2. If two players are in the same league, a link between the two players will have a synergy score of 1.
3. If two players are on the same team, a link between the two players will have a synergy score of 2.
4. If two players are from the same country and in the same league, a link between the two players will have a synergy score of 2.
5. If two players are from the same country and on the same team, a link between the two players will have a synergy score of 3.

A team can only be in one league and no two teams will have the same name unless they are the same team.

He has to place the players on his team into a formation of 10 nodes which can be represented as an undirected graph. The illustration shows the first sample. Therefore, Johnny has to place a player in each node of the graph. Given a particular formation and the members of Johnny's team, output whether it is possible for Johnny to organize his team to get a perfect team.

A team is a perfect team if and only if every player is placed on a node with a synergy score that is greater or equal to the node's degree. A node's degree is the number of links to which it is linked in the formation. A player placed on a node in the formation derives synergy from all players placed on nodes to which the player is linked in the formation. Thus, a node's synergy score is the sum of the synergy scores of all links to which the node is connected.

Input

The input will contain 1 test case. The first line of the input will have one integer c ($0 \leq c \leq 45$). c represents the number of edges in the formation. The next c lines represent the connections between nodes represented as two integers a ($0 \leq a < 10$) and b ($0 \leq b < 10$), where a is not equal to b. Then, the next 10 lines will be the players on Johnny's team. Each line will contain four strings in the following order: player name, nation, league, team. These are guaranteed to be non-empty with no spaces, no longer than 15 characters, and delimited by a single space between each piece of information.

Output

If a perfect team can be organized by Johnny, print `yes`. Otherwise, print `no`.

Sample Input 1

```
15
0 1
1 2
2 3
0 4
1 5
2 6
3 7
4 5
5 6
6 7
4 8
5 8
6 9
7 9
8 9
Griezmann France LaLiga AtleticoMadrid
Benzema France LaLiga RealMadrid
Ntep France Ligue1 StadeRennais
Sissoko France PremierLeague Spurs
Tolisso France Ligue1 Lyon
Diarra France Ligue1 OM
Evra France CalcioA Juventus
Koscielny France PremierLeague Arsenal
Varane France LaLiga RealMadrid
Sagna France PremierLeague ManCity
```

Sample Output 1

```
yes
```

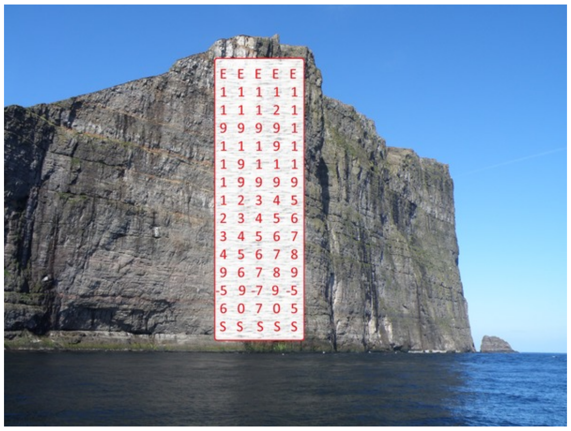
Sample Input 2

```
15
0 1
1 2
2 3
0 4
1 5
2 6
3 7
4 5
5 6
6 7
4 8
5 8
6 9
7 9
8 9
PlayerA France A1 A1-1
PlayerB France B1 B1-1
PlayerC France C1 C1-1
PlayerD France D1 D1-1
PlayerE France E1 E1-1
PlayerF France F1 F1-1
PlayerG France G1 G1-1
PlayerH France H1 H1-1
PlayerI France I1 I1-1
PlayerJ Germany J1 J1-1
```

Sample Output 2

```
no
```

Problem J
Rock Climbing



Peter is attempting to deep-water solo a rock climbing cliff over the ocean. Deep-water soloing (DWS) is a form of solo rock climbing that relies solely upon the presence of water at the base of the climb to protect against injury from falling.

Rock climbing is very exhausting and takes lots of energy. Since Peter is not very flexible, he can only move 1 unit in any of the four directions: Up, Down, Left, and Right. Traveling to a different square will decrease Peter's energy by the amount on that square. Note that the amount of energy on a square can be negative. In this case, Peter will gain energy.

If Peter's energy is negative, he will fall into the water.

Peter doesn't want to get wet, so he asks you how much energy he needs to complete the climb, assuming he takes the best route.

Input

The first line of the input will contain two integers, R, C $(1 \leq R, C \leq 15)$. The second line of input will consist of a row of C `E` characters, separated by spaces, representing the top of the cliff. These take 0 units of energy to enter. Peter can choose any of them.

Next, there will be R rows of C columns of numbers $X_{r,c}$, where $(-9 \leq X_{r,c} \leq 9)$, the energy required to enter that section of cliff. The final line of input will consist of a row of C `S` characters, representing the possible start points of the climb. These take 0 units of energy to enter. Returning to a starting position is allowed.

Output

Output a single integer, the amount of energy necessary to complete the climb without falling.

Sample Input 1

```
5 5
E E E E E
1 2 3 4 5
5 4 3 2 1
-2 -2 -2 -2 -2
8 8 8 8 8
9 9 9 9 9
S S S S S
```

Sample Output 1

```
17
```

Sample Input 2

```
13 5
E E E E E
1 1 1 1 1
1 1 1 2 1
9 9 9 9 1
1 1 1 9 1
1 9 1 1 1
1 9 9 9 9
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
9 6 7 8 9
-5 9 -7 9 -5
6 0 7 0 5
S S S S S
```

Sample Output 2

```
32
```