2014 ACM ICPC Southeast USA Regional Programming Contest

15 November, 2014

Division 2

Hosted by:

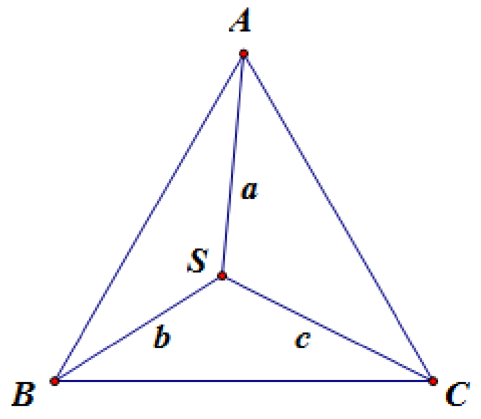
Florida Institute of Technology

Georgia Institute of Technology

University of West Florida

A: Stained Carpet

The Algebraist Carpet Manufacturing (ACM) group likes to produce area carpets based upon various geometric figures. The 2014 ACM carpets are all equilateral triangles. Unfortunately, due to a manufacturing defect, some of the carpets are not as stain-resistant as intended. The ACM group is offering to replace each defective carpet that contains a stain.



The web form used to report the stained carpet requests the three distances that the stain is away from the corners of the rug. Based upon these three numbers, you need to compute the area of the rug that is to be sent to the customer, or indicate that the customer's carpet doesn't come from ACM.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will consist of a single line with three floating point numbers ***a***, ***b*** and ***c*** ($0<a,b,c\leq 100$) representing the distances from the stain to each of the three corners of the carpet. There will be a single space between ***a*** and ***b***, and between ***b*** and ***c***.

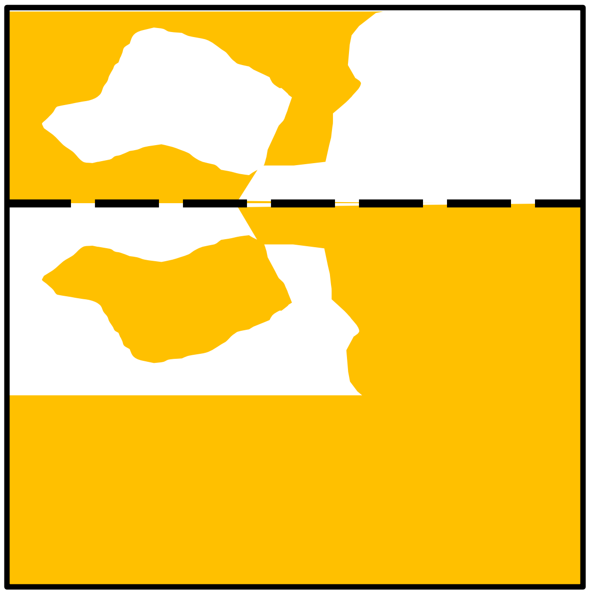
Output

Output a single line with a single floating point number. If there is a carpet that satisfies the constraints, output the area of this carpet. If not, output `-1.000`. Output this number to exactly three decimal places, rounded. Output no spaces.

Sample Input	Sample Output
`1 1 1.732051`	`1.732`
`1 1 3.0`	`-1.000`
`1.732051 1.732051 1.732051`	`3.897`

B: Gold Leaf

Gold Leaf is a very thin layer of gold, with a paper backing. If the paper gets folded and then unfolded, the gold leaf will stick to itself more readily than it will stick to the paper, so there will be patches of gold and patches of exposed paper. Note that the gold leaf will always stick to itself, rather than the paper. In the following example, the paper was folded along the dashed line. Notice how the gold leaf always sticks to one side or the other, never both.



Consider a crude digital image of a sheet of gold leaf. If the area covered by a pixel is mostly gold, that will be represented by a '**#**'. If it's mostly exposed paper, it will be represented by a '**.**'. Determine where the sheet was folded. The sheet was folded exactly once, along a horizontal, vertical, or 45 degree or 135 degree diagonal line. If the fold is horizontal or vertical, it is always between rows/columns. If the fold is diagonal, then the fold goes through a diagonal line of cells, and the cells along the fold are always '**#**'.

The Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will begin with a line with two integers, ***n*** and ***m*** (2≤***n***,***m***≤25), where ***n*** is the number of rows, and ***m*** is the number of columns of the image. Each of the next ***n*** lines will contain exactly ***m*** characters, all of which will be either '**#**' or '**.**'. This represents a crudely collected digital image of the sheet of gold leaf. There is guaranteed to be at least one '**.**', and there is guaranteed to be a solution.

The Output

Output a single line with four integers, with a single space between integers, indicating the places where the fold hits the edges of the paper. Output no extra spaces. Output the integers in this order:

r1 c1 r2 c2

where (***r1***,***c1***) and (***r2***,***c2***) are row/column coordinates (***r***=row, ***c***=column). The top left character of the image is (1,1) and the bottom right is (***n***,***m***).

If the fold is horizontal or diagonal, list the left coordinates before the right. If the fold is vertical, list the top coordinates before the bottom.

If the fold is horizontal, use the coordinates above the fold. If the fold is vertical, use the coordinates to the left of the fold. If the fold is diagonal, use the coordinates of the edge pixels that the fold goes through.

If more than one fold is possible, choose the one with the smallest first coordinate, then the smallest second coordinate, then third, then fourth.

Sample Input

```
8 10
#.#..##..#
####..####
###.##....
...#..####
....##....
.#.##..##.
##########
##########
```

```
5 20
###########.#.#.#.#.
###########...#.###.
##########..##.#..##
###########..#.#.##.
###########.###...#.
```

```
5 5
.####
###.#
##..#
#..##
#####
```

Sample Output

```
3 1 3 10
```

```
1 15 5 15
```

```
4 1 1 4
```

C: Hill Number

A *Hill Number* is a positive integer, the digits of which possibly rise and then possibly fall, but never fall and then rise. For example:

12321 is a hill number.

12223 is a hill number.

33322111 is a hill number.

1232321 is **not** a hill number.

Given a positive integer, if it is a hill number, print the number of positive hill numbers less than or equal to it. If it is not a hill number, print -1.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will consist of a single integer $\boldsymbol{n}$ ($1 \le n \le 10^{18}$).

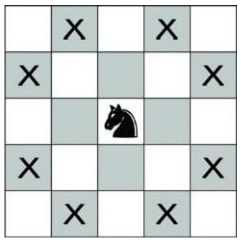
Output

Output a single line with a single integer. If the input is a hill number, then output the number of hill numbers less than or equal to it. If the input is not a hill number, then output -1. Output no spaces.

Sample Input	Sample Output
10	10
55	55
101	-1
1000	715
1234321	94708

D: Knight Moves

In Chess, a knight can move two squares in one direction and then one in a perpendicular direction. It can 'jump', meaning that it only requires that the destination square be open - the path between can be occupied. In this diagram, the knight could move to any of the Xs.



Given a grid, a starting point and destination point, determine the least number of moves the knight must make to get from the start to the destination. Some squares of the grid may be occupied, so that the knight cannot move there.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will begin with two integers ***n*** and ***m*** (2≤***n***,***m***≤100), indicating the height and width of the grid. Each of the next ***n*** lines will hold ***m*** characters, representing the grid. The grid will consist only of '`.`' (open square), '**`#`**' (occupied square), '**`K`**' (the knight's starting position) or '**`X`**' (the knight's destination). There will be exactly one '**`K`**' and exactly one '**`X`**' in each test case.

Output

Output a single line with a single integer indicating the minimum number of moves the knight needs to get to the destination, or **`-1`** if the knight cannot make it. Output no spaces.

Sample Input	Sample Output

`5 4`
`K...`
`....`
`.#..`
`.#..`
`...X`
	`5`

`3 3`
`K..`
`.X.`
`...`
	`-1`

E: Marble Madness

Mirko is playing a game of marbles. This is no ordinary game. In this game there are ***n*** bins placed from left to right. In each bin there is a number of marbles. In one move, Mirko can move a single marble from a bin to a directly adjacent bin. An adjacent bin is one that shares a side. He can move the same marble multiple times, but it's considered a different move every time. At the end of the game, the score is the calculated by adding up the absolute value of the difference in the number of marbles between each pair of adjacent bins.

Consider this example:

9	**8**	**3**	**2**	**7**	**2**	**3**	**4**	**6**

If the game ended this way, Mirko's score would be:

|9-8|+|8-3|+|3-2|+|2-7|+|7-2|+|2-3|+|3-4|+|4-6| = 1 + 5 + 1 + 5 + 5 + 1 + 1 + 2 = 21

Mirko would like to maximize his score, and then minimize the number of moves to achieve this score. Find these two values.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will begin with a single integer ***n*** (1≤***n***≤100,000) indicating the number of bins. On the next line there will be ***n*** integers ***m*** (0≤***m***≤1,000), with a single space between them, indicating the number of marbles in each bin, in order from left to right.

Output

Output a single line with two integers, separated by a single space. The first is the maximum score that Mirko can achieve, and the second is the minimum number of moves Mirko needs to get that score. Output no extra spaces.

Sample Input	**Sample Output**

`1`
`2`
	`0 0`

`2`
`2 3`
	`5 2`

`5`
`2 0 20 0 2`
	`48 8`

`5`
`9 1 2 1 9`
	`44 20`

F: Polling

Midterm elections are here! Help your local election commission by counting votes and telling them the winner. If more than one candidate ties with the most votes, print out all of their names in alphabetical order.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will begin with an integer ***n*** (1<=***n***<=1,000), indicating the number of votes. The next ***n*** lines will hold the votes. The candidates' names will appear one per line, and consist of between 1 and 20 capital letters only.

Output

Output the name of the candidate with the most votes. If there is a tie, output out all of the names of candidates with the most votes, one per line, in alphabetical order. Do not output any spaces, and do not output blank lines between names.

Sample Input	Sample Output
5 FRED BARNEY FRED FRED BARNEY	FRED
5 PORTHOS ATHOS ARAMIS PORTHOS ATHOS	ATHOS PORTHOS

G: Runes

You are helping an archaeologist decipher some runes. He knows that this ancient society used a Base 10 system, and that they never start a number with a leading zero unless it is exactly zero (single digit). He's figured out most of the digits, as well as a few operators, but he needs your help to figure out the rest.

The professor will give you a simple math expression. He has converted all of the runes he knows into digits. The only operators he knows are addition (**+**), subtraction (**−**), and multiplication (*****), so those are the only ones that will appear. Each number will be in the range from -999999 to 999999, and will consist of only the digits **0**-**9**, possibly a leading **−**, and possibly a few **?**s. The **?**s represent a digit rune that the professor doesn't know (never an operator, an **=**, or a leading **−**). All of the **?**s in an expression will represent the same digit (0-9), and it won't be one of the other given digits in the expression.

Given an expression, figure out the value of the rune represented by the question mark. If more than one digit works, give the lowest one. If no digit works, well, that's bad news for the professor - it means that he's gotten some of his runes wrong. Output **−1** in that case.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will consist of a single line, of the form:

[number][op][number]=[number]

Each ***[number]*** will consist of only the digits **0**-**9**, with possibly a single leading minus (**-**), and possibly some **?**s. No number will begin with a leading **0** unless it is **0**, no number will begin with **-0**, and no number will have more than 6 places (digits or **?**s). The ***[op]*** will separate the first and second ***[number]***s, and will be one of: **+**, **-** or *****. The **=** will always be present between the second and third ***[number]***s. There will be no spaces, tabs, or other characters. There is guaranteed to be at least one **?** in every equation.

Output

Output a single line with the lowest digit that will make the equation work when substituted for the **?**s, or output **−1** of no digit will work. Output no spaces.

Sample Input	**Sample Output**
`1+1=?`	`2`
`123*45?=5?088`	`6`
`-5?*-1=5?`	`0`
`19--45=5?`	`-1`
`??*??=302?`	`5`

H: Shuffles

The most common technique for shuffling a deck of cards is called the Riffle or Dovetail shuffle. The deck is split into two stacks, which are then interleaved with each other. The deck can be split anywhere, and the two stacks can be interleaved in any way.

For example, consider a deck with 10 unique cards:

```
1 2 3 4 5 6 7 8 9 10
```

Split them somewhere:

```
1 2 3 4 5 6          7 8 9 10
```

And interleave them in some way:

```
1 2 7 3 8 9 4 5 10 6
```

Do it again. Split them somewhere:

```
1 2 7          3 8 9 4 5 10 6
```

And interleave them in some way:

```
3 8 1 9 4 5 2 7 10 6
```

This is one possible ordering after 2 shuffles. Suppose there are ***n*** unique cards, and that they start out perfectly ordered: 1, 2, 3, ..., ***n***. Given an ordering of the deck, what is the smallest number of shuffles that could possibly put the deck in that order?

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will begin with a single integer ***n*** (1≤***n***≤1,000,000) indicating the number of cards in the deck. On the next line will be ***n*** unique integers ***c*** (1≤***c***≤***n***), with a single space between them, indicating an ordering of the ***n*** cards. The values ***c*** are guaranteed to be a permutation of the numbers 1..***n***.

Output

Output a single line with a single integer indicating the minimum number of shuffles that could possibly put the deck in the given order. Output no spaces.

Sample Input	Sample Output
10 1 2 7 3 8 9 4 5 10 6	1
10 3 8 1 9 4 5 2 7 10 6	2
8 2 1 4 3 6 5 8 7	3

I: Top 25

In College Football, many different sources create a list of the Top 25 (or, Top ***n***) teams in the country. Since it's subjective, these lists often differ, but they're usually very similar. Your job is to compare two of these lists, and determine where they are similar. In particular, you are to partition them into sets, where each set represents the same contiguous positions in both lists, and has the same teams, and is as small as possible. If the lists agree completely, you'll have ***n*** sets, where ***n*** is the number of teams in each list. For example consider these two lists:

A	A
B	C
C	D
D	B
E	E

In this case, there are 3 sets: A, BCD, and E.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will begin with an integer ***n*** (1≤***n***≤1,000,000), indicating the number of teams ranked. The next ***n*** lines will hold the first list, in order. The team names will appear one per line, and consist of between 1 and 8 capital letters only. After this will be ***n*** lines, in the same format, indicating the second list. Both lists will contain the same team names, and all ***n*** team names will be unique.

Output

Output the size of each set, in order, one per line. Do not output any spaces, and do not output blank lines between numbers.

Sample Input	Sample Output
5 A B C D E A C D B E	1 3 1
3 RED BLUE ORANGE RED BLUE ORANGE	1 1 1
3 MOE LARRY CURLY CURLY MOE LARRY	3