GPC-UPC Brute Force II - Recursion - Bitmask Contest

A. Boxes Packing

1 second, 256 megabytes

Mishka has got n empty boxes. For every i ($1 \le i \le n$), i-th box is a cube with side length a_i.

Mishka can put a box i into another box j if the following conditions are met:

- i-th box is not put into another box;
- j-th box doesn't contain any other boxes;
- box i is smaller than box j ($a_i < a_j$).

Mishka can put boxes into each other an arbitrary number of times. He wants to minimize the number of *visible* boxes. A box is called *visible* iff it is not put into some another box.

Help Mishka to determine the minimum possible number of *visible* boxes!

Input
The first line contains one integer n ($1 \le n \le 5000$) — the number of boxes Mishka has got.

The second line contains n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^9$), where a_i is the side length of i-th box.

Output
Print the minimum possible number of *visible* boxes.

input
```
3
1 2 3
```
output
```
1
```

input
```
4
4 2 4 3
```
output
```
2
```

In the first example it is possible to put box 1 into box 2, and 2 into 3.

In the second example Mishka can put box 2 into box 3, and box 4 into box 1.

B. Remove Extra One

2 seconds, 256 megabytes

You are given a permutation p of length n. Remove one element from permutation to make the number of records the maximum possible.

We remind that in a sequence of numbers $a_1, a_2, ..., a_k$ the element a_i is a *record* if for every integer j ($1 \le j < i$) the following holds: $a_j < a_i$.

Input
The first line contains the only integer n ($1 \le n \le 10^5$) — the length of the permutation.

The second line contains n integers $p_1, p_2, ..., p_n$ ($1 \le p_i \le n$) — the permutation. All the integers are distinct.

Output
Print the only integer — the element that should be removed to make the number of records the maximum possible. If there are multiple such elements, print the smallest one.

input
```
1
1
```
output
```
1
```

input
```
5
5 1 2 3 4
```
output
```
5
```

In the first example the only element can be removed.

C. Binary Number

1 second, 256 megabytes

Little walrus Fangy loves math very much. That's why when he is bored he plays with a number performing some operations.

Fangy takes some positive integer x and wants to get a number one from it. While x is not equal to 1, Fangy repeats the following action: if x is odd, then he adds 1 to it, otherwise he divides x by 2. Fangy knows that for any positive integer number the process ends in finite time.

How many actions should Fangy perform to get a number one from number x?

Input
The first line contains a positive integer x in a **binary system**. It is guaranteed that the first digit of x is different from a zero and the number of its digits does not exceed 10^6.

Output
Print the required number of actions.

input
```
1
```
output
```
0
```

input
```
1001001
```
output
```
12
```

input
```
101110
```
output
```
8
```

Let's consider the third sample. Number 101110 is even, which means that we should divide it by 2. After the dividing Fangy gets an odd number 10111 and adds one to it. Number 11000 can be divided by 2 three times in a row and get number 11. All that's left is to increase the number by one (we get 100), and then divide it by 2 two times in a row. As a result, we get 1.

D. Shockers

2 seconds, 256 megabytes

Valentin participates in a show called "Shockers". The rules are quite easy: jury selects one letter which Valentin doesn't know. He should make a small speech, but every time he pronounces a word that contains the selected letter, he receives an electric shock. He can make guesses which letter is selected, but for each incorrect guess he receives an electric shock too. The show ends when Valentin guesses the selected letter correctly.

Valentin can't keep in mind everything, so he could guess the selected letter much later than it can be uniquely determined and get excessive electric shocks. Excessive electric shocks are those which Valentin got after the moment the selected letter can be uniquely determined. You should find out the number of excessive electric shocks.

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the number of actions Valentin did.

The next n lines contain descriptions of his actions, each line contains description of one action. Each action can be of one of three types:

1. Valentin pronounced some word and didn't get an electric shock. This action is described by the string "`. w`" (without quotes), in which "`.`" is a dot (ASCII-code 46), and w is the word that Valentin said.
2. Valentin pronounced some word and got an electric shock. This action is described by the string "`! w`" (without quotes), in which "`!`" is an exclamation mark (ASCII-code 33), and w is the word that Valentin said.
3. Valentin made a guess about the selected letter. This action is described by the string "`? s`" (without quotes), in which "?" is a question mark (ASCII-code 63), and s is the guess — a lowercase English letter.

All words consist only of lowercase English letters. The total length of all words does not exceed 10^5.

It is guaranteed that last action is a guess about the selected letter. Also, it is guaranteed that Valentin didn't make correct guesses about the selected letter before the last action. Moreover, it's guaranteed that if Valentin got an electric shock after pronouncing some word, then it contains the selected letter; and also if Valentin didn't get an electric shock after pronouncing some word, then it does not contain the selected letter.

Output

Output a single integer — the number of electric shocks that Valentin could have avoided if he had told the selected letter just after it became uniquely determined.

input
```
5
! abc
. ad
. b
! cd
? c
```
output
```
1
```

input
```
8
! hello
! codeforces
? c
. o
? d
? h
. l
? e
```
output
```
2
```

input
```
7
! ababahalamaha
? a
? b
? a
? b
? a
? h
```
output
```
0
```

In the first test case after the first action it becomes clear that the selected letter is one of the following: a, b, c. After the second action we can note that the selected letter is not a. Valentin tells word "b" and doesn't get a shock. After that it is clear that the selected letter is c, but Valentin pronounces the word cd and gets an excessive electric shock.

In the second test case after the first two electric shocks we understand that the selected letter is e or o. Valentin tries some words consisting of these letters and after the second word it's clear that the selected letter is e, but Valentin makes 3 more actions before he makes a correct hypothesis.

In the third example the selected letter can be uniquely determined only when Valentin guesses it, so he didn't get excessive electric shocks.

E. Short Program

2 seconds, 256 megabytes

Petya learned a new programming language CALPAS. A program in this language always takes one non-negative integer and returns one non-negative integer as well.

In the language, there are only three commands: apply a bitwise operation AND, OR or XOR with a given constant to the current integer. A program can contain an arbitrary sequence of these operations with arbitrary constants from 0 to 1023. When the program is run, all operations are applied (in the given order) to the argument and in the end the result integer is returned.

Petya wrote a program in this language, but it turned out to be too long. Write a program in CALPAS that does the same thing as the Petya's program, and consists of no more than 5 lines. Your program should return the same integer as Petya's program for all arguments from 0 to 1023.

Input

The first line contains an integer n ($1 \le n \le 5 \cdot 10^5$) — the number of lines.

Next n lines contain commands. A command consists of a character that represents the operation ("&", "`|`" or "^" for AND, OR or XOR respectively), and the constant x_i $0 \le x_i \le 1023$.

Output

Output an integer k ($0 \le k \le 5$) — the length of your program.

Next k lines must contain commands in the same format as in the input.

input
```
3
| 3
^ 2
| 1
```
output
```
2
| 3
^ 2
```

input
```
3
& 1
& 3
& 5
```
output
```
1
& 1
```

input
```
3
^ 1
^ 2
^ 3
```
output
```
0
```

You can read about bitwise operations in https://en.wikipedia.org/wiki/Bitwise_operation.

Second sample:

Let x be an input of the Petya's program. It's output is $((x\&1)\&3)\&5 = x\&(1\&3\&5) = x\&1$. So these two programs always give the same outputs.

F. Colorful Points

2 seconds, 256 megabytes

You are given a set of points on a straight line. Each point has a color assigned to it. For point a, its neighbors are the points which don't have any other points between them and a. Each point has at most two neighbors - one from the left and one from the right.

You perform a sequence of operations on this set of points. In one operation, you delete all points which have a neighbor point of a different color than the point itself. Points are deleted simultaneously, i.e. first you decide which points have to be deleted and then delete them. After that you can perform the next operation etc. If an operation would not delete any points, you can't perform it.

How many operations will you need to perform until the next operation does not have any points to delete?

Input
Input contains a single string of lowercase English letters 'a'-'z'. The letters give the points' colors in the order in which they are arranged on the line: the first letter gives the color of the leftmost point, the second gives the color of the second point from the left etc.

The number of the points is between 1 and 10^6.

Output
Output one line containing an integer - the number of operations which can be performed on the given set of points until there are no more points to delete.

input
```
aabb
```
output
```
2
```

input
```
aabcaa
```
output
```
1
```

In the first test case, the first operation will delete two middle points and leave points "ab", which will be deleted with the second operation. There will be no points left to apply the third operation to.

In the second test case, the first operation will delete the four points in the middle, leaving points "aa". None of them have neighbors of other colors, so the second operation can't be applied.

G. I'm bored with life

1 second, 256 megabytes

Holidays have finished. Thanks to the help of the hacker Leha, Noora managed to enter the university of her dreams which is located in a town Pavlopolis. It's well known that universities provide students with dormitory for the period of university studies. Consequently Noora had to leave Vičkopolis and move to Pavlopolis. Thus Leha was left completely alone in a quiet town Vičkopolis. He almost even fell into a depression from boredom!

Leha came up with a task for himself to relax a little. He chooses two integers A and B and then calculates the greatest common divisor of integers " A factorial" and " B factorial". Formally the hacker wants to find out GCD$(A!, B!)$. It's well known that the factorial of an integer x is a product of all positive integers less than or equal to x. Thus $x! = 1 \cdot 2 \cdot 3 \cdot ... \cdot (x - 1) \cdot x$. For example $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$. Recall that GCD$(x, y)$ is the largest positive integer q that divides (without a remainder) both x and y.

Leha has learned how to solve this task very effective. You are able to cope with it not worse, aren't you?

Input
The first and single line contains two integers A and B $(1 \le A, B \le 10^9, min(A, B) \le 12)$.

Output
Print a single integer denoting the greatest common divisor of integers $A!$ and $B!$.

input
```
4 3
```
output
```
6
```

Consider the sample.

$4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$. $3! = 1 \cdot 2 \cdot 3 = 6$. The greatest common divisor of integers 24 and 6 is exactly 6.

H. Almost Prime

2 seconds, 256 megabytes

A number is called almost prime if it has exactly two distinct prime divisors. For example, numbers 6, 18, 24 are almost prime, while 4, 8, 9, 42 are not. Find the amount of almost prime numbers which are between 1 and n, inclusive.

Input
Input contains one integer number n $(1 \le n \le 3000)$.

Output
Output the amount of almost prime numbers between 1 and n, inclusive.

input
```
10
```
output
```
2
```

input
```
21
```
output
```
8
```

I. New Year and Buggy Bot

1 second, 256 megabytes

Bob programmed a robot to navigate through a 2d maze.

The maze has some obstacles. Empty cells are denoted by the character '.', where obstacles are denoted by '#'.

There is a single robot in the maze. Its start position is denoted with the character 'S'. This position has no obstacle in it. There is also a single exit in the maze. Its position is denoted with the character 'E'. This position has no obstacle in it.

The robot can only move up, left, right, or down.

When Bob programmed the robot, he wrote down a string of digits consisting of the digits 0 to 3, inclusive. He intended for each digit to correspond to a distinct direction, and the robot would follow the directions in order to reach the exit. Unfortunately, he forgot to actually assign the directions to digits.

The robot will choose some random mapping of digits to distinct directions. The robot will map distinct digits to distinct directions. The robot will then follow the instructions according to the given string in order and chosen mapping. If an instruction would lead the robot to go off the edge of the maze or hit an obstacle, the robot will crash and break down. If the robot reaches the exit at any point, then the robot will stop following any further instructions.

Bob is having trouble debugging his robot, so he would like to determine the number of mappings of digits to directions that would lead the robot to the exit.

Input

The first line of input will contain two integers n and m $(2 \le n, m \le 50)$, denoting the dimensions of the maze.

The next n lines will contain exactly m characters each, denoting the maze.

Each character of the maze will be '`.`', '#', 'S', or 'E'.

There will be exactly one 'S' and exactly one 'E' in the maze.

The last line will contain a single string s $(1 \le |s| \le 100)$ — the instructions given to the robot. Each character of s is a digit from 0 to 3.

Output

Print a single integer, the number of mappings of digits to directions that will lead the robot to the exit.

input
```
5 6
.....#
S....#
.#....
.#....
...E..
333300012
```
output
```
1
```

input
```
6 6
......
......
..SE..
......
......
......
01232123212302123021
```
output
```
14
```

input
```
5 3
...
.S.
###
.E.
...
3
```
output
```
0
```

For the first sample, the only valid mapping is $0 \to D, 1 \to L, 2 \to U, 3 \to R$, where D is down, L is left, U is up, R is right.

J. Pairs of Numbers

1 second, 256 megabytes

Let's assume that we have a pair of numbers (a, b). We can get a new pair ($a + b, b$) or ($a, a + b$) from the given pair in a single step.

Let the initial pair of numbers be (1,1). Your task is to find number k, that is, the least number of steps needed to transform (1,1) into the pair where at least one number equals n.

Input

The input contains the only integer n $(1 \le n \le 10^6)$.

Output

Print the only integer k.

input
```
5
```
output
```
3
```

input
```
1
```
output
```
0
```

The pair (1,1) can be transformed into a pair containing 5 in three moves: (1,1) → (1,2) → (3,2) → (5,2).

K. Lucky Sum

2 seconds, 256 megabytes

Petya loves lucky numbers. Everybody knows that lucky numbers are positive integers whose decimal representation contains only the lucky digits **4** and **7**. For example, numbers **47**, **744**, **4** are lucky and **5**, **17**, **467** are not.

Let $next(x)$ be the minimum lucky number which is larger than or equals x. Petya is interested what is the value of the expression $next(l) + next(l + 1) + ... + next(r - 1) + next(r)$. Help him solve this problem.

Input

The single line contains two integers l and r $(1 \le l \le r \le 10^9)$ — the left and right interval limits.

Output

In the single line print the only number — the sum $next(l) + next(l + 1) + ... + next(r - 1) + next(r)$.

Please do not use the `%lld` specificator to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specificator.

input
```
2 7
```
output
```
33
```

input
```
7 7
```
output
```
7
```

In the first sample: $next(2) + next(3) + next(4) + next(5) + next(6) + next(7) = 4 + 4 + 4 \cdot$

In the second sample: $next(7) = 7$

L. Dreamoon and WiFi

1 second, 256 megabytes

Dreamoon is standing at the position 0 on a number line. Drazil is sending a list of commands through Wi-Fi to Dreamoon's smartphone and Dreamoon follows them.

Each command is one of the following two types:

1. Go 1 unit towards the positive direction, denoted as `'+'`
2. Go 1 unit towards the negative direction, denoted as `'-'`

But the Wi-Fi condition is so poor that Dreamoon's smartphone reports some of the commands can't be recognized and Dreamoon knows that some of them might even be wrong though successfully recognized. Dreamoon decides to follow every recognized command and toss a fair coin to decide those unrecognized ones (that means, he moves to the 1 unit to the negative or positive direction with the same probability 0.5).

You are given an original list of commands sent by Drazil and list received by Dreamoon. What is the probability that Dreamoon ends in the position originally supposed to be final by Drazil's commands?

Input

The first line contains a string s_1 — the commands Drazil sends to Dreamoon, this string consists of only the characters in the set {`'+'`, `'-'`}.

The second line contains a string s_2 — the commands Dreamoon's smartphone recognizes, this string consists of only the characters in the set {`'+'`, `'-'`, `'?'`}. `'?'` denotes an unrecognized command.

Lengths of two strings are equal and do not exceed 10.

Output

Output a single real number corresponding to the probability. The answer will be considered correct if its relative or absolute error doesn't exceed 10^{-9}.

input

```
++-+-
+-+-+
```

output

```
1.000000000000
```

input

```
+-+-
+-??
```

output

```
0.500000000000
```

input

```
+++
??-
```

output

```
0.000000000000
```

For the first sample, both s_1 and s_2 will lead Dreamoon to finish at the same position $+1$.

For the second sample, s_1 will lead Dreamoon to finish at position 0, while there are four possibilites for s_2: {`"+-++"`, `"+-+-"`, `"+--+"`, `"+---"`} with ending position {+2, 0, 0, -2} respectively. So there are 2 correct cases out of 4, so the probability of finishing at the correct position is 0.5.

For the third sample, s_2 could only lead us to finish at positions {+1, -1, -3}, so the probability to finish at the correct position $+3$ is 0.