Kattis

AUPC Contest

2025-03-29

Problems

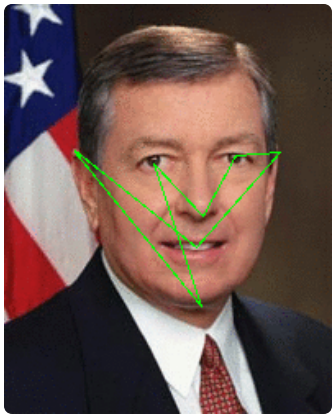
A. Biometrics
B. A Flea on a Chessboard
C. A Feast For Cats
D. 0-1 Sequences
E. if-then-else
F. 3D Printed Statues
G. Window Shopping
H. Array Smoothening
I. ilove Strings
J. Typo
K. Cracking The Safe

Advice, hints, and general information

- Your solution programs should read input from standard input (e.g. System.in in Java or std::cin in C++) and produce output on standard output (e.g. System.out in Java or std::cout in C++). Anything written on standard error will be ignored. For further details and examples, please refer to the documentation in the help pages for your favorite language on Kattis.
- If you think some problem is ambiguous or underspecified, you may ask the judges for a clarification request through the Kattis system. The most likely response is “No comment, read problem statement”, indicating that the answer can be deduced by carefully reading the problem statement or by checking the sample test cases given in the problem.

Biometrics

CPU TIME LIMIT	MEMORY LIMIT
1 second	1024 MB



Recently, the term *Biometrics* has been used to refer to the emerging field of technology devoted to identification of individuals using biological traits, such as those based on retinal or iris scanning, fingerprints, or face recognition.
A simple biometric system translates a human image into a polygon by considering certain features (eyes, nose, ears, etc.) to be vertices and connecting them with line segments. The polygon has distinct vertices but may be degenerate in that the line segments could intersect. Because these polygons are generally created from remote images, there is some uncertainty as to their scale and rotation. Your job is to determine whether or not two polygons are similar; that is, can they be made equal by repositioning, rotating and magnifying them?

Input

Input consists of several test cases. Each test case consists of three lines containing:

- f, the number of features
- f coordinate pairs giving the vertices of the first polygon
- f coordinate pairs giving the vertices of the second polygon

The vertices for both polygons correspond to the same set of features in the same order; for example, *right ear tip*, *chin cleft*, *right eye*, *nose*, *left eye*, *left ear tip*, *space between front teeth*.

Each polygon has f distinct vertices; each vertex is given as an x and y coordinate pair. There are at least three and no more than ten features. Coordinates are integers between -2000

and 2000. A line containing 0 follows the last test case.

Output

For each case, output a line "`similar`" or "`dissimilar`" as appropriate. The two polygons are similar if, after some combination of translation, rotation, and scaling (but not reflection) both vertices corresponding to each feature are in the same position.

Sample Input 1

```
4
0 0 0 1 1 1 1 0
0 1 1 0 0 -1 -1 0
3
0 0 10 0 10 10
0 0 -10 0 -10 10
3
0 0 10 10 20 20
0 0 11 11 22 22
3
0 0 10 10 20 20
0 0 11 11 20 20
0
```

Sample Output 1

```
similar
dissimilar
similar
dissimilar
```

A Flea on a Chessboard

CPU TIME LIMIT	MEMORY LIMIT
1 second	1024 MB

An infinite chessboard is obtained by extending a finite chessboard to the right and up infinitely. Each square of the chessboard is either black or white with the side length of S millimeters. The leftmost bottom square of the chessboard is black. A flea is positioned on the chessboard at the point (x, y) (given in millimeters) and makes jumps by jumping dx millimeters to the right and dy millimeters up. That is, a flea at position (x, y) after one jump lands at position $(x + dx, y + dy)$.

Given the starting position of the flea on the board your task is to find out after how many jumps the flea will reach a white square. If the flea lands on a boundary between two squares then it does not count as landing on the white square. Note that it is possible that the flea never reaches a white square.

Input

Input consists of multiple test cases, at most 1024. Each test case consists of one line of input containing five integers S, x, y, dx, and dy ($1 \leq S \leq 1000$, $0 \leq x, y \leq 10^6$, $1 \leq dx, dy \leq 10^6$). An input line containing five zeroes follows the last test case.

Output

For each test case print one line of output in the format shown in the sample.

Sample Input 1

```
10 2 3 3 2
100 49 73 214 38
25 0 0 5 25
407 1270 1323 1 1
18 72 6 18 6
407 1270 1170 100 114
0 0 0 0 0
```

Sample Output 1

```
After 3 jumps the flea lands at (11, 9).
After 1 jumps the flea lands at (263, 111).
The flea cannot escape from black squares.
After 306 jumps the flea lands at (1576, 1629).
The flea cannot escape from black squares.
After 0 jumps the flea lands at (1270, 1170).
```

A Feast For Cats

CPU TIME LIMIT	MEMORY LIMIT
6 seconds	1024 MB

Your crazy aunt has asked you to watch her cats while she's attending a seminar about making cat hats out of cat fur. Your aunt owns a great number of cats – all toms – and, due to the complex social structure of cats, every cat has a specific but different amount of hate reserved for every other cat.

Over the years, the cats have been spoiled to the point that they demand (at the least) 1 milliliter of milk at tea time. Your task is to feed the cats the aforementioned milk, in order to dull their primal instincts and prevent an all-out cat war.

The task is complicated by three things:

1. Your aunt has a limited amount of milk stored in the fridge.
2. For each pair of cats, there is a distance they have to be kept from each other to avoid infighting.
3. The excessive amount of meowing has driven you to drink large amounts of hard liquor, and now you can't move liquids without constant spilling. You will spill 1 milliliter of the milk you carry for every meter you walk.

Given an amount of milk, and a set of cats – each pair of cats being a given distance apart from each other – determine if it's possible to feed every cat at least 1 milliliter of milk. Every cat has a cat bowl that can hold an arbitrary amount of milk. For every meter you walk, you will definitely spill 1 milliliter of milk (instantly spoiling it), but instead of carrying all the milk at once, you may serve a cat more milk than needed and pick it up when backtracking

(the cat will not drink the excess milk before you're finished walking). The cats will never move, but they will eye each other with intense hate.

Input

The first line of the input consists of a single integer T, the number of test cases.
Each of the following T cases begins with a line containing two positive integers M and C, separated by a space, denoting the amount of milk in the fridge in milliliters and the number of cats, respectively.
This is followed by $\frac{C \cdot (C-1)}{2}$ (combinations of cats) lines containing three positive integers i j $D_{i,j}$, separated by single spaces, describing the distance between the cat with index i and the cat with index j in meters. Cats are numbered from 0 up to and including $C-1$, and each pair of cats will be listed exactly once. You may assume that the fridge is placed at the very same location as cat 0.

- $1 \leq T \leq 20$
- $1 \leq M \leq 20\,000$
- $1 \leq C \leq 2\,000$
- $1 \leq D_{i,j} \leq 3\,000$
- Cats co-exist in dimensions far greater than our three, so you can assume that every pair of cats is correctly separated by *exactly* the given distance.

Output

Output `yes` if it's possible to serve every cat at least 1 milliliter of milk. Output `no` if it's not.

Sample Input 1

```
1
20 5
0 1 4
0 2 3
0 3 10
0 4 15
1 2 7
1 3 3
1 4 5
2 3 4
2 4 3
3 4 8
```

Sample Output 1

```
yes
```

0-1 Sequences

CPU TIME LIMIT	MEMORY LIMIT
1 second	1024 MB

You are given a sequence, in the form of a string with characters '0', '1', and '?' only. Suppose there are k '?'s. Then there are 2^k ways to replace each '?' by a '0' or a '1', giving 2^k different 0-1 sequences (0-1 sequences are sequences with only zeroes and ones).

For each 0-1 sequence, define its number of inversions as the minimum number of adjacent swaps required to sort the sequence in non-decreasing order. In this problem, the sequence is sorted in non-decreasing order precisely when all the zeroes occur before all the ones. For example, the sequence 11010 has 5 inversions. We can sort it by the following moves: 11010 $\rightarrow$ 11001 $\rightarrow$ 10101 $\rightarrow$ 01101 $\rightarrow$ 01011 $\rightarrow$ 00111.

Find the sum of the number of inversions of the 2^k sequences, modulo $1\,000\,000\,007$ ($10^9 + 7$).

Input

The first and only line of input contains the input string, consisting of characters '0', '1', and '?' only, and the input string has between 1 to $500\,000$ characters, inclusive.

Output

Output an integer indicating the aforementioned number of inversions modulo $1\,000\,000\,007$.

Sample Input 1

```
?0?
```

Sample Output 1

```
3
```

if-then-else

CPU TIME LIMIT	MEMORY LIMIT
1 second	1024 MB

Saladin loves to make computers and languages. This time, he has made a CPU that works on 12-bit words, and created his own language called Salang on top of it. The language itself works well, except for one weird bug. To fix the problem, he has designed a decompiler: A program that takes the machine code and creates an intermediate representation of the program.

The intermediate representation is in a small language that can only do assignments, integer addition and multiplication, if statements and printing. The integer operations silently overflow and wrap around, and the representation has no scoping (a variable `@var` has the same memory address everywhere). It uses the following EBNF syntax:

```
<letter> := 'a' | 'b' ... 'y' | 'z'
<var> := '@' letter { letter }
<uint> := ? base 10 of an unsigned int lower than 2^12 ?
<val> := <var> | <uint>
<op> := '+' | '*'
<cmp> := '==' | '<=' | '<'
<assign> := <var> '=' <val> [ <op> <val> ] '\n'
<if> := 'if' <val> <cmp> <val> 'then' '\n' { <statement> }
        [ 'else' '\n' { <statement> } ]
        'endif' '\n'
<print> := 'print' <val>
<statement> := <assign> | <if> | <print>
<program> := { <statement> }
```

where all values are separated by exactly one space, unless if they begin on a new line.

But the decompiler has forgotten to output the first line! Although all variables are initialised to 0 before the program starts, the program always begins by assigning the magic variable `@a` to a fixed number. Before Saladin can start with the debugging, he needs to know the initial value of `@a`.

Input

The input begins with a single line with two integers: I and O. Then comes I lines, the program itself. Finally comes O lines, the numbers this program printed when it ran.

Output

Output the value `@a` has at the start of the program. If there are multiple valid answers, output the lowest one. If there are no valid answers, output "`no solution`".

Limits

- $0 \leq O < I \leq 100$
- The program matches the EBNF above
- The length of a line will always be less than 80 characters

Sample Input 1

```
12 2
@b = @a + @a
@c = @a * @a
if @c <= @b then
@dx = @b + 3996
if @dx < @a then
print @c
else
endif
endif
@a = @b * @dx
@a = @a + @c
print @a
129
4029
```

Sample Output 1

```
65
```

3D Printed Statues

CPU TIME LIMIT	MEMORY LIMIT
1 second	1024 MB

You have a single 3D printer, and would like to use it to produce n statues. However, printing the statues one by one on the 3D printer takes a long time, so it may be more time-efficient to first use the 3D printer to print a new printer. That new printer may then in turn be used to print statues or even more printers. Print jobs take a full day, and every day you can choose for each printer in your possession to have it print a statue, or to have it 3D print a new printer (which becomes available for use the next day).

Picture by Ariosvaldo Gonzáfoles, cc-by

What is the minimum possible number of days needed to print at least n statues?

Input

The input contains a single integer n ($1 \leq n \leq 10\,000$), the number of statues you need to print.

Output

Output a single integer, the minimum number of days needed to print at least n statues.

Sample Input 1

```
1
```

Sample Output 1

```
1
```

Sample Input 2

```
5
```

Sample Output 2

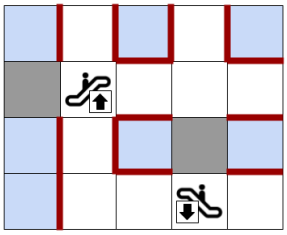
```
4
```

Window Shopping

CPU TIME LIMIT	MEMORY LIMIT
1 second	2048 MB

You are to help a newly built shopping mall design its floor layout. The shopping mall can be viewed as a rectangular grid of cells. Two cells are adjacent if they share an edge. A cell may be empty, a pillar, or an escalator. There are exactly two escalators: one goes upstairs and the other downstairs. Note that the escalators are small, so that customers to the mall can pass freely through their locations without going up or down.

You are to choose some of the empty cells to build shops. All remaining empty cells, plus the two escalator cells, are *walkable* by the customers. If an empty cell is reachable from both escalators via some walkable cells, it becomes a *hallway*. A shop window can be installed on an edge between a shop and a hallway.



One optimal way of installing windows for the first sample case. The blue cells are the shops. The red edges are the windows.

After the floor planning, it is allowed to have some empty cells not reachable from both escalators. These empty cells are thus not hallways and cannot have windows installed around them. Also, an escalator is not a hallway and windows cannot be installed on any side of an escalator.

Based on the latest customer analysis report, the profit of the shopping mall grows proportionally to the number of windows inside the mall. You need to therefore determine the maximum number of shop windows that can be installed in the shopping mall.

Input

The first line of input contains two positive integers r and c ($4 \leq r \cdot c \leq 99$), which are the size of the shopping mall. The mall is a grid with r rows and c columns of cells.

Each of the next r lines contains a string of length c. Each character in the string is one of:

- a dot '`.`' representing an empty cell
- a hash '`#`' representing a pillar
- a letter '`U`' representing an Up escalator (Exactly one of these will appear)
- a letter '`D`' representing a Down escalator (Exactly one of these will appear)

It is guaranteed that initially all empty cells are reachable from both escalators.

Output

Output a single integer, which is the maximum number of windows the shopping mall can accommodate.

Sample Input 1

```
4 5
.....
#U...
...#.
...D.
```

Sample Output 1

```
13
```

Sample Input 2

```
4 4
..#U
..#D
..#.
....
```

Sample Output 2

```
5
```

Sample Input 3

```
3 2
##
.D
U.
```

Sample Output 3

```
0
```

Kattis

Array Smoothening

CPU TIME LIMIT	MEMORY LIMIT
1 second	1024 MB

You've just finished writing a random array generator for your ~~competitive programming~~ practical examination problem. However, when you tested it, you noticed that the distribution is skewed; some values may appear much more times than the other values. You wanted to smoothen the distribution, but you don't want to modify your generator anymore. Thus, you decide to remove exactly K elements from the array, such that the maximum occurrence of a value is minimized. Now, given an array A of size N, determine the minimum possible maximum occurrence of a value from the array after you remove exactly K elements from it.

Input

The first line contains two integers N ($1 \leq N \leq 200\,000$) and K ($0 \leq K \leq N$).

The next line contains N integers $A[i]$ ($1 \leq A[i] \leq 10^9$), the elements of the array A.

Output

Print the minimum possible maximum occurrence of a value of A after exactly K removal.

Subtasks

1. (10 **Points**): $K = 0$, $N \leq 3\,000$, $A[i] \leq 3\,000$, and A is sorted in non-decreasing order.
2. (10 **Points**): $K = 0$, $N \leq 3\,000$, and $A[i] \leq 3\,000$.
3. (10 **Points**): $K = 0$ and $N \leq 3\,000$.
4. (20 **Points**): $K = 0$.

5. (30 **Points**): $N \leq 3\,000$.

6. (20 **Points**): No additional constraints.

Explanation

In the first sample, 1 appears twice, 7 appears three times, 3 appears five times, and none of the elements will be removed. Thus, the maximum occurrence is five.

In the second sample, we can remove two occurrences of 3 and one occurrence of 7. Hence, 1 and 7 appear twice, and 3 appears three times. Thus, the maximum occurrence is three.

Warning

The I/O files are large. Please use fast I/O methods.

Sample Input 1

```
10 0
1 1 3 3 3 3 3 7 7 7
```

Sample Output 1

```
5
```

Sample Input 2

```
10 3
3 1 7 3 3 3 7 3 1 7
```

Sample Output 2

```
3
```

ilove Strings

CPU TIME LIMIT	MEMORY LIMIT
2 seconds	1024 MB

It's that time of year when love is in the air. You're no stranger to love. You are obsessed with strings but not just any strings. You love "ilove" Strings. An "ilove" String is a string of length 5 with the following properties:

- Alternates between vowels (excluding 'y' and 'Y') and consonants (including 'y' and 'Y')
- Begins with a vowel (excluding 'y' and 'Y')
- Consists of 5 pairwise distinct characters (distinguishing between upper and lower case)

Examples of "ilove" Strings includes "`ilove`", "`image`", "`IxoXO`", and "`abide`". Examples of non-"ilove" Strings include , "`ideas`", "`maker`", "`inane`", "`oxOXo`" and "`abides`".

The loveliness of a string is the number of subsequences of the string that form an "ilove" String. Although "`ilooove`" is not an "ilove" String, it does have a loveliness of 3.

Input

Input contains a single string of between 1 and 100 000 lowercase and uppercase Latin characters, representing the string whose loveliness is to be computed.

Output

For the provided string, print one line with a single integer L — the loveliness of the string modulo $10^9 + 7$.

Sample Input 1

```
ilovestrings
```

Sample Output 1

```
4
```

Sample Input 2

```
idont
```

Sample Output 2

```
0
```

Sample Input 3

```
CAPital
```

Sample Output 3

```
1
```

Typo

CPU TIME LIMIT	MEMORY LIMIT
6 seconds	1024 MB

It is now far into the future and human civilization is ancient history. Archaeologists from a distant planet have recently discovered Earth. Among many other things, they want to decipher the English language.

They have collected many printed documents to form a dictionary, but are aware that sometimes words are not spelled correctly (typos are a universal problem). They want to classify each word in the dictionary as either correct or a typo. Naïvely, they do this using a simple rule: a typo is any word in the dictionary such that deleting a single character from that word produces another word in the dictionary.

Help these alien archaeologists out! Given a dictionary of words, determine which words are typos. That is, which words result in another word in the dictionary after deleting a single character.

For example if our dictionary is {`hoose`, `hose`, `nose`, `noises`}. Then `hoose` is a typo because we can obtain `hose` by deleting a single '`o`' from `hoose`. But `noises` is not a typo because deleting any single character does not result in another word in the dictionary.

However, if our dictionary is {`hoose`, `hose`, `nose`, `noises`, `noise`} then the typos are `hoose`, `noises`, and `noise`.

Input

The first line of input contains a single integer n, indicating the number of words in the dictionary.

The next n lines describe the dictionary. The i^{th} of which contains the i^{th} word in the dictionary. Each word consists only of lowercase English letters. All words are unique.

The total length of all strings is at most 1 000 000.

Output

Display the words that are typos in the dictionary. These should be output in the same order they appear in the input. If there are no typos, simply display the phrase `NO TYPOS`.

Sample Input 1

```
5
hoose
hose
nose
noises
noise
```

Sample Output 1

```
hoose
noises
noise
```

Sample Input 2

```
4
hose
hoose
oose
moose
```

Sample Output 2

```
hoose
moose
```

Sample Input 3

```
5
banana
bananana
bannanaa
orange
orangers
```

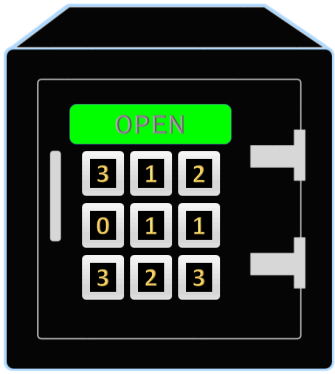
Sample Output 3

```
NO TYPOS
```

Cracking The Safe

CPU TIME LIMIT	MEMORY LIMIT
1 second	1024 MB

Your little sister misplaced the code for her toy safe - can you help her?
This particular safe has 9 buttons with digital displays. Each button shows a single digit in the range $0..3$. When you push one of the buttons, the number it displays is incremented by 1, circling around from 3 to 0. However, pushing a button will also increment the other digits in the same row and the same column as the button pushed.



The safe opens when the display shows nine zeros.

For instance, if you pushed the top-left, center, center, and middle-right buttons, in this order, the safe's display would change like so:

```
3 1 2     0 2 3     0 3 3     0 0 3     0 0 0
0 1 1  -> 1 1 1  -> 2 2 2  -> 3 3 3  -> 0 0 0
3 2 3     0 2 3     0 3 3     0 0 3     0 0 0
```

Write a program to determine if the safe can be opened, and if so, how many button pushes it would take!

Input

The input is a single test case, given as 9 digits d, $(0 \leq d \leq 3)$ on 3 lines, representing the digits that are initially displayed on the safe's buttons. Your program will be run multiple times on different inputs.

Output

Output the number of times buttons need to be pushed to open the safe! (The same button may need to be pushed more than once, and you do not have to output which buttons must be pushed.) If the safe cannot be opened, output `-1`.

Sample Input 1

```
3 1 2
0 1 1
3 2 3
```

Sample Output 1

```
4
```

Sample Input 2

```
0 0 3
2 2 3
2 2 1
```

Sample Output 2

```
-1
```