Problem A: Collatz Conjecture

The Collatz conjecture which is also known as the $3n + 1$ conjecture is a very well known and old conjecture in mathematics. The conjecture is as follows. Take any natural number n. If n is even, divided by two to get $n/2$ and if n is odd number greater than 1, triple it and add one to obtain $3n + 1$. Repeat this process to get a sequence of natural numbers known as the Hailstone sequence. The conjecture is that no matter what number you start, you always reach 1. The hailstone sequence for $n = 3$ is "$3, 10, 5, 16, 8, 4, 2, 1$". Paul Erdos said "Mathematics is not yet ripe for such problems" and offered $500 for its solution. Now it 's time to show Erdos that the Collatz conjecture can be proved for small numbers in 11^{th} Iran Internet Programming Contest. You are to write a program that computes the length of the Hailstone sequence for the given n.

Input (Standard Input)

There are multiple test cases in the input. Each test case consists of a line containing a non-negative integers $0 \leq n \leq 100$. The input terminates with "0" which should not be processed.

Output (Standard Output)

For each test case, output the length of the Hailstone sequence in one line.

Sample Input and Output

Standard Input	Standard Output
1 2 3 0	1 2 8

Problem B: Smart File Name Sorting

You have likely tried to sort a number of files in a directory based on their names. As you have noticed, in old basic environments, the file names are sorted in the ASCII-based lexicographic order. The sorting of the alphanumeric ASCII characters is as follows:

$$0 < 1 < \ldots < 9 < A < B < \ldots < Z < a < b < \ldots < z$$

Thus, the following file names would be placed in the following order:

A, A0, A01, A02, A1, A10, A2, AA, AB, Aa, Ab, B, B0, a, a0

But, the sorting that we usually would like is the following:

a, A, a0, A0, A01, A1, A02, A2, A10, Aa, AA, Ab, AB, B, B0

Our desired sorting can be formally defined with specifying the way of comparing two file names:

1. If two file names are exactly the same, they are equal. Otherwise, they are not considered to be equal!
2. Any maximal block of consecutive digits in the file name should be considered as a single number. So, a file name is in fact a sequence of letters and numbers.
3. Two unequal file names are compared in two phases. Phase 2 is used only if the order of the two file names could not be distinguished during Phase 1.
4. Phase 1 (soft comparison): The file names are compared lexicographically based on the following rules:
    a. Numbers precede letters ($a1 < aa$).
    b. Numbers with lower values precede numbers with higher values ($a2 < a10$).
    c. Numbers with the same value are not distinguished in this phase.
    d. The letters are compared case-insensitively (in this phase only).
5. Phase 2 (exact/strict comparison): The file names are compared lexicographically based on the following rules:
    a. Numbers with the same value (but with different sequence of digits) are compared lexicographically ($01 < 1 < 02 < 2 < 10$).
    b. Lower case of each letter precedes its upper case form ($a < A < b < B$).

Now, you have to write the “compare” method of our desired sorting algorithm.

Input (Standard Input)

Each test case consists of two lines. The first string and the second string appear on the first line and the second line, respectively. Both strings are strings of at most 255 alphanumeric characters. The input terminates with "###" which should not be processed.

Output (Standard Output)

For each test case output ‘<’, ‘=’ or ‘>’ (omit the quotes) in one line as described in the following.

- ‘<’: if the first string precedes the second one (in our desired sorting)
- ‘=’: if the two strings are exactly the same
- ‘>’: if the first string succeeds the second one (in our desired sorting)

Sample Input and Output

Standard Input	Standard Output
A A A b A a 1 b Abc aBD Qwerty10 Qwerty10 100 100 010 2 010 10 A10 a2 a aa A aa 10c03 10b3 1a2b3d 01a002b3d0 ###	= < > < < = = > < > < < > <

Problem C: RNA Molecules

The RNA molecule is a sequence of four nucleotides A, C, G, and U. Due to the chemical structure of nucleotides, there could be hydrogen bonds established between any pair of (A, U), (U, A), (G, C) or (C, G) nucleotides, but not for the other cases (i.e. for instance A can't make a hydrogen bond with either C or G). Every nucleotide can have hydrogen bond with at most one other nucleotide, and there could not be any hydrogen bond between two nucleotides that are adjacent in the RNA sequence as there is already a covalent interaction between them.

In the living cells, RNA will fold and there would be many hydrogen bonds established. But the hydrogen bonds can't cross each other, i.e. if there is a bond from nucleotide i to nucleotide j ($i < j$), the nucleotides $i + 1, \ldots, j - 1$ can have bonds only between themselves, and can't have any bonds to either $1, \ldots, i - 1$ or $j + 1, \ldots, n$ where n is the number of nucleotides in the RNA.

The real case for RNA is more complicated, but we are not going to face you so much complexities in this competition, making your life easier.

Your task is to find the maximum possible bonds for a set of given RNA molecules.

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing a non-negative integers $0 \leq n \leq 500$. The second line of each test case contains a string (RNA molecule) of size n consisting of characters A, C, G, and U. The input terminates with "0" which should not be processed.

Output (Standard Output)

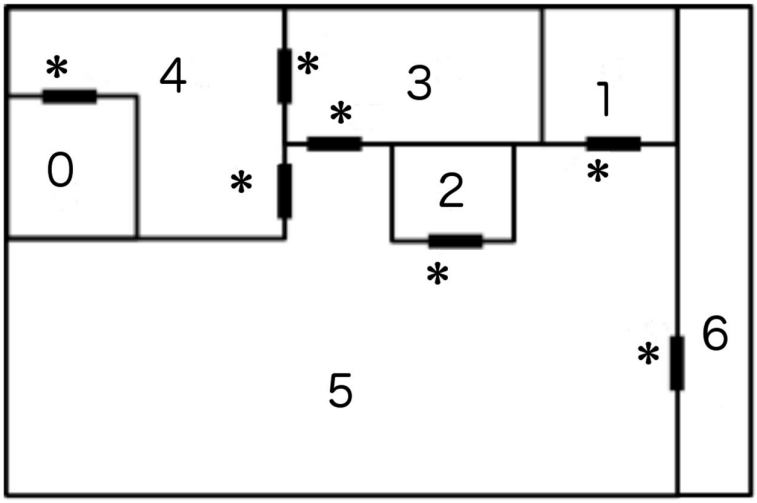
For each test case, output the maximum number of hydrogen bonds for the given RNA molecule.

Sample Input and Output

Standard Input	Standard Output
2 AU 4 AGCU 4 AGUC 0	0 1 1

Problem D: Pirates

A ship has been attacked by pirates. The ship is within the distance of about 30 minutes from the nearest navy, who are approaching to rescue them. The ship's crews have gathered in the safe zone of the ship. The captain wants to secure the safe zone in order to prevent the pirates from taking them as hostages, which makes the situation more complicated for the rescuers. The ship has many zones connected by some doors. Each door can be locked and unlocked from just one side of the door. The pirates have occupied some of the ship's zones, and the captain wants to lock the minimum number of doors in order to prevent the pirates entering the safe zone. For example, in the following setting, a ship with 7 zones is depicted:



Rooms are numbered from 0 through 6, and a '*' is shown in one side of each door denoting the door can be locked/unlocked from that side. If the crews are in Zone 4 and all pirates are in Zone 5, to protect the crews, the captain needs to lock at least two doors, namely the door between Zones 3 and 5, and the door between Zones 4 and 5. Note that locking the door between Zones 3 and 4 instead of the door between Zones 3 and 5 is a big mistake, as the pirates can easily unlock the door between Zones 3 and 4 and enter the safe zone.

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing two non-negative integers z and s ($0 \leq z \leq 20$, $0 \leq s \leq 19$), where z indicates the number of zones in the ship and s is the number of the safe zone. Suppose zones are numbered from 0 through $z-1$. On the following z lines, the i^{th} line represents the information for Zone $i-1$. Each line starts with either 'P' or 'NP, where 'P' means the zone has been occupied by some pirates, and 'NP' means the zone is still clean. After 'P'/'NP', the number of doors which are lock/unlock-able from inside Zone $i-1$ appears. The line is followed with the list of reachable zones from these doors. For instance, in the above example, we don't list Zones 3 and 4 in the line representing Zone 5, because the doors being adjacent to Zones 3 and 4 are not lock/unlock-able from inside zone 5. The output terminates with "0 0" which should not be processed.

Output (Standard Output)

For each test case, output the minimum number of doors which must be looked in order to protect the safe zone. If it is impossible to secure the safe zone, just output 'Sorry Captain!'. Assume that all doors are open at the beginning, and there is not any pirate in the safe zone.

Sample Input and Output

Standard Input	Standard Output
7 4 NP 0 NP 0 NP 0 NP 2 5 4 NP 2 5 0 P 3 6 2 1 NP 0 7 4 NP 0 NP 0 NP 0 P 2 5 4 NP 2 5 0 NP 3 6 2 1 P 0 4 0 NP 4 2 2 1 1 NP 1 3 NP 1 1 P 0 0 0	2 Sorry Captain! 1

Problem E: Shuffling Strings

Suppose S_1 and S_2 are two strings of size n consisting of characters A through H (capital letters). We plan to perform the following step several times to produce a given string S. In each step we shuffle S_1 and S_2 to get string S_{12}. Indeed, S_{12} is obtained by scanning S_1 and S_2 from left to right and putting their characters alternatively in S_{12} from left to right. The shuffling operation always starts with the leftmost character of S_2. After this operation, we set S_1 and S_2 to be the first half and the second half of S_{12}, respectively. For instance, if $S_1 = ABCHAD$ and $S_2 = DEFDAC$, then $S_{12} = DAEBFCDHAACD$, and for the next step $S_1 = DAEBFC$ and $S_2 = DHAACD$. For the given string S of size $2n$, the goal is to determine whether $S_{12} = S$ at some step.

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing a non-negative integers $0 \leq n \leq 100$ which is the length of S_1 and S_2. The remainder of each test case consists of three lines. The first and the second lines contain strings S_1 and S_2 with size n, respectively, and the last line contains string S with size $2n$. The input terminates with "0" which should not be processed.

Output (Standard Output)

For each test case, output -1 if S is not reachable. Otherwise, output the minimum number of steps to reach S. To make your life easier, we inform you that **the output is not greater than 50 for the given input.**

Sample Input and Output

Standard Input	Standard Output
4 AHAH HAHA HHAAAAHH 3 CDE CDE EEDDCC 0	2 -1

Problem F: Guess Number

My tech-lover son has been attracted by the exciting game "Guess Number". In this game, the player must find a hidden positive integer number by at most T guesses (or turns). The parameter T together with a health parameter H is determined at the beginning of the game. In each turn, the player must enter a number. If the number is equal to the hidden number, he wins provided that $H \geq 0$. If the number is bigger than the hidden number, H is decreased by 1 unit of health. Otherwise, H remains unchanged. When H becomes negative or T reaches 0, the player definitely loses. The player can see the remaining turns and units of health after each turn. Although the game seems to be constructive, but something makes me suspicious! my son always wins. He claims he has a searching algorithm to find the hidden number but I can't believe him as for the given H and T there must be a big number which is not guessable by any searching algorithm. To prove my son claim is wrong, I kindly ask you to help me find the smallest M for which at least a number from 1 through M as the hidden number can't be guessed for the given T and H. For example, there is not any algorithm for finding all positive integers not greater than $M = 3$ by 2 turns and 0 units of health.

Input (Standard Input)

There are multiple test cases. Each test case consists of one line containing two non-negative integers T and H ($0 \leq T, H \leq 100$). The input terminates with "0 0" which should not be processed.

Output (Standard Output)

For each test case output M described above in one line. As M maybe too large, output it modulo 10^9+7.

Sample Input and Output

Standard Input	Standard Output
3 0 3 1 0 0	4 7

Problem G: Subset Sum

For a given set X of n not-necessarily-distinct numbers and a given number t, the goal is to compute the number of non-empty subsets Y of X with the properties that the sum over all members of Y is at most t and adding any member in $X - Y$ to Y makes the summation greater than t. Note that the numbers in the set may have the same values, but they must be considered inherently different.

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing two non-negative integers $0 \leq n \leq 30$ and $0 \leq t \leq 1000$. The remainder of each test case consists of one or more lines containing n non-negative numbers belonging to X. The input terminates with "0 0" which should not be processed.

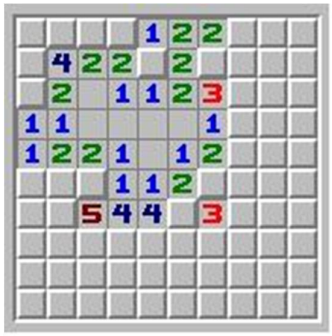
Output (Standard Output)

For each test case, output the number of subsets defined above.

Sample Input and Output

Standard Input	Standard Output
6 25 8 9 8 7 16 5 30 250 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 0 0	15 16509438

Problem H: Minesweeper



It is not surprising that you have played the Minesweeper game at least once. In this game, the player is initially presented with a grid (minefield) of squares. Some randomly selected squares, unknown to the player, are designated to contain mines. A square is unsafe if it contains a mine. Otherwise, it is called safe. The game is played by revealing the squares of the grid, typically by mouse clicks. If an unsafe square is revealed, the player loses the game. Otherwise, a digit is revealed in the safe square, indicating the number of adjacent squares (out of the possible eight) that are unsafe. In typical implementation, if this number is zero, the square appears blank, and the surrounding squares are automatically also revealed. In a variant of this game, the number of unsafe squares is given at the beginning of the game but in our variant this number is hidden. The player wins if he can reveal all safe squares. For a given minefield, your job is to find out whether the minefield can be deterministically cleaned with starting from the given start square, i.e. until all safe squares are not revealed, at each step we can find more safe squares just using the numbers appearing on revealed squares (not by clicking squares randomly.)

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing two non-negative integers n and m not exceeding 10 where n and m are the number of rows and columns of the minefield. Each of the next n lines contains a string of size m consisting of {'+', '*', '-'}. Character '-' denotes a blank square, '*' denotes an unsafe square and '+' denotes the starting square which by default is safe. There is exactly one '+' in each test case. The input terminates with "0 0" which should not be processed.

Output (Standard Output)

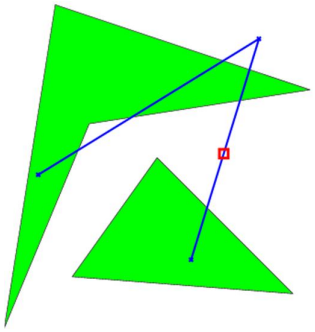
For each test case, output "Yes" if the given minefield can be deterministically cleaned. Otherwise, output "No".

Sample Input and Output

Standard Input	Standard Output
1 3 -+* 3 4 *--+ **-- *--* 3 4 ***- *--- *--+ 0 0	No No Yes

Problem I: Tunb Airline

Tunb airline has several flight plans in the Persian Gulf in which there are several beautiful islands. Each flight path looks like a polygonal path on a 2D map starting at one island and ending at another island as depicted in the figure. Recently, the airline has decided to determine how much safe its flights are. For a flight plan, one of the safety measures to be determined is to find out the maximum of dangerousness(p) over all points p on the flight path where dangerousness(p) is the minimum Euclidean distance of p to islands located in the Persian Gulf. If this measure is low, the chance of surviving is high if an incident happens. Your job is to compute this safety measure for a given flight plan.



Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing two non-negative integers n and m $0 \leq n,\ m \leq 20$ where n is the number of islands in the Persian Gulf and m is the number of vertices of the flight path. Then it is followed by m lines each containing two coordinates of the path vertices from first to last. Finally each test case is terminated with the description of islands which are disjoint polygons. For each polygon, first the number of vertices, t, appears in a line (t is at least 3 and at most 30). In the next t lines, the coordinates of the vertices of the polygon is given in either clockwise or counter-clockwise order; each line containing two integers. All coordinates are within range -10,000 and 10,000. The input finishes with a line containing "0 0" (omit the quotes) which should not be processed as a test case.

Output (Standard Output)

For each test case, output the safety measure defined above rounded to exactly three digits after the decimal point.

Sample Input and Output

Standard Input	Standard Output
1 2 -9 -6 5 1 3 0 16 -16 -12 17 -6 2 3 12 4 16 17 3 9 4 1 0 4 19 19 14 6 12 3 10 10 5 3 18 2 0 0	0.000 2.943