COCI 2009/2010 **Task KAJAK**

Contest #6 - March 20, 2010 1 second / 32 MB / 30 points

Mirko and Slavko are sports commentators on a local kayaking competition. They have a live satellite feed of the entire track. Since there are too much teams for our dynamic duo to follow, they asked you to help them. They would like you to write a program that will display current standings team-by-team.

The satellite feed is encoded as a table of **R** rows **C** characters each. The first character in each row is the starting line, encoded by the character 'S'. The last character in each row is the finishing line, encoded by 'F'. There are **exactly nine** kayaks on the image. Each kayak is marked by his number, and each spans **exactly three consecutive columns**. Water is marked by '.'.

Teams are ranked by their distance to the finish line. Smaller is better. If two teams are at the same distance, they share their place.

INPUT

The first line of input contains two integers **R** and **C** ($10 \leq$ **R**, **C** ≤ 50), the number of rows and columns of the encoded satellite image.
Each of the following **R** lines contains exactly **S** characters '.', 'S', 'F' and 'digits 1' to '9'. **Each row will contain at most one kayak.**
Each image contains all 9 kayaks.

OUTPUT

Output nine lines, one for each kayak. The i^{th} line should contain the current rank of the i^{th} team.

SAMPLE TEST CASES

Input:	**Input:**
10 10 S.....111F S....222.F S...333..F S..444...F S.555....F S666.....F S.777....F S..888...F S...999..F S........F	10 15 S..........222F S.....111.....F S...333.......F S...555.......F S.......444...F S.............F S......777....F S..888........F S........999..F S...666.......F
Output:	**Output:**
1 2 3 4 5 6 5 4 3	5 1 6 3 6 6 4 7 2

COCI 2009/2010 **Task NATJECANJE**

Contest #6 - March 20, 2010 1 second / 32 MB / 50 points

As you know, a kayaking competition is going on as we speak. Unfortunately strong winds have damaged a few kayaks, and the race starts in 5 minutes!. Fortunately, some teams have brought reserve kayaks. Since kayaks are bulky and hard to carry, teams are willing to lend kayaks to opposing teams if and only if they are starting immediately next to them. For example, team with the starting number 4 will lend its reserve kayak only to teams 3 and 5. Of course if some team did bring a reserve and **its**' kayak was damaged, they will use it themselves and not lend it to anyone.

You as the organizer now need to know, what is the minimal number of teams that cannot start the race, not even in borrowed kayaks.

INPUT

The first line of input contains three integers **N**, ($2 \leq$ **N** ≤ 10), total number of teams, **S**, ($2 \leq$ **S** $\leq$ **N**), number of teams with damaged kayaks and **R**, ($2 \leq$ **R** $\leq$ **N**), number fo teams with reserve kayaks.

The second line contains exactly **S** numbers, the starting numbers of teams with damaged kayaks. **The second line will not contain duplicates.**

The third line contains exactly R numbers, the starting numbers of teams with reserve kayaks. **The third line will not contain duplicates.**

OUTPUT

The first and only line of output should contain the smallest number of teams that cannot start the competition.

SAMPLE TEST CASES

Input:	**Input:**
5 2 3 2 4 1 3 5	5 2 1 2 4 3
Output:	**Output:**
0	1

COCI 2009/2010 **Task DOSADAN**

Contest #6 - March 20, 2010 1 second / 32 MB / 70 points

Mirko received a message from his friend Slavko. Slavko, being a world class cryptologist, likes to encrypt messages he sends to Mirko. This time, he decided to use One Time Pad encryption. OTP is impenetrable if used correctly, and Slavko knows this. He however, doesn't want Mirko to bang his head on an impossible task, so he sent a few hints along with his message.

Mirko knows that Slavkos original plaintext contained **only** small letters of the English alphabet ('a' - 'z'), full stop '.' and space ' ' (ASCII 32_{10}). Also, he knows that Slavko used only digits '0' to '9' as his key. After much thought, he realized he can determine locations of all spaces and full stops in the plaintext. He now asked you to write a program that will do so automatically.

From his previous dealings with Slavko, Mirko knows how OTP encryption works. Let's look at a simple example. Suppose you want to encode the string "abc efg" using "0120123" as key.

abc efg 0120123	61 62 63 20 65 66 67 30 31 32 30 31 32 33	51 53 51 10 54 54 54
Start	ASCII hexadecimal	encrypted message

First, you transform both the key and plaintext into hexadecimal numbers using ASCII encoding. Then you align them and preform XOR operation on each pair. The resulting sequence is the encrypted message.

INPUT

The first line of input contains one integer **N** ($1 \leq$ **N** ≤ 1000), number of characters in the encrypted message.

Next line contains **N** integers, written in hexadecimal, larger than or equal to 0_{10} and smaller than or equal to 127_{10}, the encrypted message.

OUTPUT

The first and only line of output should contain **N** characters, each representing one character in the plaintext. If the i^{th} character of plaintext is a letter, the i^{th} character of output should be a dash '-', if not, you should output a full stop '.'.

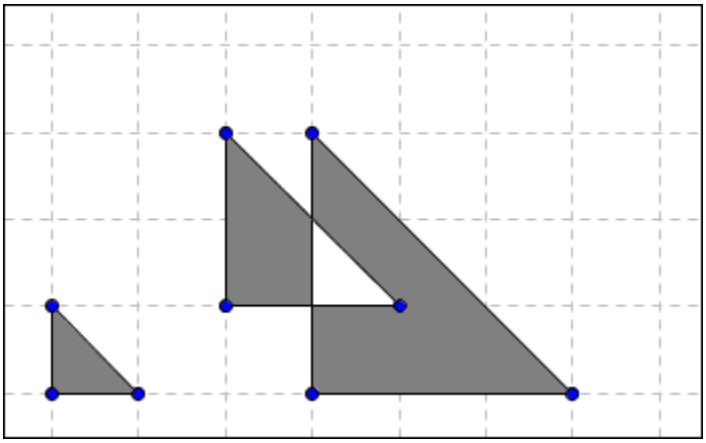
SAMPLE TEST CASES

Input:	**Input:**
7 51 53 51 10 54 54 54	7 53 53 51 54 54 51 10
Output:	**Output:**
---.---	------.

COCI 2009/2010 **Task XOR**

Contest #6 - March 20, 2010. 1 second / 32 MB / 100 points

Mirko and Slavko have built their own LED display. The display is initially white. During each of the **N** parts of the testing phase, Mirko attached three electrodes to the display in such way that they formed a **right isosceles triangle**. He noticed that, after attaching the electrodes, all pixels in the enclosing triangle are **inverted** (white pixels become black, and black pixels become white).



Watching Mirko play with the electrodes, Slavko observed interesting shapes emerging on the screen. Mathematically inclined as he is, first thing that crossed his mind was how to calculate total area covered by black pixels. Help him by writing a program to do just that!

INPUT

First line of input contains an integer **N** ($1 \leq \mathbf{N} \leq 10$), number of triangles formed by Mirko's fiddling with electrodes. Each of the following **N** lines contains three integers **X**, **Y** and **R** ($1 \leq \mathbf{X}, \mathbf{Y}, \mathbf{R} \leq 10^6$), describing a triangle. (**X**, **Y**) are the coordinates of the lower left corner of the triangle, while **R** represents the length of the two sides of the triangle.

OUTPUT

The first and only line of output should contain the area covered by black pixels, rounded to one decimal place.

SAMPLE TEST CASES

Input:	**Input:**	**Input:**
3 1 1 2 7 1 6 5 3 4	5 5 5 99 5 5 99 5 5 99 5 5 99 5 5 99	4 5 5 99 5 5 99 5 5 99 5 5 99
Output:	**Output:**	**Output:**
24.0	4900.5	0.0

COCI 2009/2010 **Task HOLMES**

Contest #6 - March 20, 2010 1 second / 32 MB / 120 points

Sherlock Holmes is a renowned detective. His Scotland Yard colleagues often provide him with a set of evidence and ask for his help in solving the mysteries. After many years of practice, Holmes has gained an enormous amount of experience and already knows causes for many common events, which, combined with his extraordinary deductive capabilities, enables him to solve cases in a matter of minutes, from the comfort of his chair.

Holmes' knowledge base can be modeled as a set of implications of the form $A \to B$ (where A and B represent events), which means that, if A occurred, event B must have also occurred (remember that logical implication is **false** only if A is true and B is false). Of course, implications can form chains of reasoning, e.g. $A \to B \to C$. However, there will **never** be a **circular** chain of implications (e.g. $A \to B \to C \to \cdots \to A$).

Holmes is given a set S = {S_1, S_2, S_3, ..., S_n} of events that are known to have occurred. He can then, using his extensive knowledge and deductive powers, find **all events** that have **certainly** occurred.

It's important to note that Holmes' knowledge is so mind-bogglingly huge that he knows **all possible causes of events**. In other words, there is no implication that is true, but not included in Holmes' knowledge base.

Many detective agencies would highly appreciate Holmes' one of a kind capabilities, so you were given a task to accomplish with a computer what is out of reach for ordinary mortals. Write a program to find all events that have certainly occurred based on the given implications and evidence collected by your colleague detectives.

INPUT

First line of input consists of three integers, **D** (1 ≤ **D** ≤ 1000), the number of different types of events, **M** (1 ≤ **M** ≤ 100000), the number of implications, and **N** (1 ≤ **N** ≤ **D**), the number of evidence collected by the detectives.

Each of the **M** lines that follow contains two integers **A** and **B** (1 ≤ **A**, **B** ≤ **D**), describing an implication $A \to B$.

Finally, each of the last **N** lines contain an integer **X** (1 ≤ **X** ≤ **D**) describing

an event that must have occurred, based on the evidence collected by detectives.

OUTPUT

The first and only line of output should contain integers (in any order) representing events that have certainly occurred.

SAMPLE TEST CASES

Input:	**Input:**	**Input:**
3 2 1 1 2 2 3 2	3 2 1 1 3 2 3 3	4 4 1 1 2 1 3 2 4 3 4 4
Output:	**Output:**	**Output:**
1 2 3	3	1 2 3 4

Explanation of the second sample case:
The knowledge base contains implications 1 -> 3 and 2 -> 3. Therefore, Holmes knows that event 3 can be caused only by events 1 and 2, but (without any extra information), he can't be certain **which one** of those events actually caused 3. As a result, only event that must have occurred is the one given in the input.

Explanation of the third sample case:
Holmes doesn't know which event from the set {2, 3} is directly responsible for event 4. However, as both of those events are caused **only** by event 1, Holmes can deduce that event 1 **must have occurred**. As a consequence, events 2, 3 and 4 (given by the detectives) have also occurred.

COCI 2009/2010 **Task GREMLINI**

Contest #6 - March 20, 2010 1 second / 32 MB / 130 points

Gremlins are small, funny, fuzzy creatures. In times long gone they used to cause quite a ruckus, but now most of them live decent, honest family lives.

There are **N** different types of gremlins, coded with numbers 1 to **N** for your convenience. Legend has it that **T** years ago a laboratory accident occurred causing **N** gremlins, one of each type to be born.

As you all know, in order for gremlins to reproduce, no mating rituals are required. Just add a dash of watter and you instantly get a few new cocoons.

Gremlins of type **i** need exactly $\mathbf{Y_i}$ years to mature and spawn $\mathbf{K_i}$ cocoons. For each cocoon you know how much years does it take to hatch, and which gremlin type is contained within. The original gremlin unfortunately dies while producing cocoons.

Now, **T** years after the original accident. Scientist wonder what is the longest ancestral chain whose genome is currently represented. They are only interested in ancestors of hatched gremlins, not those still cocooned. You are safe to assume that all gremlins that were supposed to hatch this year did so.

INPUT

The first line of input contains two integers **N** and **T** ($1 \leq \mathbf{N} \leq 100$, $1 \leq \mathbf{T} \leq 10^{15}$), number of gremlin types and number of years after the original accident.

Next $3 \cdot \mathbf{N}$ lines contain gremlin type descriptions, three lines per type:

- First line contains integers $\mathbf{K_i}$ and $\mathbf{Y_i}$ ($1 \leq \mathbf{K_i} \leq 1000$, $1 \leq \mathbf{Y_i} \leq 1000$), number of cocoons formed when this gremlin type spawns and number of years this gremlin type needs to reach maturity.
- The second line contains $\mathbf{K_i}$ integers between 1 and **N** inclusive, types of gremlins hatched from cocoons formed by this gremlin type.
- The third line contains $\mathbf{K_i}$ integers between 1 and 1000, representing hatching time for each cocoon, in years.

OUTPUT

The first and only line of output should contain the length of the currently longest ancestral chain.

SAMPLE TEST CASES

Input:	**Input:**	**Input:**
1 42 1 10 1 5	2 42 1 10 1 5 1 5 1 5	3 8 4 5 1 2 3 2 1 2 1 3 1 1 3 1 2 1 1 2 2 1
Output: 2	**Output:** 3	**Output:** 4

Second sample case description:

The original gremlin hatched in the accident after 10 years spawns a single cocoon and dies.

15 years after the accident, a new gremlin hatches from the cocoon. His ancestral chain contains one gremlin. 25 years after the accident this gremlin spawns a new cocoon and dies.

30 years after the accident, a new gremlin hatches from the cocoon. His ancestral chain contains two gremlins. 40 years after the accident this gremlin spawns a new cocoon and dies.

42 years after the accident, the current cocoon is not yet hatched, so the longest ancestral chain ever recorded is two gremlins in length.

COCI 2009/2010 **Task SPAVANAC**

7th round, 24. April 2010. 1 second / 32 MB / 30 points

Every school morning Mirko is woken up by the sound of his alarm clock. Since he is a bit forgetful, quite often he leaves the alarm on on Saturday morning too. That's not too bad tough, since he feels good when he realizes he doesn't have to get up from his warm and cozy bed.

He likes that so much, that he would like to experience that on other days of the week too! His friend Slavko offered this simple solution: set his alarm clock 45 minutes early, and he can enjoy the comfort of his bed, fully awake, for 45 minutes each day.

Mirko decided to heed his advice, however his alarm clock uses 24-hour notation and he has issues with adjusting the time. Help Mirko and write a program that will take one time stamp, in 24-hour notation, and print out a new time stamp, 45 minutes earlier, also in 24-hour notation.

Note: if you are unfamiliar with 24-hour time notation yourself, you might be interested to know it starts with 0:00 (midnight) and ends with 23:59 (one minute before midnight).

INPUT

The first and only line of input will contain exactly two integers **H** and **M** ($0 \leq$ **H** ≤ 23, $0 \leq$ **M** ≤ 59) separated by a single space, the input time in 24-hour notation. **H** denotes hours and **M** minutes.

OUTPUT

The first and only line of output should contain exactly two integers, the time 45 minutes before input time.

SAMPLE TEST CASES

Input:	**Input:**	**Input:**
10 10	0 30	23 40
Output:	**Output:**	**Output:**
9 25	23 45	22 55

COCI 2009/2010 **Task COKOLADA**

7th round, 24. April 2010. 1 second / 32 MB / 50 points

A new type of chocolate arrived in the local shop. The chocolate comes in bars, each bar consisting of **N** squares. Bars are factory made and only come in sizes which are full powers of two. In other words a single bar has 1, 2, 4, 8, 16, ... squares.

To fully asses the quality of chocolate Mirko must sample at least **K** squares. His friend Slavko would also like to try some of the chocolate. Since Mirko is in a hurry to try the chocolate himself, he decides to break the bar he bought in pieces, such that he has **exactly K** squares, and leaves the rest (if any) to Slavko. The bars are a bit brittle, so Mirko can break them only on their exact center. In other words, from one bar with **D** squares, he can get two bars with **D**/2 squares.

Write a program that will determine the **minimal number of breaks** Mirko must perform in order to obtain exactly K **squares** (not necessarily in one piece). Also, determine the smallest bar size Mirko must buy in order to have at least **K** squares.

INPUT

The first and only line of input will contain one integer **K** ($1 \leq K \leq$ **1 000 000**) , number of squares Mirko must sample.

OUTPUT

The first and only line of output should contain two integers, separated by a single space. The first integer is the smallest bar size Mirko must buy. The second the smallest number of breaks.

SAMPLE TEST CASES

Input: 6	**Input:** 7	**Input:** 5
Output: 8 2	**Output:** 8 3	**Output:** 8 3

COCI 2009/2010 **Task BAKICE**

7th round, 24. April 2010. 1 second / 32 MB / 70 points

When it comes to trams, a lot of people are civilized individuals who know how to behave in one. However, there are always those few who upon spotting a place to sit will run towards it in supersonic speeds. If they spot more than one place, they always try the **closest one first**.

Problems arise when two or more such individuals aim for the same spot. If one of them is the **closest**, he or she will sit, and others won't event attempt to move in and instead turn their attention to the **next closest spo**t. If however they are **all equally close**, they will all run to the seat resulting in a massive explosion that usually ends with complete **destruction of both them and the seat**.

You are given a description of one tram. It is represented as a table with **R** rows and **C** columns. The rude passengers are marked with the letter 'X'. Empty seats are marked with 'L' and the tram floor is marked with '.'. Note that although there are other passengers, the force projected by these idiots is more than enough to simply walk through them.

Distance between two cells is the Euclid distance between their centers. Write a program that will determine the number of explosions which will take place before all people are seated, or destroyed, or they run out of chairs.

INPUT

The first line of input contains two integers, **R** ($1 \leq \mathbf{R} \leq 100$) and **C** ($1 \leq \mathbf{C} \leq 100$), number of rows and columns.

The next **R** lines contain **C** characters each. '.', 'X' or 'L'.

There will always be at least one character 'X' and at least one 'L' in the input. **Also, there will be no two 'L' characters such that they are both equally distant to some 'X'.**

OUTPUT

The first and only line of input should contain the number of explosion for the given layout.

SAMPLE TEST CASES

Input:	**Input:**	**Input:**
4 4 .LX. .X.. .... .L..	4 4 .XLX .X.. ...L .X..	7 7 ...X.X. XL....L ....L... .....XL ....X...
Output:	**Output:**	**Output:**
1	2	1

COCI 2009/2010 **Task SVEMIR**

7th round, 24. April 2010. 2 sekunde / 32 MB / 100 points

Fourth Great and Bountiful Human Empire is developing a transconduit tunnel network connecting all it's planets. The Empire consists of **N** planets, represented as points in the 3D space. The cost of forming a transconduit tunnel between planets A and B is:

$$TunnelCost[A,B] = min\{ |x_A-x_B|, |y_A-y_B|, |z_A-z_B| \}$$

where (x_A, y_A, z_A) are 3D coordinates of planet A, and (x_B, y_B, z_B) are coordinates of planet B. The Empire needs to build exactly **N** - 1 tunnels in order to fully connect all planets, either by direct links or by chain of links. You need to come up with the lowest possible cost of successfully completing this project.

INPUT

The first line of input contains one integer **N** ($1 \leq$ **N** $\leq 100\,000$), number of planets.

Next **N** lines contain exactly 3 integers each. All integers are between -10^9 and 10^9 inclusive. Each line contains the x, y, and z coordinate of one planet (in order).

No two planets will occupy the exact same point in space.

OUTPUT

The first and only line of output should contain the minimal cost of forming the network of tunnels.

SAMPLE TEST CASES

Input:	**Input:**	**Input:**
2	3 -1 -1 -1	5 11 -15 -15

1 5 10 7 8 2	5 5 5 10 10 10	14 -5 -15 -1 -1 -5 10 -4 -1 19 -4 19
Output: 3	**Output:** 11	**Output:** 4

COCI 2009/2010 **Task KRALJEVI**

7th round, 24. April 2010. 1 second / 64 MB / 120 points

Mirko and Slavko are playing a chess like game. The game is played on a non-standard chess board sized **R** rows by **C** columns. Each player starts with some number of chess kings. In chess kings can move from their current field to any of the 8 neighbouring fields.

Player spread is defined as the complete **sum of distances between all pairs** of pieces of the given player. The distance between two pieces is the smallest number of moves required for both pieces to reach the same field. No actual moves are performed when calculating the distance and as such enemy pieces do not influence the result.

Mirko knows that the spread is a vital piece of strategic information and would like you to make him a program that will calculate both his and Slavko's spread.

INPUT

The first line of input contains two integers **R** and **C** ($1 \leq$ **R**, **C** $\leq 1\,000$), number of rows and columns.

Next **R** lines contain **C** characters each. Character '**M**' denotes Mirko's piece, '**S**' Slavko's piece and '**.**' denotes an empty field.

There is at least one piece per player on the board. Otherwise the game would be over.

OUTPUT

In the first and only line of output you need to print exactly two integers. The first integers is the spread of Mirko's and the second Slavko's pieces.

SCORING

In test cases worth **20%** of total points the number of pieces on the board will be less then or equal to 5000.
In test cases worth **60%** of total points, the numbers **R** and **C** will be smaller than or equal to 300.

SAMPLE TEST CASES

Input:	**Input:**	**Input:**
2 3 SMS MMS	2 3 S.M M..	4 5 M.... ..S.M SS..S .M...
Output:	**Output:**	**Output:**
3 5	2 0	10 13

COCI 2009/2010 **Task RESTORAN**

7th round, 24. April 2010. 1 second / 128 MB / 130 points

In Croatia there are **N** cities connected by **E** twoway roads. Two large food chains have recently reached an agreement on market sharing. In the middle of each road, exactly one chain will be given rights to build a restaurant.

To ensure the market is shared fairly, each city must have at **least one restaurant from each chain** on the roads connected to that city. However, there are cities with only one road, or no roads at all, and for them it is impossible to have both chains. Such cities are doomed to visit one chain, or travel a bit further.

Write a program that will determine for each road the chain that should build there so that these requirements are met.

INPUT

The first line of input contains tow integers **N** and **E** ($1 \leq \mathbf{N}, \mathbf{E} \leq 100\,000$), number of cities and number of roads.

The next **E** lines contain two integers each. Each line describes one road. Integers $\mathbf{A_i}$ and $\mathbf{B_i}$ ($1 \leq \mathbf{A_i}, \mathbf{B_i} \leq \mathbf{N}$; $\mathbf{A_i} \neq \mathbf{B_i}$) denote a road connecting cities $\mathbf{A_i}$ and **Bi**

There will never be two or more roads connecting the same cities.

OUTPUT

If there is no way to fairly assign the roads, the first and only line of input should contain "0".

Otherwise output exactly E lines, one for each road, in the same order as they were given in the input. The i^{th} line should contain "1" if the first chain has the right to build onthis road, or "2" if the second one does.

Note: if the solution is not unique, you may output any valid one.

SCORING

This taks has test cases grouped into test runs. Each test run consists of one or more test cases. In order to score points for the test run, all test cases in the run must be solved correctly.
You **do not** need to worry about test runs during input and output. The system does that for you. Follow the format described in INPUT/OUTPUT to the letter!
Test runs worth **60**% have **N** ≤ 1000, **E** ≤ 5000.

SAMPLE TEST CASES

Input:	**Input:**	**Input:**
5 6 1 2 2 3 3 1 3 4 1 4 4 5	7 7 1 2 2 3 3 1 4 5 5 6 6 7 7 4	77777 4 1 2 1 3 1 4 1 5
Output:	**Output:**	**Output:**
1 2 1 2 2 1	0	1 2 2 2