CROATIAN OPEN COMPETITION IN INFORMATICS

6th ROUND

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 '.'.

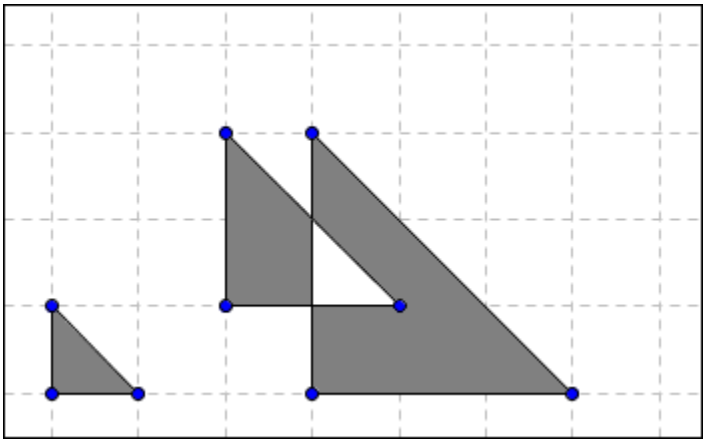
SAMPLE TEST CASES

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.