ACM International Collegiate Programming Contest

2011

Latin American Regional Contests

November 4th-5th, 2011

Contest Session

This problem set contains 11 problems; pages are numbered from 1 to 20.

This problem set is used in simultaneous contests hosted in the following countries:

- Argentina
- Bolivia
- Brazil
- Chile
- Colombia
- Cuba
- Peru
- Mexico
- Venezuela

General Information

Unless otherwise stated, the following conditions hold for all problems.

Input

1. The input must be read from standard input.
2. The input contains several test cases. Each test case is described using a number of lines that depends on the problem.
3. When a line of data contains several values, they are separated by *single* spaces. No other spaces appear in the input. There are no empty lines.
4. Every line, including the last one, has the usual end-of-line mark.
5. The end of input is indicated with a line containing certain values that depend on the problem. This line should not be processed as a test case.

Output

1. The output must be written to standard output.
2. The result of each test case must appear in the output using a number of lines that depends on the problem.
3. When a line of results contains several values, they must be separated by *single* spaces. No other spaces should appear in the output. There should be no empty lines.
4. Every line, including the last one, must have the usual end-of-line mark.
5. No special mark should be written to indicate the end of output.

Problem A

Army buddies

Problem code name: `army`

Nlogonia is fighting a ruthless war against the neighboring country of Cubiconia. The Chief General of Nlogonia's Army decided to attack the enemy with a linear formation of soldiers, that would advance together until conquering the neighboring country. Before the battle, the Chief General ordered that each soldier in the attack line, besides protecting himself and attacking, should also protect his two (nearest) neighbors in the line, one to his left and one to his right. The Chief General told the soldiers that for each of them, his "buddies" would be these two neighbors, if such neighbors existed (because the leftmost soldier does not have a left neighbor and the rightmost soldier does not have a right neighbor). The Chief General also told the soldiers that protecting their buddies was very important to prevent the attack line from being broken. So important that, if the left or right buddy of a soldier is killed, then the next living neighbor to the left or to the right of the soldier, respectively, should become his buddy.

The battle is fierce, and many soldiers in the attack line are being killed by fire shots, grenades and bombs. But following the Chief General's orders, immediately after knowing about losses in the attack line, the Army's information systems division has to inform the soldiers who their new buddies are.

You are given the number of soldiers in the attack line, and a sequence of loss reports. Each loss report describes a group of contiguous soldiers in the attack line that were just killed in the battle. Write a program that, for each loss report, prints the new buddies formed.

Input

Each test case is described using several lines. The first input line contains two integers S and B representing respectively the number of soldiers in the attack line, and the number of loss reports ($1 \leq B \leq S \leq 10^5$). Soldiers are identified by different integers from 1 to S, according to their positions in the attack line, being 1 the leftmost soldier and S the rightmost soldier. Each of the next B input lines describes a loss report using two integers L (left) and R (right), meaning that soldiers from L to R were killed ($1 \leq L \leq R \leq S$). You may assume that until that moment those soldiers were alive and were just killed.

The last test case is followed by a line containing two zeros.

Output

For each test case output $B+1$ lines. In the i-th output line write the new buddies formed by removing from the attack line the soldiers that were just killed according to the i-th loss report. That is, for the loss report '`L R`', print the first surviving soldier to the left of `L`, and the first surviving soldier to the right of `R`. For each direction, print the character '`*`' (asterisk) if there is no surviving soldier in that direction. Print a line containing a single character '`-`' (hyphen) after each test case.

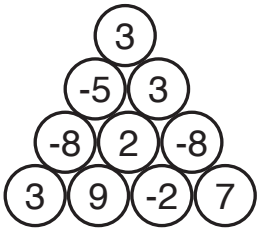
Sample input	**Output for the sample input**
1 1 1 1 10 4 2 5 6 9 1 1 10 10 5 1 1 1 0 0	* * - 1 6 1 10 * 10 * * - * 2 -

Problem B

Ball Stacking

Problem code name: `ball`

The XYZ TV channel is developing a new game show, where a contestant has to make some choices in order to get a prize. The game consists of a triangular stack of balls, each of them having an integer value, as the following example shows.



The contestant must choose which balls he is going to take and his prize is the sum of the values of those balls. However, the contestant can take any given ball only if he also takes the balls directly on top of it. This may require taking additional balls using the same rule. Notice that the contestant may choose not to take any ball, in which case the prize is zero.

The TV show director is concerned about the maximum prize a contestant can make for a given stack. Since he is your boss and he does not know how to answer this question, he assigned this task to you.

Input

Each test case is described using several lines. The first line contains an integer N representing the number of rows of the stack ($1 \leq N \leq 1000$). The i-th of the next N lines contains i integers B_{ij} ($-10^5 \leq B_{ij} \leq 10^5$ for $1 \leq j \leq i \leq N$); the number B_{ij} is the value of the j-th ball in the i-th row of the stack (the first row is the topmost one, and within each row the first ball if the leftmost one).

The last test case is followed by a line containing one zero.

Output

For each test case output a line with an integer representing the maximum prize a contestant can make from the stack.

Sample input	**Output for the sample input**
4 3 -5 3 -8 2 -8 3 9 -2 7 2 -2 1 -10 3 1 -5 3 6 -4 1 0	7 0 6

Problem C

Candy's Candy

Problem code name: `candy`

Candy has a stock of candy of F different flavors. She is going to make several packs of candy to sell them. Each pack must be either a *flavored* pack, containing candy of a single flavor, or a *variety* pack, containing candy of every flavor. Candy wants to make a nice packing with her candy. She decided that a nice packing must honor the following conditions:

- Each piece of candy must be placed in exactly one pack.
- Each pack, regardless of its type, must contain at least 2 pieces of candy.
- Each pack, regardless of its type, must contain the same number of pieces of candy.
- Within each variety pack, the number of pieces of candy of each flavor must be the same.
- There must be at least one variety pack.
- There must be at least one flavored pack of each flavor.

Candy is wondering how many different nice packings of candy she could make. Two nice packings of candy are considered different if and only if they differ in the number of flavored packs, or in the number of variety packs, or in the number of pieces of candy per pack. Since Candy will sell her candy during the closing ceremony of this contest, you are urged to answer her question as soon as you can.

Input

Each test case is described using two lines. The first line contains an integer F indicating the number of flavors ($2 \leq F \leq 10^5$). The second line contains F integers C_i, indicating the number of pieces of candy of each flavor ($1 \leq C_i \leq 10^9$ for $1 \leq i \leq F$).

The last test case is followed by a line containing one zero.

Output

For each test case output a line with an integer representing the number of different nice packings of candy, according to the rules given above.

Sample input	**Output for the sample input**
3 15 33 21 2 1 1 2 2 2 2 3 3 3 1000000000 1000000000 1000000000 0	4 0 0 1 832519396

Problem D

Diccionário Portuñol

Problem code name: `diccionario`

Portuñol is a special language that was naturally developed in Latin America. Since almost half of Latin America speaks Portuguese (Português) and almost half speaks Spanish (Español), the mixing of both languages is natural.

Each word in Portuñol is made by taking a non-empty prefix of a Portuguese word and a non-empty suffix of a Spanish word, and concatenating them together. A *prefix* of a word is any word that can be obtained by erasing zero or more characters from its right end. A *suffix* of a word is any word that can be obtained by erasing zero or more characters from its left end. The name of the language itself comes from taking a prefix of the word "Português" (Portu) and a suffix of the word "Español" (ñol), and concatenating them.

Of course, not every possible way of combining two words will result in something meaningful, or even pronounceable, but that is not important. We want you to write a program to count the number of different Portuñol words.

You will be given two non-empty sets of words to test your program. The first set will represent Portuguese words and the second set will represent Spanish words. You need to calculate the number of different Portuñol words that can be made using the prefix and suffix rule described above. Note that the same word may be constructed in several ways, but it still needs to be counted as one. Also note that the input sets are just to test your program, so they do not need to be made out of actual Portuguese or Spanish words.

Input

Each test case is described using several lines. The first line contains two integers P and S representing respectively the number of Portuguese words and the number of Spanish words ($1 \leq P, S \leq 1000$). Each of the next P lines contains a Portuguese word, and after that each of the next S lines contains a Spanish word. Each word is a non-empty string of at most 1000 characters; each character is one of the 26 standard lowercase letters (from '`a`' to '`z`'). You may assume that within each test case no two Portuguese words are the same, and that the sum of the lengths of all the Portuguese words is at most 10^5. The same holds for the Spanish words.

The last test case is followed by a line containing two zeros.

Output

For each test case output a line with an integer representing the number of different words that can be constructed by concatenating a non-empty prefix of a word in the first set (Portuguese words) and a non-empty suffix of a word in the second set (Spanish words).

Sample input	**Output for the sample input**
3 3 mais grande mundo mas grande mundo 1 5 a aaaaa aaaaaa aaaaaaa a aaaaaaaaa 1 1 abc abc 0 0	182 9 8

Problem E

Electrical Pollution

Problem code name: `electrical`

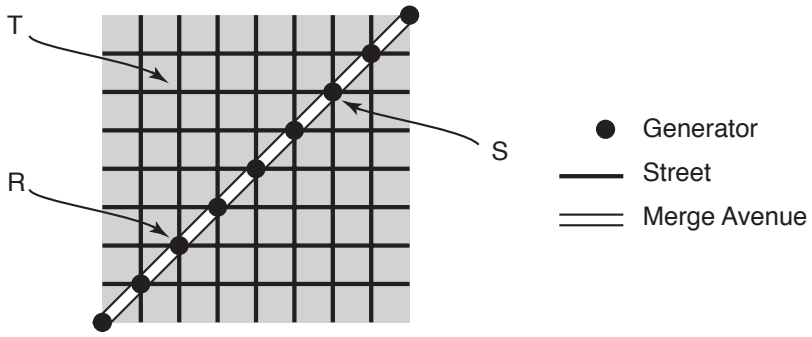
Sortonia is the capital of the North Nlogonia province. The city is laid out with almost all of its streets in a square grid, aligned to either the North-South or the West-East direction. The only exception is Merge Avenue, which runs Southwest-Northeast, splitting city blocks along their diagonals.

Sortonia is also one of the greenest cities in Nlogonia. The local university developed technology to harness the magnetic field of Earth for energy generation. As a consequence, all intersections of Merge Avenue have power generators installed, supplying all the homes and businesses of the city.

This technology was praised by environmentalists at the time for eliminating Sortonia's carbon footprint, but soon after its introduction, thousands of bees and birds were found dead in the city. Puzzled, the Queen of Nlogonia ordered the queendom's biophysicists to investigate the phenomenon.

After many months of study, they discovered that the generators used by Sortonians created anomalies in the local magnetic field. The birds and bees that use the Earth's magnetic field to guide their flight were confused by these anomalies, started flying in circles and eventually died of exhaustion.

According to the biophysicists' theoretical models, each generator creates an anomaly that is represented as an integer value. Each anomaly propagates indefinitely in all four compass directions. Points that are not directly north, south, west or east of the generator are unaffected by it. On the other hand, if a point is aligned with two generators then the anomaly at that point is the sum of the two anomalies produced by those generators. As an example, consider the picture below that represents a certain portion of Sortonia. The anomaly at point R is just the one produced by the generator at that point, while the anomaly at point T is the sum of the anomalies produced by the generator at point R and the generator at point S.



The biophysicists would like to measure the anomalies for some city intersections, but these measurements require expensive equipment and technical expertise. So they plan to measure only a subset of the city's intersections and extrapolate other data from them. Predicting an anomaly from a set of measurements might require combining several of them in complicated ways. Thus, the Queen ordered you to write a program that predicts the anomalies at certain intersections, given the measurements previously made.

Input

Each test case is described using several lines. The first line contains two integers M and Q representing respectively the number of measurements and the number of queries ($1 \leq M, Q \leq 10^4$). Each of the next M lines describes a measurement using three integers X, Y and A, indicating that the measured anomaly at point (X, Y) is A ($-10^7 \leq X, Y \leq 10^7$ and $-10^4 \leq A \leq 10^4$). After that, each of the

next Q lines describes a query using two integers X' and Y', indicating that the anomaly at point (X', Y') must be predicted ($-10^7 \leq X', Y' \leq 10^7$). All positions are measured in city blocks; the first coordinate increases from West to East, while the second coordinate increases from South to North. Point $(0, 0)$ is located on Merge Avenue. You may assume that within each test case each point is not measured more than once. Likewise, each point is not queried more than once. You may also assume that all the measurements are consistent.

The last test case is followed by a line containing two zeros.

Output

For each test case output $Q + 1$ lines. In the i-th line write the answer to the i-th query. If the information given by the measurements is enough to predict the anomaly at the queried point, then write an integer representing the predicted anomaly at the queried point. Otherwise write the character '`*`' (asterisk). Print a line containing a single character '`-`' (hyphen) after each test case.

Sample input	**Output for the sample input**
3 3 30 -10 3 30 20 15 40 20 2 -10 40 40 -10 -10 -10 6 8 0 1 11 0 3 8 1 0 11 3 0 8 4 4 0 3 5 6 1 5 0 3 3 0 4 3 0 2 2 4 4 4 5 5 0 0	-10 -10 * - 9 8 8 * * * 0 * -

Problem F

File Retrieval

Problem code name: `file`

The operating system of your computer indexes the files on your hard disk based on their contents, and provides textual search over them. The content of each file is a non-empty string of lowercase letters. To do a search, you specify a key, which is also a non-empty string of lowercase letters. The result is a list of all the files that contain the key as a substring. A string s is a substring of a string t if t contains all characters of s as a contiguous sequence. For instance, "foofoo", "cafoo", "foota" and "foo" all contain "foo" as a substring, while "foa", "fofo", "fioo" and "oofo" do not.

You know the content of each file on your hard disk, and wonder whether each subset of the files is searchable. A subset of the files is searchable if there exists at least one key that produces exactly the list of those files as a result. Given the contents of the files on your hard disk, you are asked to compute the number of non-empty searchable subsets.

Input

Each test case is described using several lines. The first line contains an integer F representing the number of files on your hard disk ($1 \leq F \leq 60$). Each of the next F lines indicates the content of one of the files. The content of a file is a non-empty string of at most 10^4 characters; each character is one of the 26 standard lowercase letters (from '`a`' to '`z`').

The last test case is followed by a line containing one zero.

Output

For each test case output a line with an integer representing the number of non-empty searchable subsets.

Sample input	**Output for the sample input**
6 form formal malformed for man remake 3 cool cool old 0	11 3

Problem G

Garden Fence

Problem code name: `garden`

Gary is a careful gardener that has a rectangular field full of trees. There are two kinds of trees in his land: pines and larches. To improve their vitality, he decided to start using a specific fertilizer for each kind of tree, instead of the generic fertilizer he was using so far.

Since Gary has many trees, fertilizers cannot be placed individually on each tree. For this reason he decided to build a fence to separate the field in two, and use the pine fertilizer on one side and the larch fertilizer on the other side. The new fence will be built over a straight line connecting two distinct points located on the boundary of the land.

Sadly, each fertilizer is great for the kind of tree it is intended, but deadly for the other. After building the fence and deciding which fertilizer will be used on each side, larches in pines' side and pines in larches' side will be cut down, to prevent a slow death that will ruin the landscape. Furthermore, before building the fence it is necessary to cut down trees of any kind lying directly over the line where the fence will be located.

Of course, Gary loves his trees. Depending on their kind, age and other factors, each tree has a certain value. The gardener wants to build the fence and select where to use each fertilizer in such a way that his loss is minimized, where the loss is the sum of the values of the trees that will be cut down.

You were hired to build the fence. Before starting your work, tell Gary how much he will lose when choosing optimally the location of the fence and the fertilizer for each side.

Input

Each test case is described using several lines. The first line contains two integers P and L, representing respectively the number of pines and the number of larches ($1 \leq P, L \leq 1000$). Each of the next P lines describes a pine. After that, each of the next L lines describes a larch. Trees are modeled as points in the XY plane. Each tree is described using three integers X, Y and V, where X and Y are the coordinates of the tree ($-10^5 \leq X, Y \leq 10^5$), and V is its value ($1 \leq V \leq 1000$). You may assume that within each test case no two trees have the same location.

The last test case is followed by a line containing two zeros.

Output

For each test case output a line with an integer representing the minimum possible loss for the gardener.

Sample input	**Output for the sample input**
2 3 2 2 10 4 4 10 2 4 10 4 2 10 3 3 10 2 3 2 2 20 4 4 20 2 4 10 4 2 10 3 3 10 1 1 -10000 -10000 1000 10000 10000 1000 2 2 0 0 4 0 2 2 0 1 3 0 4 1 4 1 0 1 1000 0 -1 1000 1 0 1000 -1 0 1000 0 0 1 0 0	10 20 0 2 1

Problem H

Hedge Mazes

Problem code name: `hedge`

The Queen of Nlogonia is a fan of mazes, and therefore the queendom's architects built several mazes around the Queen's palace. Every maze built for the Queen is made of rooms connected by corridors. Each corridor connects a different pair of distinct rooms and can be transversed in both directions.

The Queen loves to stroll through a maze's rooms and corridors in the late afternoon. Her servants choose a different challenge for every day, that consists of finding a simple path from a start room to an end room in a maze. A simple path is a sequence of *distinct* rooms such that each pair of consecutive rooms in the sequence is connected by a corridor. In this case the first room of the sequence must be the start room, and the last room of the sequence must be the end room. The Queen thinks that a challenge is good when, among the routes from the start room to the end room, *exactly* one of them is a simple path. Can you help the Queen's servants to choose a challenge that pleases the Queen?

For doing so, write a program that given the description of a maze and a list of queries defining the start and end rooms, determines for each query whether that choice of rooms is a good challenge or not.

Input

Each test case is described using several lines. The first line contains three integers R, C and Q representing respectively the number of rooms in a maze ($2 \leq R \leq 10^4$), the number of corridors ($1 \leq C \leq 10^5$), and the number of queries ($1 \leq Q \leq 1000$). Rooms are identified by different integers from 1 to R. Each of the next C lines describes a corridor using two distinct integers A and B, indicating that there is a corridor connecting rooms A and B ($1 \leq A < B \leq R$). After that, each of the next Q lines describes a query using two distinct integers S and T indicating respectively the start and end rooms of a challenge ($1 \leq S < T \leq R$). You may assume that within each test case there is at most one corridor connecting each pair of rooms, and no two queries are the same.

The last test case is followed by a line containing three zeros.

Output

For each test case output $Q + 1$ lines. In the i-th line write the answer to the i-th query. If the rooms make a good challenge, then write the character '`Y`' (uppercase). Otherwise write the character '`N`' (uppercase). Print a line containing a single character '`-`' (hyphen) after each test case.

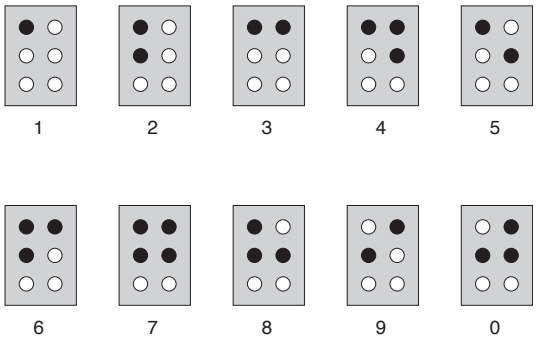
Sample input	**Output for the sample input**
6 5 3 1 2 2 3 2 4 2 5 4 5 1 3 1 5 2 6 4 2 3 1 2 2 3 1 4 1 3 1 2 0 0 0	Y N N - N Y Y -

Problem I

In Braille

Problem code name: `inbraille`

The Braille system, designed by Louis Braille in 1825, revolutionized written communication for blind and visually impaired persons. Braille, a blind Frenchman, developed a tactile language where each element is represented by a cell with six dot positions, arranged in three rows and two columns. Each dot position can be raised or not, allowing for 64 different configurations which can be felt by trained fingers. The figure below shows the Braille representation for the decimal digits (a black dot indicates a raised position).



In order to develop a new software system to help teachers to deal with blind or visual impaired students, a Braille dictionary module is necessary. Given a message, composed only by digits, your job is to translate it to or from Braille. Can you help?

Input

Each test case is described using three or five lines. The first line contains an integer D representing the number of digits in the message ($1 \leq D \leq 100$). The second line contains a single uppercase letter '`S`' or '`B`'. If the letter is '`S`', the next line contains a message composed of D decimal digits that your program must translate to Braille. If the letter is '`B`', the next three lines contain a message composed of D Braille cells that your program must translate from Braille. Braille cells are separated by single spaces. In each Braille cell a raised position is denoted by the character '`*`' (asterisk), while a not raised position is denoted by the character '`.`' (dot).

The last test case is followed by a line containing one zero.

Output

For each test case print just the digits of the corresponding translation, in the same format as the input (see the examples for further clarification).

Sample input	**Output for the sample input**
10 S 1234567890 3 B *. *. ** .. *. .. .. 2 S 00 0	*. *. ** ** *. ** ** *. .* .* .. *. .. .* .* *. ** ** *. ** .. .. 123 .* .* ** ** .. ..

Problem J

Jupiter Attacks!

Problem code name: `jupiter`

Jupiter is invading! Major cities have been destroyed by Jovian spacecrafts and humanity is fighting back. Nlogonia is spearheading the counter-offensive, by hacking into the spacecrafts' control system.

Unlike Earthling computers, in which usually a byte has 2^8 possible values, Jovian computers use bytes with B possible values, $\{0, 1, \ldots, B-1\}$. Nlogonian software engineers have reverse-engineered the firmware for the Jovian spacecrafts, and plan to sabotage it so that the ships eventually self-destruct.

As a security measure, however, the Jovian spacecrafts run a supervisory program that periodically checks the integrity of the firmware, by hashing portions of it and comparing the result against known good values. To hash the portion of the firmware from the byte at position i to the byte at position j, the supervisor uses the hash function

$$H(f_i, \ldots f_j) = \sum_{k=0}^{j-i} B^k f_{j-k} \pmod{P}$$

where P is a prime number. For instance, if $B = 20$ and $P = 139$, while bytes 2 to 5 of the firmware have the values $f_2 = 14$, $f_3 = 2$, $f_4 = 2$, and $f_5 = 4$, then

$$\begin{aligned} H(f_2, \ldots f_5) &= B^0 f_5 + B^1 f_4 + B^2 f_3 + B^3 f_2 \pmod{P} \\ &= 20^0 \times 4 + 20^1 \times 2 + 20^2 \times 2 + 20^3 \times 14 \pmod{139} \\ &= 4 + 40 + 800 + 112000 \pmod{139} \\ &= 112844 \pmod{139} \\ &= 115 \end{aligned}$$

The Nlogonian cryptologists need to find a way to sabotage the firmware without tripping the supervisor. As a first step, you have been assigned to write a program to simulate the interleaving of two types of commands: editing bytes of the firmware by the Nlogonian software engineers, and computing hashes of portions of the firmware by the Jovian supervisory program. At the beginning of the simulation the value of every byte in the firmware is zero.

Input

Each test case is described using several lines. The first line contains four integers B, P, L and N, where B is the number of possible values of a Jovian byte, P is the modulus of the Jovian hash ($2 \leq B < P \leq 10^9$ and P prime), L is the length (number of Jovian bytes) of the spacecrafts' firmware, and N is the number of commands to simulate ($1 \leq L, N \leq 10^5$). At the beginning of the simulation the value of every byte in the firmware is $f_i = 0$ for $1 \leq i \leq L$. Each of the next N lines describes a command to simulate. Each command description starts with an uppercase letter that is either '`E`' or '`H`', with the following meanings.

'`E`' → The line describes an *edit command*. The letter is followed by two integers I and V indicating that the byte at position I of the firmware (that is, f_I) must receive the value V ($1 \leq I \leq L$ and $0 \leq V \leq B-1$).

'`H`' → The line describes a *hash command*. The letter is followed by two integers I and J indicating that $H(f_I, \ldots f_J)$ must be computed ($1 \leq I \leq J \leq L$).

The last test case is followed by a line containing four zeros.

Output

For each test case output the results of the hash commands in the input. In the i-th line write an integer representing the result of the i-th hash command. Print a line containing a single character '-' (hyphen) after each test case.

Sample input	**Output for the sample input**
20 139 5 7 E 1 12 E 2 14 E 3 2 E 4 2 E 5 4 H 2 5 E 2 14 10 1000003 6 11 E 1 3 E 2 4 E 3 5 E 4 6 E 5 7 E 6 8 H 1 6 E 3 0 E 3 9 H 1 3 H 4 6 999999935 999999937 100000 7 E 100000 6 E 1 7 H 1 100000 E 50000 8 H 1 100000 H 25000 75000 H 23987 23987 0 0 0 0	115 - 345678 349 678 - 824973478 236724326 450867806 0 -

Problem K

King's Poker

Problem code name: `king`

Poker is one of the most widely played card games, and King's Poker is one of its variations. The game is played with a normal deck of 52 cards. Each card has one of 4 suits and one of 13 ranks. However, in King's Poker card suits are not relevant, while ranks are Ace (rank 1), 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack (rank 11), Queen (rank 12) and King (rank 13). The name of the game comes from the fact that in King's Poker, the King is the highest ranked card. But this is not the only difference between regular Poker and King's Poker. Players of King's Poker are dealt a hand of just three cards. There are three types of hands:

- A *set*, made of three cards of the same rank.
- A *pair*, which contains two cards of the same rank, with the other card unmatched.
- A *no-pair*, where no two cards have the same rank.

Hands are ranked using the following rules:

- Any set defeats any pair and any no-pair.
- Any pair defeats any no-pair.
- A set formed with higher ranked cards defeats any set formed with lower ranked cards.
- If the matched cards of two pairs have different ranks, then the pair with the higher ranked matched cards defeats the pair with the lower ranked matched cards.
- If the matched cards of two pairs have the same rank, then the unmatched card of both hands are compared; the pair with the higher ranked unmatched card defeats the pair with the lower ranked unmatched card, unless both unmatched cards have the same rank, in which case there is a tie.

A new software house wants to offer King's Poker games in its on-line playing site, and needs a piece of software that, given a hand of King's Poker, determines the set or pair with the lowest rank that beats the given hand. Can you code it?

Input

Each test case is described using a single line. The line contains three integers A, B, and C representing the ranks of the cards dealt in a hand ($1 \leq A, B, C \leq 13$).

The last test case is followed by a line containing three zeros.

Output

For each test case output a single line. If there exists a set or a pair that beats the given hand, write the lowest ranked such a hand. The beating hand must be written by specifying the ranks of their cards, in non-decreasing order. If no set or pair beats the given hand, write the character '`*`' (asterisk).

Sample input	**Output for the sample input**
1 1 1 1 1 12 1 1 13 1 13 1 10 13 10 1 2 2 13 13 13 13 12 13 12 12 12 3 1 4 1 5 9 0 0 0	2 2 2 1 1 13 1 2 2 1 2 2 1 11 11 2 2 3 * 1 1 1 13 13 13 1 1 2 1 1 2

Problem L. Gates

Let us consider a circuit consisting of n gates. The gates are numbered from 0 to $n-1$. Each gate has a certain number of inputs and exactly one output. Each of them (inputs and outputs) may be in either one of the states 0, 1 or 1/2. Each input is connected to exactly one output of some gate. Input's state equals the state of the output it is connected to. Each output may be connected to an arbitrary number of inputs. The gates with numbers 0 and 1 are special — they don't have any input at all while their outputs are always in the following states: 0 for a gate with a number 0, 1 for a gate with a number 1. We say that the state of the output of a gate (in short: gate's state) is "valid", if:

1. it equals 0 and the gate has more inputs in state 0 than in state 1.
2. it equals 1/2 and the gate has the same number of inputs in state 0 as in state 1.
3. it equals 1 and the gate has more inputs in state 1 than it has in state 0.
4. the gate is special, i.e. it's number is 0 or 1, and its state is 0 or 1 respectively.

We say that a circuit's state is "valid" if all the states of its gates are valid. We say that a gate's state is "fixed" if the gate is in the same state in all circuit's valid states.

Write a programme that reads the circuit's description from the input file, for each gate checks if it's state is fixed, and determines it, if so, writes the determined states of gates to the standard output.

Input

The first line of the standard input contains the number of gates n, $2 \leq n \leq 10000$. The following $n-2$ lines contain the descriptions of gates' connections — line no. i describes inputs of the gate no. i. There is the number k_i of inputs of this gate, followed by k_i numbers of gates, $k_i \geq 1$. Those are the numbers of gates whose outputs are connected to successive inputs of the gate's no. i. Numbers in each line are separated by single spaces. The total number of all inputs of all gates does not exceed 200000.

Output

Your programme should write n lines to the standard output. Depending on the state of gate no. $i-1$, i-th line should contain:

- 0 — if it is determined and equals 0,
- 1/2 — if it is determined and equals 1/2,
- 1 — if it is determined and equals 1,
- ? (question mark) — if it is not determined.

Example

standard input	standard output
5 2 0 1 2 4 2 2 2 4	0 1 1/2 ? ?

Problem M. Game

Let us consider a game on a rectangular board $m \times 1$ consisting of m elementary squares numbered successively from 1 to m. There are n pawns on the board, each on a distinct square. None of them occupies the square with number m. Each single move in the is the following action: the moving player picks a pawn from any occupied square chosen at will and places it on the first unoccupied square with a larger number. The two players make moves in turn.

The one who puts a pawn on the last square, i.e. the square with a number m, wins.

We say a player's move is winning if after making it he can win the game, no matter what moves his opponent makes.

Write a programme that reads the size of a board and the initial setup of pawns from the input file, determines the number of distinct winning moves the starting player may choose in the given initial situation, and writes the result to the standard output.

Input

The first line of the input contains two integers m and n ($2 \leq m \leq 10^9, 1 \leq n \leq 10^6, n < m$) separated by a single space. The second line contains n increasing numbers — these are the numbers of squares the pawns are set on. Numbers in the line are separated by single spaces.

Output

The first and only output line should contain the number of distinct winning moves possible for the starting player in the given initial situation.

Example

standard input	standard output
5 2 1 3	1
5 2 2 3	0