ACM Collegiate Programming Contest 2015 (Hong Kong)

CO-ORGANIZERS:

Venue:	The Hong Kong Polytechnic University
Time:	2015-06-13 [Sat] 1400—1800
Number of Questions:	7

(This is a blank page.)

Problem A. SORTING TEXT, SORTING NUMBERS

Input:	Standard Input
Output:	Standard Output
Time Limit:	30 seconds
Memory Limit:	64 Megabytes

In this problem you will be given a series of lists that contain both words and integers. You have to write a program to sort each of the lists, in such a way that all words are in alphabetical order and all integers are in numerical order. Furthermore, if the n^{th} element in the list is a number it must remain a number; if it is a word it must remain a word.

INPUT

Each line of the input contains a list of words and integers. Each element of the list is separated by a comma followed by a space. Every list ends with a period. The input will be terminated by a line storing a single period.

There are at most 100 lists and each list contains at most 100,000 elements.

OUTPUT

For each list in the input, output the sorted list in a line. The elements should be separated by a comma followed by a space. Each list should end with a period.

EXAMPLE

Standard Input	Standard Output
0. banana, Apple, Orange. Banana, apPle, oraNGe. 10, 8, 6, 4, 2, 0. x, 30, -20, z, 1000, 1, Y. 50, 7, kitten, puppy, 2, bird. .	0. Apple, banana, Orange. apPle, Banana, oraNGe. 0, 2, 4, 6, 8, 10. x, -20, 1, Y, 30, 1000, z. 2, 7, bird, kitten, 50, puppy.

(This is a blank page.)

Problem B. N-CREDIBLE MAZES

Input:	Standard Input
Output:	Standard Output
Time Limit:	30 seconds
Memory Limit:	64 Megabytes

An **n-tersection** is defined as a location in an n-dimensional space, n being a positive integer, having all non-negative integer coordinates. For example, the location (1, 2, 3) represents an n-tersection in a three-dimensional space. Two n-tersection are said to be **adjacent** if they have the same number of dimensions and their coordinates differ by exactly 1 in a single dimension only. For example, (1, 2, 3) is adjacent to (0, 2, 3), (2, 2, 3) and (1, 2, 4), but not to (2, 3, 3) nor (3, 2, 3) nor (1, 2). An **n-teresting space** is defined as a collection of paths between adjacent n-tersections. Finally, an **n-credible maze** is defined as an n-teresting space combined with two specific n-tersections in that space, one of which is identified as the starting n-tersection and the other as the ending n-tersection.

INPUT

The input file will consist of the descriptions of one or more n-credible mazes. The first line of the description of a maze will specify n, the dimension of the n-teresting space ($0 \leq n \leq 10$). All coordinate values will be less than 10. The next line will contain $2n$ non-negative integers. The first n of which describe the starting n-tersection, and the next n of which describe the ending n-tersection. Next will be a non-negative number of lines containing $2n$ non-negative integers each, identifying paths between adjacent n-tersections in the n-teresting space. The list is terminated by a line containing only the value -1. Several such maze descriptions may be present in the file. The end of input is signaled by a space dimension of zero.

OUTPUT

For each maze output its position in the input; e.g., the first maze is "Maze #1", the second is "Maze #2", etc. If it is possible to travel through the n-credible maze's n-teresting space from the starting n-tersection to the ending n-tersection, also output "can be travelled" on the same line. If such travel is not possible, output "cannot be travelled" instead.

EXAMPLE

Standard Input	Standard Output
2 0 0 2 2 0 0 0 1 0 1 0 2 0 2 1 2 1 2 2 2 -1 3 1 1 1 1 2 3 1 1 2 1 1 3 1 1 3 1 2 3 1 1 1 1 1 0 1 1 0 1 0 0 1 0 0 0 0 0 -1 0	Maze #1 can be travelled Maze #2 cannot be travelled

Problem C. FRAME STACKING

Input:	Standard Input
Output:	Standard Output
Time Limit:	10 seconds
Memory Limit:	64 Megabytes

Consider the following 5 frames in a 9 × 8 array:

```
........ ........ ........ ........ .CCC....
EEEEEE.. ........ ........ ..BBBB.. .C.C....
E....E.. DDDDDD.. ........ ..B..B.. .C.C....
E....E.. D....D.. ........ ..B..B.. .CCC....
E....E.. D....D.. ....AAAA ..B..B.. ........
E....E.. D....D.. ....A..A ..BBBB.. ........
E....E.. DDDDDD.. ....A..A ........ ........
E....E.. ........ ....AAAA ........ ........
EEEEEE.. ........ ........ ........ ........

   1         2         3         4        5
```

If we stack them together, starting with 1 at the bottom and ending with 5 at the top, then from the top, we will see some frames cover some other frames. Viewing the stack of 5 given frames resembles the following picture.

```
.CCC....
ECBCBB..
DCBCDB..
DCCC.B..
D.B.ABAA
D.BBBB.A
DDDDAD.A
E...AAAA
EEEEEE..
```

Your program should read a picture of such stacked frames and determine the order in which the frames are stacked from bottom to top. In this example, your program should output "EDABC".

Here are some additional rules about the frame stacking:

1. The width of the frame is always exactly 1 character, and the sides are never shorter than 3 characters.
2. It is possible to see at least one part of each of the four sides of a frame. A corner shows two sides.

3. The frames will be lettered with capital letters, and no two frames will be assigned the same letter.

INPUT

The input contains a number of input blocks. Each input block contains the height h ($h \leq 30$) in the first line and the width w ($w \leq 30$) in the second line. A picture of the stacked frames is then given as h strings with w characters each. An input block with height 0 indicates the end of input.

OUTPUT

For each input block your program should output the solution to standard output. Each line of the solution contains the stacking order of the frames, from bottom to top. If there are multiple possibilities for an ordering, list all such possibilities in alphabetical order, one possibility a line. You can assume the inputs are correct and there will always be at least one legal ordering for each input block. List the output for all blocks in the input sequentially, without any blank lines.

EXAMPLE

Standard Input	Standard Output
9 8 .CCC.... ECBCBB.. DCBCDB.. DCCC.B.. D.B.ABAA D.BBBB.A DDDDAD.A E...AAAA EEEEEE.. 3 6 AAABBB A.AB.B AAABBB 0	EDABC AB BA

Problem D. OCTAL FRACTIONS

Input:	Standard Input
Output:	Standard Output
Time Limit:	10 seconds
Memory Limit:	64 Megabytes

Fractions in octal (base 8) notation can be expressed exactly in decimal notation. For example, 0.75 in octal is 0.963125 (7/8 + 5/64) in decimal. All octal numbers of n digits to the right of the octal point can be expressed in no more than $3n$ decimal digits to the right of the decimal point.

Your task is to write a program to convert octal numerals between 0 and 1, inclusive, into equivalent decimal numerals.

INPUT

The first line of the input indicates the number (≤ 100) of octal numerals to be converted, and each subsequent line stores one octal numeral. Each octal numeral may have up to 100,000 digits after the octal point.

OUTPUT

Your program should output each conversion in the following format:

$$d_0.d_1d_2d_3 \ldots d_k\ [8] = D_0.D_1D_2D_3 \ldots D_m\ [10]$$

where the left side is the input (in octal), and the right hand side is the decimal equivalent. There must be no trailing zeros on both sides, i.e., d_k and D_m must not be zero.

EXAMPLE

Standard Input	Standard Output
3 0.75 0.0001 0.1	0.75 [8] = 0.953125 [10] 0.0001 [8] = 0.000244140625 [10] 0.1 [8] = 0.125 [10]

(This is a blank page.)

Problem E. SAY A PASSWORD

Input:	Standard Input
Output:	Standard Output
Time Limit:	5 seconds
Memory Limit:	64 Megabytes

Password security is a tricky thing. Users prefer simple passwords that are easy to remember (like "hello"), but such passwords are often insecure. Some sites use random computer-generated passwords (like "xcwpcdg"), but users have a hard time remembering them. One postential solution is to generate "pronounceable" passwords that are relatively secure but still easy to remember.

Your program will verify if a password generated by a generator is acceptable or not. To be acceptable, a password must satisfy the following 3 rules:

1. It must contain at least one vowel.
2. It cannot contain three consecutive vowels or three consecutive consonants.
3. It cannot contain two consecutive occurrences of the same letter, except for "ee" or "oo".

(For the purposes of this problem, the vowels are "a", "e", "i", "o" and "u"; all other letters are consonants.)

INPUT

The input consists of one or more potential passwords, one per line, followed by a line containing only the word "end" that signals the end of the file. Each password is at least one and at most twenty letters long and consists of lower case letters.

OUTPUT

For each password, output whether or not it is acceptable, using the precise format shown in the example.

EXAMPLE

Standard Input	Standard Output
a tv ptoui bontres zoggax wiinq eep houctuh end	<a> is acceptable. <tv> is not acceptable. <ptoui> is not acceptable. <bontres> is not acceptable. <zoggax> is not acceptable. <wiinq> is not acceptable. <eep> is acceptable. <houctuh> is acceptable.

(This is a blank page.)

Problem F. HEX MINER

Input:	Standard Input
Output:	Standard Output
Time Limit:	30 seconds
Memory Limit:	64 Megabytes

In 2115, a space explorer has arrived a planet beyond Pluto called Nibiru, or Planet X. The surface of Nibiru is covered by hexagonal rock columns, i.e. hexagonal tiling, with the side equal to 1. The rock column contains one of three elements A, B and C found in the planet and arranged in way that no two adjacent columns will contain the same element (Figure 1). The amount of a particular element contain in a column is affected by the location of "hotspots" of the corresponding element, which is also a rock column. The hotspots actually are the source of elements which under the rock column and therefore, the content of corresponding element is relative high. For the column containing the same element outside the hotspot, its content will be dropped by 10 units on each ring away from the hotspot (Figure 2). The ring is formed by the minimum columns that can enclose the hotspot or the inner ring. The actual content of a column can be obtained by adding the amount provided by all the corresponding hotspots around plus a base amount, which is 10, i.e. in the absence of hotspots, a column will contain 10 units of its element.

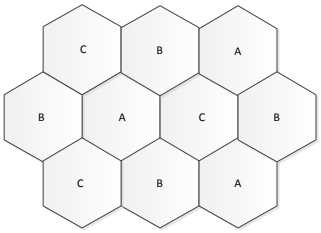

Figure 1

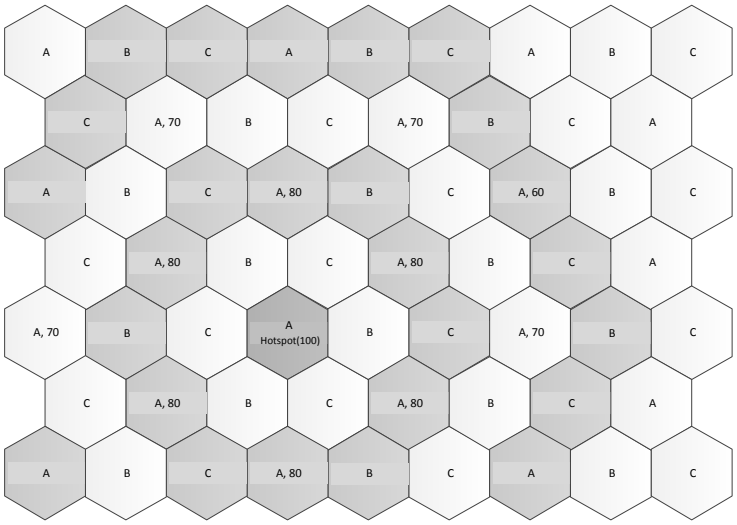

Figure 2

The purpose of the explorer is to collect these three elements to synthesize a compound call Bia, which is the fuel of the explorer. Since the chemical formula of Bia is $A_3B_2C_7$, i.e. 3 units of A, 2 units of B and 7 units of C are needed to synthesize Bia, the explorer is expected to collect these three elements in this proportion so that the maximum amount of Bia can be made. The explorer can only sent out one mining robot to the center of a rock column and extend its arms to two adjacent cells to collect the elements, i.e. only three cells will be mined and each of them should contain different elements and connected.

Before send out the robot, the explorer will use its radiation sensor to find out the hotspots around the center column, where the explorer is currently parked above. The explorer uses 3 elements coordinate system to index the column (figure 3). The distribution of the columns and the elements are same as the one in figure 2 with A located at (0, 0, 0) and B located at (-1, 0, 1).

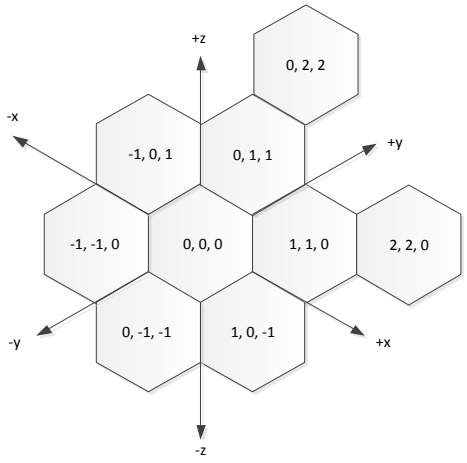

Figure 3

Your task is to find out the landing position of the robot where the maximum amount of Bia can be made and close to the explorer location (0, 0, 0). You may assume the robot can only land on the column with element A only and the maximum number of hotspot is 100.

INPUT

The input contains a number of input blocks. For each block, the first input is an integer N ($N \leq 100$) that indicates the numbers of hotspots detected, followed by N records. Each record contains four integers and one character. The first three integers are the x, y and z index of the column, followed by the content of the element. The last character is the element type of the hotspot. The value of x, y, z could be in the range of $-30{,}000$ to 30,000. The input ends with a block with -1 number of hotspots.

OUTPUT

For each input block, your program should output the location of column that the robot will be landed. In case more than one column are suitable for landing, list the locations in ascending order.

EXAMPLE

Standard Input	Standard Output
1 0 0 0 100 A 3 -1 1 2 25 A 2 1 -1 25 A -1 0 1 30 B	0 0 0 -1 1 2

Problem G. THE FUTURE KEYBOARD

Input:	Standard Input
Output:	Standard Output
Time Limit:	5 seconds
Memory Limit:	64 Megabytes

Imagine that you are living in the year of 2215. After 200 years of excessive consumption, the resources on earth become scarce. To reduce materials used, keyboards are made to have only five keys arranged from left to right. The keyboard user would press multiple keys simultaneously to input one of the 26 letters in the English alphabet and the space character.

Let's use 0 to represent a key not being pressed and 1 to represent a key being pressed. The user will use the keyboard pattern 00001 to input the first character A, 00010 to input the second character B, 00011 to input the third character C, … and 11010 (decimal value 26) to input Z. The user also uses keyboard pattern 11011 (decimal value 27) to input the space character. Imagine that in 2215, people don't use other punctuations or distinguish upper and lower case characters.

Your task is to write a program to recognize the user input on the future keyboard. Assume that the time sequence is divided into discrete moments numbered 1, 2, 3, 4, and so on. User input will only be read at those discrete moments.

INPUT

The input contains a number of input blocks. Each input block is consist of 5 lines. A line contains pairs of numbers as follows.

$a_1,b_1,a_2,b_2,\ldots$

The above pattern means that the respective key is pressed at moment a_1, released at moment b_1, pressed again at a_2, released again at b_2, and so on. The 5 lines of program input begin from the least significant digit (key) and progress to the most significant digit (key). In the first sample input block below, the least significant key was pressed at moment 2, released at moment 3, pressed at moment 5 and released at moment 9. (If a key has never been pressed, the corresponding line will be blank.)

OUTPUT

Your program should output one line of decoded message for each input block.

EXAMPLE

Standard Input	Standard Output
2,3,5,9 5,10 1,6,7,9,10,12 3,7,8,9,10,11 6,8,9,10 2,3,5,7,10,12 5,7,10,12 1,6,10,12,13,15 3,7,11,12,13,14 6,7,10,13	hello world hello world

Note that the second input block in the sample is different from the first sample and yet it produces the same output of "hello world". It is because the user did not press any key at moments 7, 8 and 9 as he or she was thinking about how to spell the word "world". No symbol will be generated in the output when no key was pressed at one of the discrete moments.

You can assume that only correct key combination will be pressed. In other words, the user will not input the key pattern of 11100 (decimal value 28) because it is not a character supported by the future keyboard.

— End of Problem Set —