Problem A
The Balance
Input: A.txt

Ms. Iyo Kiffa-Australis has a balance and only two kinds of weights to measure a dose of medicine.

For example, to measure 200mg of aspirin using 300mg weights and 700mg weights, she can put one 700mg weight on the side of the medicine and three 300mg weights on the opposite side (Figure 1). Although she could put four 300mg weights on the medicine side and two 700mg weights on the other (Figure 2), she would not choose this solution because it is less convenient to use more weights.

You are asked to help her by calculating how many weights are required.

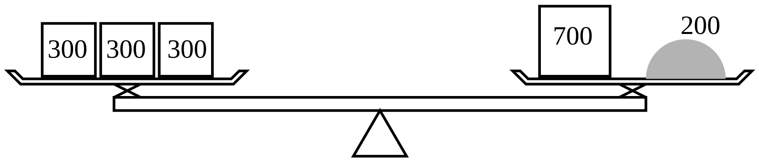

Figure 1: To measure 200mg of aspirin using three 300mg weights and one 700mg weight

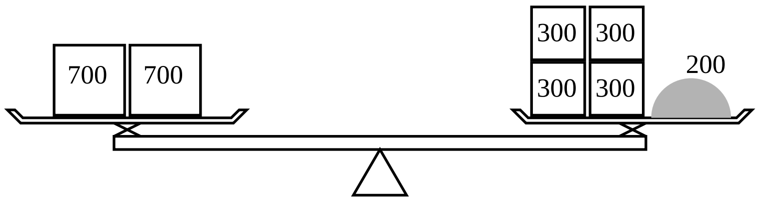

Figure 2: To measure 200mg of aspirin using four 300mg weights and two 700mg weights

Input

The input is a sequence of datasets. A dataset is a line containing three positive integers a, b, and d separated by a space. The following relations hold: $a \neq b$, $a \leq 10000$, $b \leq 10000$, and $d \leq 50000$. You may assume that it is possible to measure d mg using a combination of a mg and b mg weights. In other words, you need not consider "no solution" cases.

The end of the input is indicated by a line containing three zeros separated by a space. It is not a dataset.

Output

The output should be composed of lines, each corresponding to an input dataset (a, b, d). An output line should contain two nonnegative integers x and y separated by a space. They should satisfy the following three conditions.

- You can measure d mg using x many a mg weights and y many b mg weights.
- The total number of weights $(x + y)$ is the smallest among those pairs of nonnegative integers satisfying the previous condition.
- The total mass of weights $(ax + by)$ is the smallest among those pairs of nonnegative integers satisfying the previous two conditions.

No extra characters (e.g. extra spaces) should appear in the output.

Sample Input

```
700 300 200
500 200 300
500 200 500
275 110 330
275 110 385
648 375 4002
3 1 10000
0 0 0
```

Output for the Sample Input

```
1 3
1 1
1 0
0 3
1 1
49 74
3333 1
```

Problem B
Make a Sequence
Input: B.txt

Your company's next product will be a new game, which is a three-dimensional variant of the classic game "Tic-Tac-Toe". Two players place balls in a three-dimensional space (board), and try to make a sequence of a certain length.

People believe that it is fun to play the game, but they still cannot fix the values of some parameters of the game. For example, what size of the board makes the game most exciting? Parameters currently under discussion are the board size (we call it n in the following) and the length of the sequence (m). In order to determine these parameter values, you are requested to write a computer simulator of the game.

You can see several snapshots of the game in Figures 3–5. These figures correspond to the three datasets given in the Sample Input.

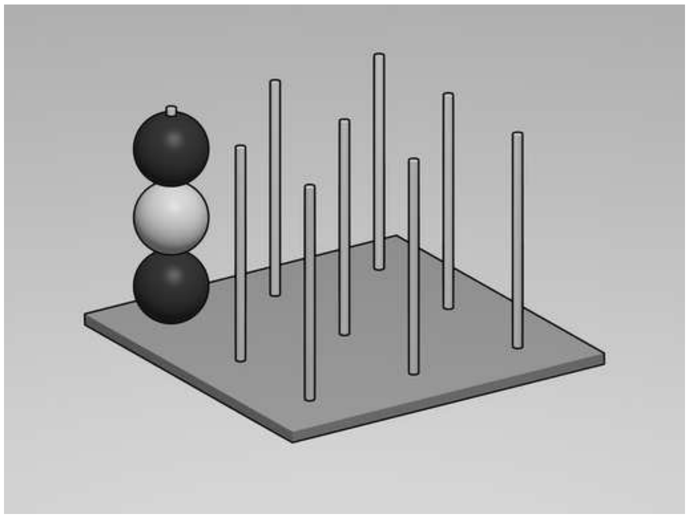

Figure 3: A game with $n = m = 3$

Here are the precise rules of the game.

1. Two players, Black and White, play alternately. Black plays first.
2. There are $n \times n$ vertical pegs. Each peg can accommodate up to n balls. A peg can be specified by its x- and y-coordinates ($1 \leq x, y \leq n$). A ball on a peg can be specified by its z-coordinate ($1 \leq z \leq n$). At the beginning of a game, there are no balls on any of the pegs.

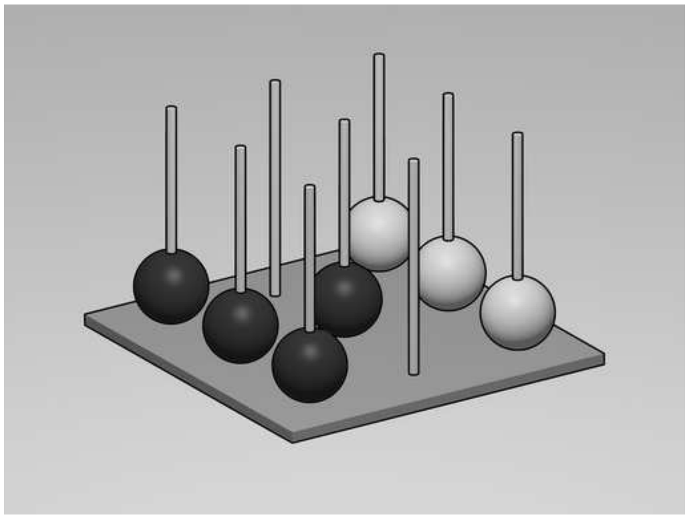

Figure 4: A game with $n = m = 3$ (White made a 3-sequence before Black)

3. On his turn, a player chooses one of $n \times n$ pegs, and puts a ball of his color onto the peg. The ball follows the law of gravity. That is, the ball stays just above the top-most ball on the same peg or on the floor (if there are no balls on the peg). Speaking differently, a player can choose x- and y-coordinates of the ball, but he cannot choose its z-coordinate.

4. The objective of the game is to make an m-sequence. If a player makes an m-sequence or longer of his color, he wins. An m-sequence is a row of m consecutive balls of the same color. For example, black balls in positions $(5, 1, 2)$, $(5, 2, 2)$ and $(5, 3, 2)$ form a 3-sequence.

   A sequence can be horizontal, vertical, or diagonal. Precisely speaking, there are 13 possible directions to make a sequence, categorized as follows.

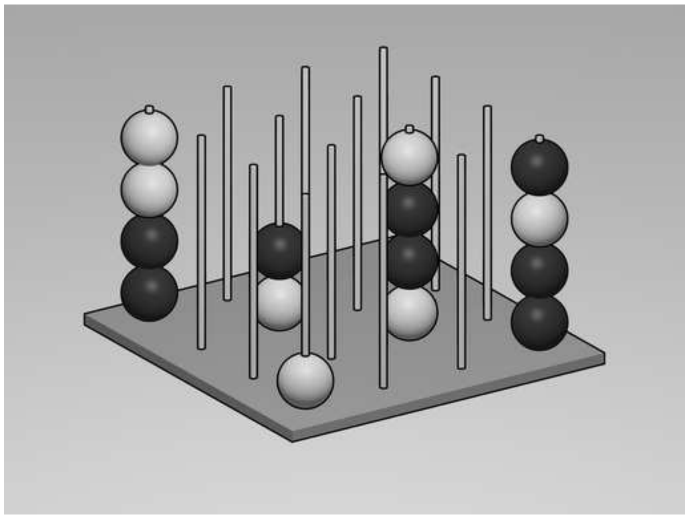

Figure 5: A game with $n = 4, m = 3$ (Black made two 4-sequences)

(a) One-dimensional axes. For example, $(3, 1, 2)$, $(4, 1, 2)$ and $(5, 1, 2)$ is a 3-sequence. There are three directions in this category.

(b) Two-dimensional diagonals. For example, $(2, 3, 1)$, $(3, 3, 2)$ and $(4, 3, 3)$ is a 3-sequence. There are six directions in this category.

(c) Three-dimensional diagonals. For example, $(5, 1, 3)$, $(4, 2, 4)$ and $(3, 3, 5)$ is a 3-sequence. There are four directions in this category.

Note that we do not distinguish between opposite directions.

As the evaluation process of the game, people have been playing the game several times changing the parameter values. You are given the records of these games. It is your job to write a computer program which determines the winner of each recorded game.

Since it is difficult for a human to find three-dimensional sequences, players often do not notice the end of the game, and continue to play uselessly. In these cases, moves after the end of the game, i.e. after the winner is determined, should be ignored. For example, after a player won making an m-sequence, players may make additional m-sequences. In this case, all m-sequences but the first should be ignored, and the winner of the game is unchanged.

A game does not necessarily end with the victory of one of the players. If there are no pegs left to put a ball on, the game ends with a draw. Moreover, people may quit a game before making any m-sequence. In such cases also, the game ends with a draw.

Input

The input consists of multiple datasets each corresponding to the record of a game. A dataset starts with a line containing three positive integers n, m, and p separated by a space. The relations $3 \leq m \leq n \leq 7$ and $1 \leq p \leq n^3$ hold between them. n and m are the parameter values of the game as described above. p is the number of moves in the game.

The rest of the dataset is p lines each containing two positive integers x and y. Each of these lines describes a move, i.e. the player on turn puts his ball on the peg specified. You can assume that $1 \leq x \leq n$ and $1 \leq y \leq n$. You can also assume that at most n balls are put on a peg throughout a game.

The end of the input is indicated by a line with three zeros separated by a space.

Output

For each dataset, a line describing the winner and the number of moves until the game ends should be output. The winner is either "`Black`" or "`White`". A single space should be inserted between the winner and the number of moves. No other extra characters are allowed in the output.

In case of a draw, the output line should be "`Draw`".

Sample Input

```
3 3 3
1 1
1 1
1 1
3 3 7
2 2
1 3
1 1
2 3
2 1
3 3
3 1
4 3 15
1 1
2 2
1 1
3 3
3 3
1 1
3 3
3 3
4 4
1 1
4 4
4 4
4 4
4 1
2 2
0 0 0
```

Output for the Sample Input

```
Draw
White 6
Black 15
```

Problem C
Leaky Cryptography
Input: C.txt

The ACM ICPC judges are very careful about not leaking their problems, and all communications are encrypted. However, one does sometimes make mistakes, like using too weak an encryption scheme. Here is an example of that.

The encryption chosen was very simple: encrypt each chunk of the input by flipping some bits according to a shared key. To provide reasonable security, the size of both chunk and key is 32 bits.

That is, suppose the input was a sequence of m 32-bit integers.

$$N_1 \quad N_2 \quad N_3 \ \ldots \ N_m$$

After encoding with the key K it becomes the following sequence of m 32-bit integers.

$$(N_1 \wedge K)\ (N_2 \wedge K)\ (N_3 \wedge K) \ldots (N_m \wedge K)$$

where $(a \wedge b)$ is the *bitwise exclusive or* of a and b.

Exclusive or is the logical operator which is 1 when only one of its operands is 1, and 0 otherwise. Here is its definition for 1-bit integers.

$$\begin{array}{ll} 0 \oplus 0 = 0 & 0 \oplus 1 = 1 \\ 1 \oplus 0 = 1 & 1 \oplus 1 = 0 \end{array}$$

As you can see, it is identical to addition modulo 2. For two 32-bit integers a and b, their bitwise exclusive or $a \wedge b$ is defined as follows, using their binary representations, composed of 0's and 1's.

$$a \wedge b = a_{31} \ldots a_1 a_0 \wedge b_{31} \ldots b_1 b_0 = c_{31} \ldots c_1 c_0$$

where

$$c_i = a_i \oplus b_i \ (i = 0, 1, \cdots, 31).$$

For instance, using binary notation, 11010110 ^ 01010101 = 10100011, or using hexadecimal, `d6` ^ `55` = `a3`.

Since this kind of encryption is notoriously weak to statistical attacks, the message has to be compressed in advance, so that it has no statistical regularity. We suppose that $N_1\ N_2 \ldots N_m$ is already in compressed form.

However, the trouble is that the compression algorithm itself introduces some form of regularity: after every 8 integers of compressed data, it inserts a checksum, the sum of these integers. That is, in the above input, $N_9 = \sum_{i=1}^{8} N_i = N_1 + \cdots + N_8$, where additions are modulo 2^{32}.

Luckily, you could intercept a communication between the judges. Maybe it contains a problem for the finals!

As you are very clever, you have certainly seen that you can easily find the lowest bit of the key, denoted by K_0. On the one hand, if $K_0 = 1$, then after encoding, the lowest bit of $\sum_{i=1}^{8} N_i \wedge K$ is unchanged, as K_0 is added an even number of times, but the lowest bit of $N_9 \wedge K$ is changed, so they shall differ. On the other hand, if $K_0 = 0$, then after encoding, the lowest bit of $\sum_{i=1}^{8} N_i \wedge K$ shall still be identical to the lowest bit of $N_9 \wedge K$, as they do not change. For instance, if the lowest bits after encoding are `1 1 1 1 1 1 1 1 1` then K_0 must be 1, but if they are `1 1 1 1 1 1 1 0 1` then K_0 must be 0.

So far, so good. Can you do better?

You should find the key used for encoding.

Input

The input starts with a line containing only a positive integer S, indicating the number of datasets in the input. S is no more than 1000.

It is followed by S datasets. Each dataset is composed of nine 32-bit integers corresponding to the first nine chunks of a communication. They are written in hexadecimal notation, using digits '`0`' to '`9`' and lowercase letters '`a`' to '`f`', and with no leading zeros. They are separated by a space or a newline. Each dataset is ended by a newline.

Output

For each dataset you should output the key used for encoding. Each key shall appear alone on its line, and be written in hexadecimal notation, using digits '`0`' to '`9`' and lowercase letters '`a`' to '`f`', and with no leading zeros.

Sample Input

```
8
1 1 1 1 1 1 1 1 8
3 2 3 2 3 2 3 2 6
3 4 4 7 7 b a 2 2e
e1 13 ce 28 ca 6 ab 46 a6d
b08 49e2 6128 f27 8cf2 bc50 7380 7fe1 723b
4eba eb4 a352 fd14 6ac1 eed1 dd06 bb83 392bc
ef593c08 847e522f 74c02b9c 26f3a4e1 e2720a01 6fe66007
7a4e96ad 6ee5cef6 3853cd88
60202fb8 757d6d66 9c3a9525 fbcd7983 82b9571c ddc54bab 853e52da
22047c88 e5524401
```

Output for the Sample Input

```
0
2
6
1c6
4924afc7
ffff95c5
546991d
901c4a16
```

Problem D
Pathological Paths
Input: D.txt

Professor Pathfinder is a distinguished authority on the structure of hyperlinks in the World Wide Web. For establishing his hypotheses, he has been developing software agents, which automatically traverse hyperlinks and analyze the structure of the Web. Today, he has gotten an intriguing idea to improve his software agents. However, he is very busy and requires help from good programmers. You are now being asked to be involved in his development team and to create a small but critical software module of his new type of software agents.

Upon traversal of hyperlinks, Pathfinder's software agents incrementally generate a map of visited portions of the Web. So the agents should maintain the list of traversed hyperlinks and visited web pages. One problem in keeping track of such information is that two or more different URLs can point to the same web page. For instance, by typing any one of the following five URLs, your favorite browsers probably bring you to the same web page, which as you may have visited is the home page of the ACM ICPC Ehime contest.

```
http://www.ehime-u.ac.jp/ICPC/
http://www.ehime-u.ac.jp/ICPC
http://www.ehime-u.ac.jp/ICPC/../ICPC/
http://www.ehime-u.ac.jp/ICPC/./
http://www.ehime-u.ac.jp/ICPC/index.html
```

Your program should reveal such aliases for Pathfinder's experiments.

Well, . . . but it were a real challenge and to be perfect you might have to embed rather complicated logic into your program. We are afraid that even excellent programmers like you could not complete it in five hours. So, we make the problem a little simpler and subtly unrealistic. You should focus on the path parts (i.e. `/ICPC/`, `/ICPC`, `/ICPC/../ICPC/`, `/ICPC/./`, and `/ICPC/index.html` in the above example) of URLs and ignore the scheme parts (e.g. `http://`), the server parts (e.g. `www.ehime-u.ac.jp`), and other optional parts. You should carefully read the rules described in the sequel since some of them may not be based on the reality of today's Web and URLs.

Each path part in this problem is an absolute pathname, which specifies a path from the root directory to some web page in a hierarchical (tree-shaped) directory structure. A pathname always starts with a slash (`/`), representing the root directory, followed by path segments delimited by a slash. For instance, `/ICPC/index.html` is a pathname with two path segments `ICPC` and `index.html`.

All those path segments but the last should be directory names and the last one the name of an ordinary file where a web page is stored. However, we have one exceptional rule: an ordinary file name `index.html` at the end of a pathname may be omitted. For instance, a pathname `/ICPC/index.html` can be shortened to `/ICPC/`, if `index.html` is an existing ordinary file name. More precisely, if `ICPC` is the name of an existing directory just under the root and `index.html` is the name of an existing ordinary file just under the `/ICPC` directory, `/ICPC/index.html` and `/ICPC/` refer to the same web page. Furthermore, the last slash following the last path segment can also be omitted. That is, for instance, `/ICPC/` can be further shortened to `/ICPC`. However, `/index.html` can only be abbreviated to `/` (a single slash).

You should pay special attention to path segments consisting of a single period (`.`) or a double period (`..`), both of which are always regarded as directory names. The former represents the directory itself and the latter represents its parent directory. Therefore, if `/ICPC/` refers to some web page, both `/ICPC/./` and `/ICPC/../ICPC/` refer to the same page. Also `/ICPC2/../ICPC/` refers to the same page if `ICPC2` is the name of an existing directory just under the root; otherwise it does not refer to any web page. Note that the root directory does not have any parent directory and thus such pathnames as `/../` and `/ICPC/../../index.html` cannot point to any web page.

Your job in this problem is to write a program that checks whether two given pathnames refer to existing web pages and, if so, examines whether they are the same.

Input

The input consists of multiple datasets. The first line of each dataset contains two positive integers N and M, both of which are less than or equal to 100 and are separated by a single space character.

The rest of the dataset consists of $N + 2M$ lines, each of which contains a syntactically correct pathname of at most 100 characters. You may assume that each path segment enclosed by two slashes is of length at least one. In other words, two consecutive slashes cannot occur in any pathname. Each path segment does not include anything other than alphanumerical characters (i.e. '`a`'–'`z`', '`A`'–'`Z`', and '`0`'–'`9`') and periods ('`.`').

The first N pathnames enumerate all the web pages (ordinary files). Every existing directory name occurs at least once in these pathnames. You can assume that these pathnames do not include any path segments consisting solely of single or double periods and that the last path segments are ordinary file names. Therefore, you do not have to worry about special rules for `index.html` and single/double periods. You can also assume that no two of the N pathnames point to the same page.

Each of the following M pairs of pathnames is a question: do the two pathnames point to the same web page? These pathnames may include single or double periods and may be terminated by a slash. They may include names that do not correspond to existing directories or ordinary files.

Two zeros in a line indicate the end of the input.

Output

For each dataset, your program should output the M answers to the M questions, each in a separate line. Each answer should be "`yes`" if both point to the same web page, "`not found`" if at least one of the pathnames does not point to any one of the first N web pages listed in the input, or "`no`" otherwise.

Sample Input

```
5 6
/home/ACM/index.html
/ICPC/index.html
/ICPC/general.html
/ICPC/japanese/index.html
/ICPC/secret/confidential/2005/index.html
/home/ACM/
/home/ICPC/../ACM/
/ICPC/secret/
/ICPC/secret/index.html
/ICPC
/ICPC/../ICPC/index.html
/ICPC
/ICPC/general.html
/ICPC/japanese/.././
/ICPC/japanese/./../
/home/ACM/index.html
/home/ACM/index.html/
1 4
/index.html/index.html
/
/index.html/index.html
/index.html
/index.html/index.html
/..
/index.html/../..
/index.html/
/index.html/index.html/..
0 0
```

Output for the Sample Input

```
not found
not found
yes
no
yes
not found
not found
yes
not found
not found
```

Problem E
Confusing Login Names
Input: E.txt

Meikyokan University is very famous for its research and education in the area of computer science. This university has a computer center that has advanced and secure computing facilities including supercomputers and many personal computers connected to the Internet.

One of the policies of the computer center is to let the students select their own login names. Unfortunately, students are apt to select similar login names, and troubles caused by mistakes in entering or specifying login names are relatively common. These troubles are a burden on the staff of the computer center.

To avoid such troubles, Dr. Choei Takano, the chief manager of the computer center, decided to stamp out similar and confusing login names. To this end, Takano has to develop a program that detects confusing login names.

Based on the following four operations on strings, the distance between two login names is determined as the minimum number of operations that transforms one login name to the other.

1. Deleting a character at an arbitrary position.
2. Inserting a character into an arbitrary position.
3. Replacing a character at an arbitrary position with another character.
4. Swapping two adjacent characters at an arbitrary position.

For example, the distance between "`omura`" and "`murai`" is two, because the following sequence of operations transforms "`omura`" to "`murai`".

$$\texttt{omura} \xrightarrow{\text{delete `}\texttt{o}\text{'}} \texttt{mura} \xrightarrow{\text{insert `}\texttt{i}\text{'}} \texttt{murai}$$

Another example is that the distance between "`akasan`" and "`kaason`" is also two.

$$\texttt{akasan} \xrightarrow{\text{swap `}\texttt{a}\text{' and `}\texttt{k}\text{'}} \texttt{kaasan} \xrightarrow{\text{replace `}\texttt{a}\text{' with `}\texttt{o}\text{'}} \texttt{kaason}$$

Takano decided that two login names with a small distance are confusing and thus must be avoided.

Your job is to write a program that enumerates all the confusing pairs of login names.

Beware that the rules may combine in subtle ways. For instance, the distance between "`ant`" and "`neat`" is two.

$$\texttt{ant} \xrightarrow{\text{swap `a' and `n'}} \texttt{nat} \xrightarrow{\text{insert `e'}} \texttt{neat}$$

Input

The input consists of multiple datasets. Each dataset is given in the following format.

n
d
$name_1$
$name_2$
$\ldots$
$name_n$

The first integer n is the number of login names. Then comes a positive integer d. Two login names whose distance is less than or equal to d are deemed to be confusing. You may assume that $0 < n \leq 200$ and $0 < d \leq 2$. The i-th student's login name is given by $name_i$, which is composed of only lowercase letters. Its length is less than 16. You can assume that there are no duplicates in $name_i$ ($1 \leq i \leq n$).

The end of the input is indicated by a line that solely contains a zero.

Output

For each dataset, your program should output all pairs of confusing login names, one pair per line, followed by the total number of confusing pairs in the dataset.

In each pair, the two login names are to be separated only by a comma character (`,`), and the login name that is alphabetically preceding the other should appear first. The entire output of confusing pairs for each dataset must be sorted as follows. For two pairs "w_1`,`w_2" and "w_3`,`w_4", if w_1 alphabetically precedes w_3, or they are the same and w_2 precedes w_4, then "w_1`,`w_2" must appear before "w_3`,`w_4".

Sample Input

```
8
2
omura
toshio
raku
tanaka
imura
yoshoi
hayashi
miura
3
1
tasaka
nakata
tanaka
1
1
foo
5
2
psqt
abcdef
abzdefa
pqrst
abdxcef
0
```

Output for the Sample Input

```
imura,miura
imura,omura
miura,omura
toshio,yoshoi
4
tanaka,tasaka
1
0
abcdef,abdxcef
abcdef,abzdefa
pqrst,psqt
3
```

Problem F
Dice Puzzle
Input: F.txt

Let's try a dice puzzle. The rules of this puzzle are as follows.

1. Dice with six faces as shown in Figure 6 are used in the puzzle.

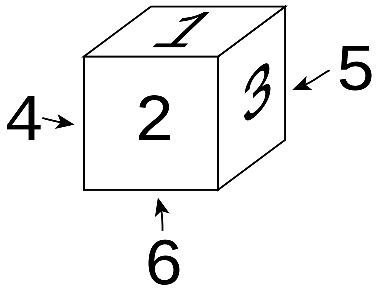

Figure 6: Faces of a die

2. With twenty seven such dice, a $3 \times 3 \times 3$ cube is built as shown in Figure 7.

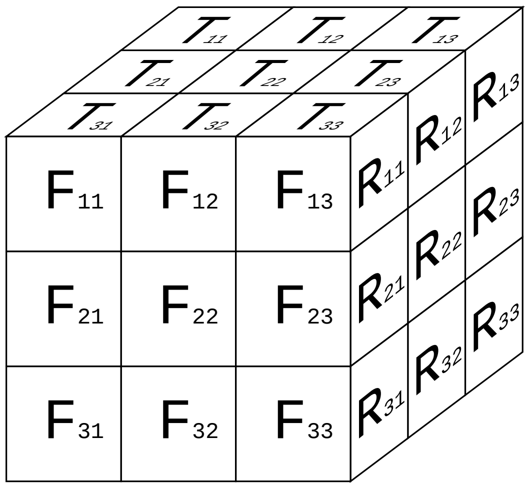

Figure 7: $3 \times 3 \times 3$ cube

3. When building up a cube made of dice, the sum of the numbers marked on the faces

of adjacent dice that are placed against each other must be seven (See Figure 8). For example, if one face of the pair is marked “2”, then the other face must be “5”.

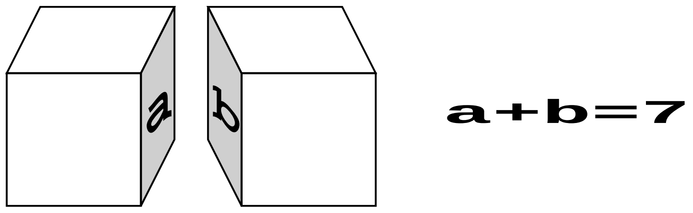

Figure 8: A pair of faces placed against each other

4. The top and the front views of the cube are partially given, i.e. the numbers on faces of some of the dice on the top and on the front are given.

T_{11}	T_{12}	T_{13}
T_{21}	T_{22}	T_{23}
T_{31}	T_{32}	T_{33}

Top

F_{11}	F_{12}	F_{13}
F_{21}	F_{22}	F_{23}
F_{31}	F_{32}	F_{33}

Front

Figure 9: Top and front views of the cube

5. The goal of the puzzle is to find all the plausible dice arrangements that are consistent with the given top and front view information.

Your job is to write a program that solves this puzzle.

Input

The input consists of multiple datasets in the following format.

N
$Dataset_1$
$Dataset_2$
...
$Dataset_N$

N is the number of the datasets.

The format of each dataset is as follows.

T_{11} T_{12} T_{13}
T_{21} T_{22} T_{23}
T_{31} T_{32} T_{33}
F_{11} F_{12} F_{13}
F_{21} F_{22} F_{23}
F_{31} F_{32} F_{33}

T_{ij} and F_{ij} ($1 \leq i \leq 3$, $1 \leq j \leq 3$) are the faces of dice appearing on the top and front views, as shown in Figure 7, or a zero. A zero means that the face at the corresponding position is unknown.

Output

For each plausible arrangement of dice, compute the sum of the numbers marked on the nine faces appearing on the right side of the cube, that is, with the notation given in Figure 7, $\sum_{i=1}^{3} \sum_{j=1}^{3} R_{ij}$.

For each dataset, you should output the right view sums for all the plausible arrangements, in ascending order and without duplicates. Numbers should be separated by a single space.

When there are no plausible arrangements for a dataset, output a zero.

For example, suppose that the top and the front views are given as follows.

1	0	0
0	2	0
0	0	0

Top view

5	1	2
5	1	2
0	0	0

Front view

Figure 10: Example

There are four plausible right views as shown in Figure 11. The right view sums are 33, 36, 32, and 33, respectively. After rearranging them into ascending order and eliminating duplicates, the answer should be "32 33 36".

4	4	4
4	4	4
3	3	3

4	4	4
4	4	4
4	4	4

3	4	4
3	4	4
4	3	3

3	4	4
3	4	4
3	4	4

Figure 11: Plausible right views

The output should be one line for each dataset. The output may have spaces at ends of lines.

Sample Input

```
4
1 1 1
1 1 1
1 1 1
2 2 2
2 2 2
2 2 2
4 3 3
5 2 2
4 3 3
6 1 1
6 1 1
6 1 0
1 0 0
0 2 0
0 0 0
5 1 2
5 1 2
0 0 0
2 0 0
0 3 0
0 0 0
0 0 0
0 0 0
3 0 1
```

Output for the Sample Input

```
27
24
32 33 36
0
```

Problem G
Color the Map
Input: G.txt

You were lucky enough to get a map just before entering the legendary magical mystery world. The map shows the whole area of your planned exploration, including several countries with complicated borders. The map is clearly drawn, but in sepia ink only; it is hard to recognize at a glance which region belongs to which country, and this might bring you into severe danger. You have decided to color the map before entering the area. "A good deal depends on preparation," you talked to yourself.

Each country has one or more territories, each of which has a polygonal shape. Territories belonging to one country may or may not "touch" each other, i.e. there may be disconnected territories. All the territories belonging to the same country must be assigned the same color. You can assign the same color to more than one country, but, to avoid confusion, two countries "adjacent" to each other should be assigned different colors. Two countries are considered to be "adjacent" if any of their territories share a border of non-zero length.

Write a program that finds the least number of colors required to color the map.

Input

The input consists of multiple map data. Each map data starts with a line containing the total number of territories n, followed by the data for those territories. n is a positive integer not more than 100. The data for a territory with m vertices has the following format:

$String$
$x_1 \;\; y_1$
$x_2 \;\; y_2$
$\ldots$
$x_m \;\; y_m$
-1

"*String*" (a sequence of alphanumerical characters) gives the name of the country it belongs to. A country name has at least one character and never has more than twenty. When a country has multiple territories, its name appears in each of them.

Remaining lines represent the vertices of the territory. A vertex data line has a pair of nonnegative integers which represent the x- and y-coordinates of a vertex. x- and y-coordinates are separated by a single space, and y-coordinate is immediately followed by a newline. Edges of the territory are obtained by connecting vertices given in two adjacent vertex data lines, and by connecting vertices given in the last and the first vertex data lines. None of x- and y-coordinates exceeds 1000. Finally, -1 in a line marks the end of vertex data lines. The number of vertices m does not exceed 100.

You may assume that the contours of polygons are simple, i.e. they do not cross nor touch themselves. No two polygons share a region of non-zero area. The number of countries in a map does not exceed 10.

The last map data is followed by a line containing only a zero, marking the end of the input data.

Output

For each map data, output one line containing the least possible number of colors required to color the map satisfying the specified conditions.

Sample Input

```
6
Blizid
0 0
60 0
60 60
0 60
0 50
50 50
50 10
0 10
-1
Blizid
0 10
10 10
10 50
0 50
-1
Windom
10 10
50 10
40 20
20 20
20 40
10 50
-1
Accent
50 10
50 50
```

```
35 50
35 25
-1
Pilot
35 25
35 50
10 50
-1
Blizid
20 20
40 20
20 40
-1
4
A1234567890123456789
0 0
0 100
100 100
100 0
-1
B1234567890123456789
100 100
100 200
200 200
200 100
-1
C1234567890123456789
0 100
100 100
100 200
0 200
-1
D123456789012345678
100 0
100 100
200 100
200 0
-1
0
```

Output for the Sample Input

```
4
2
```

Problem H
Inherit the Spheres
Input: H.txt

In the year 2xxx, an expedition team landing on a planet found strange objects made by an ancient species living on that planet. They are transparent boxes containing opaque solid spheres (Figure 12). There are also many lithographs which seem to contain positions and radiuses of spheres.

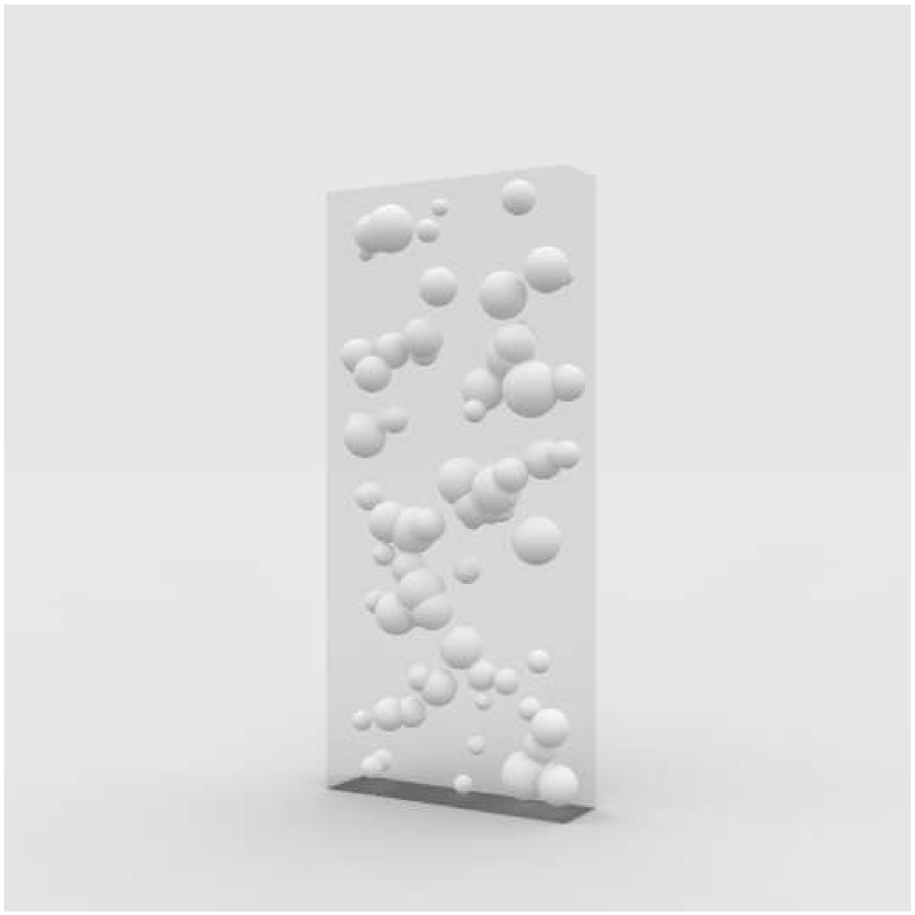

Figure 12: A strange object

Initially their objective was unknown, but Professor Zambendorf found the cross section formed by a horizontal plane plays an important role. For example, the cross section of an object changes as in Figure 13 by sliding the plane from bottom to top.

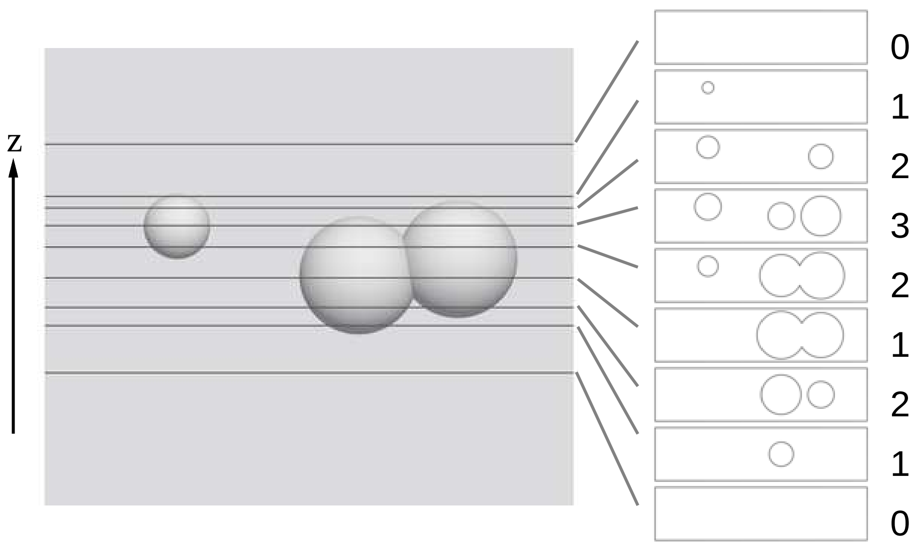

Figure 13: Cross sections at different positions

He eventually found that some information is expressed by the transition of the number of connected figures in the cross section, where each connected figure is a union of discs intersecting or touching each other, and each disc is a cross section of the corresponding solid sphere. For instance, in Figure 13, whose geometry is described in the first sample dataset later, the number of connected figures changes as 0, 1, 2, 1, 2, 3, 2, 1, and 0, at $z = 0.0000$, 162.0000, 167.0000, 173.0004, 185.0000, 191.9996, 198.0000, 203.0000, and 205.0000, respectively. By assigning 1 for increment and 0 for decrement, the transitions of this sequence can be expressed by an 8-bit binary number 11011000.

For helping further analysis, write a program to determine the transitions when sliding the horizontal plane from bottom ($z = 0$) to top ($z = 36000$).

Input

The input consists of a series of datasets. Each dataset begins with a line containing a positive integer, which indicates the number of spheres N in the dataset. It is followed by N lines describing the centers and radiuses of the spheres. Each of the N lines has four positive integers X_i, Y_i, Z_i, and R_i ($i = 1, \cdots, N$) describing the center and the radius of the i-th sphere, respectively.

You may assume $1 \leq N \leq 100$, $1 \leq R_i \leq 2000$, $0 < X_i - R_i < X_i + R_i < 4000$, $0 < Y_i - R_i < Y_i + R_i < 16000$, and $0 < Z_i - R_i < Z_i + R_i < 36000$. Each solid sphere is defined as the set of all points (x, y, z) satisfying $(x - X_i)^2 + (y - Y_i)^2 + (z - Z_i)^2 \leq R_i^2$.

A sphere may contain other spheres. No two spheres are mutually tangent. Every $Z_i \pm R_i$ and minimum/maximum z coordinates of a circle formed by the intersection of any two spheres differ from each other by at least 0.01.

The end of the input is indicated by a line with one zero.

Output

For each dataset, your program should output two lines. The first line should contain an integer M indicating the number of transitions. The second line should contain an M-bit binary number that expresses the transitions of the number of connected figures as specified above.

Sample Input

```
3
95 20 180 18
125 20 185 18
40 27 195 10
1
5 5 5 4
2
5 5 5 4
5 5 5 3
2
5 5 5 4
5 7 5 3
16
2338 3465 29034 710
1571 14389 25019 842
1706 8015 11324 1155
1899 4359 33815 888
2160 10364 20511 1264
2048 8835 23706 1906
2598 13041 23679 618
1613 11112 8003 1125
1777 4754 25986 929
2707 9945 11458 617
1153 10358 4305 755
2462 8450 21838 934
1822 11539 10025 1639
1473 11939 12924 638
1388 8519 18653 834
2239 7384 32729 862
0
```

Output for the Sample Input

```
8
11011000
2
10
2
10
2
10
28
1011100100110101101000101100
```

Problem I
Crossing Prisms
Input: I.txt

Prof. Bocchan is a mathematician and a sculptor. He likes to create sculptures with mathematics.

His style to make sculptures is very unique. He uses two identical prisms. Crossing them at right angles, he makes a polyhedron that is their intersection as a new work. Since he finishes it up with painting, he needs to know the surface area of the polyhedron for estimating the amount of pigment needed.

For example, let us consider the two identical prisms in Figure 14. The definition of their cross section is given in Figure 15. The prisms are put at right angles with each other and their intersection is the polyhedron depicted in Figure 16. An approximate value of its surface area is 194.8255.

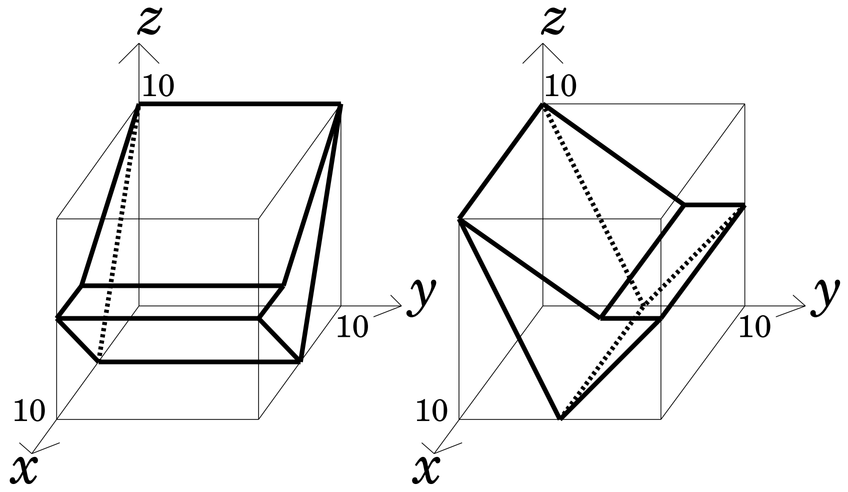

Figure 14: Two identical prisms at right angles

Given the shape of the cross section of the two identical prisms, your job is to calculate the surface area of his sculpture.

Input

The input consists of multiple datasets, followed by a single line containing only a zero. The first line of each dataset contains an integer n indicating the number of the following lines, each of which contains two integers a_i and b_i $(i = 1, \cdots, n)$.

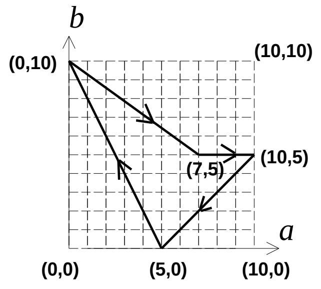

Figure 15: Outline of the cross section

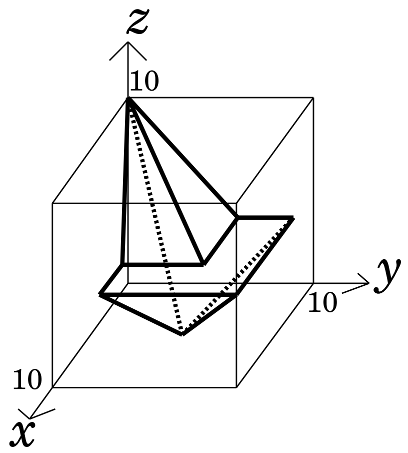

Figure 16: The intersection

A closed path formed by the given points (a_1, b_1), (a_2, b_2), $\cdots$, (a_n, b_n), $(a_{n+1}, b_{n+1})(= (a_1, b_1))$ indicates the outline of the cross section of the prisms. The closed path is simple, that is, it does not cross nor touch itself. The right-hand side of the line segment from (a_i, b_i) to (a_{i+1}, b_{i+1}) is the inside of the section.

You may assume that $3 \le n \le 4$, $0 \le a_i \le 10$ and $0 \le b_i \le 10$ $(i = 1, \cdots, n)$.

One of the prisms is put along the x-axis so that the outline of its cross section at $x = \xi$ is indicated by points $(x_i, y_i, z_i) = (\xi, a_i, b_i)$ $(0 \le \xi \le 10,\ i = 1, \cdots, n)$. The other prism is put along the y-axis so that its cross section at $y = \eta$ is indicated by points $(x_i, y_i, z_i) = (a_i, \eta, b_i)$ $(0 \le \eta \le 10,\ i = 1, \cdots, n)$.

Output

The output should consist of a series of lines each containing a single decimal fraction. Each number should indicate an approximate value of the surface area of the polyhedron defined by the corresponding dataset. The value may contain an error less than or equal to 0.0001. You may print any number of digits below the decimal point.

Sample Input

```
4
5 0
0 10
7 5
10 5
4
7 5
10 5
5 0
0 10
4
0 10
10 10
10 0
0 0
3
0 0
0 10
10 0
4
0 10
10 5
0 0
9 5
4
5 0
0 10
5 5
10 10
4
0 5
5 10
10 5
5 0
4
7 1
4 1
0 1
9 5
0
```

Output for the Sample Input

```
194.8255
194.8255
600.0000
341.4214
42.9519
182.5141
282.8427
149.2470
```