New Zealand Programming Contest 2020

Problem Set

PREAMBLE

Please note the following very important details relating to input and output:

- Read all input from the keyboard, i.e. use `stdin, System.in, cin, Console.ReadLine` or equivalent. Input will be redirected from a file to form the input to your submission.
- Do NOT prompt for input as this will appear in your output and cause a submission to be judged as wrong.
- Write all output to the screen, i.e. use `stdout, System.out`, `cout, Console.WriteLine` or equivalent. Do not write to `stderr`. Do NOT use, or even include, any module that allows direct manipulation of the screen, such as `conio, Crt` or anything similar.
- Output from your program is redirected to a file for later checking. Use of direct I/O means that such output is not redirected and hence cannot be checked. This could mean that a correct program is rejected! You have been warned.
- Unless otherwise stated, all *integers* will fit into a standard 32-bit computer word. If more than one integer appears on a line, they will be separated by spaces.
- An *uppercase letter* is a character in the sequence 'A' to 'Z'. A *lower case letter* is a character in the sequence 'a' to 'z'. A *letter* is either a lower case letter or an upper case letter.
- Unless otherwise stated, a *word* or a *name* is a continuous sequence of letters.
- Unless otherwise stated, a *string* is a continuous sequence of visible characters.
- Unless otherwise stated, words and names will contain no more than 60 characters, and strings will contain no more than 250 characters.
- If it is stated that ‘a line contains no more than *n* characters’, this does not include the character(s) specifying the end of line.
- Input files are often terminated by a ‘sentinel’ line, followed by an end of file marker. This line should not be processed.

Problem A — Jean and Joe's Clothes — 3 Points

Jean and Joe are a rather untidy couple with a similar bogun taste in clothes. Their 2 children, Jane and James are also clothed in the style.

Jean's mother came to stay and was horrified at the untidy state of the house. She formed a pile of clothes for each of the 4 people in the house, but could only do so by looking at the size of each item. Unfortunately sometimes the size had been cut off or wasn't readable, so these clothes she put in a separate pile.

If the size is 'M' or 'L', then it is Joe's.

If the size is 'S' then it is James'.

If it is 12 or greater then it is Jean's.

If it is smaller than 12 it is Jane's.

Jean's mother makes frequent visits and has to go through this process every visit. Your task in this problem is to write a program that will assist her.

Input

Input consists of data for a single visit by Jean's mother. The first line will consist of a whole number, N (0 < N <= 50), which is the total number of clothes found strewn round the house. Then follow N lines each representing the size of an item of clothing, or an 'X' if the size label is missing or unreadable. The sizes will either be a 1 or 2-digit number or a letter 'S', 'M' or 'L'.

Output

Output consists of one line which will contain five numbers, each separated by a space. The numbers represent the number of clothes belonging to Joe, Jean, Jane and James respectively in the current visit. The final number is the number of clothes unable to be assigned to anyone. If there are no clothes in a pile, the number 0 must be shown.

[**Turn over for sample input and output**]

Sample Input

```
8
M
12
X
14
10
L
S
S
```

Output for Sample Input

```
2 2 1 2 1
```

Explanation

```
Joe has 2 (M and L)

Jean has 2 (12 and 14)

Jane has 1 (10)

James has 2 (S and S)

There is one unreadable (X)
```

Whenever somebody goes to an ATM to withdraw or deposit money, a calculation has to be done to keep the person's bank balance correct. Your task in this problem is to do such calculations. There is a bank rule that says that a customer may not have an overdraft of more than $200, so any withdrawal that would take the balance below -200 must be stopped. (A minus sign is used to indicate an overdraft, or negative balance). There is no upper limit for an account balance.

Input

Input consists of a number of lines, each representing a transaction. Each transaction consists of an integer representing the starting balance (between -200 and +10,000), the letter W or the letter D (Withdraw or Deposit), followed by a second integer representing the amount to be withdrawn or deposited (between 5 and 400). Input will be terminated by a line containing 0 W 0.

Output

Output consists of one line for each line of input showing the new balance for each transaction, except that if a withdrawal would take the balance below -200, the output must be the words Not allowed

Sample Input

```
100 W 10
-200 D 300
50 W 300
0 W 0
```

Output for Sample Input

```
90
100
Not allowed
```

A nasty virus has infected my computer. Its effect has been to attack all my text files and reverse every word in them. Your job in this problem is to write the code to restore my text files to their original condition.

As far as the virus was concerned, a word was any sequence of characters that ended with a space or an end of line character. You will see what I mean when I tell you that the first line in one of my files was:

```
Get ready for the New Zealand Programming Contest on 26th September.
```

The virus turned this into:

```
teG ydaer rof eht weN dnalaeZ gnimmaroorP tsetnoC no ht62 .remetpeS
```

See how it included the full stop with September and put it before the letters?

Input

Input will consist of a single line containing from 1 to 250 characters, which will start and end with a non-white space character. There will not be any tab characters. There will be only 1 space between words or between a sentence terminator (. ? or !) and the next sentence.

Output

Output will consist of one line where each word (as defined in paragraph 2 above) in the input line has been reversed.

Sample Input 1

```
teG ydaer rof eht weN dnalaeZ gnimmaroorP tsetnoC no ht62 .remetpeS
```

Output for Sample Input 1

```
Get ready for the New Zealand Programming Contest on 26th September.
```

Sample Input 2

```
I ekil .maerc-eci
```

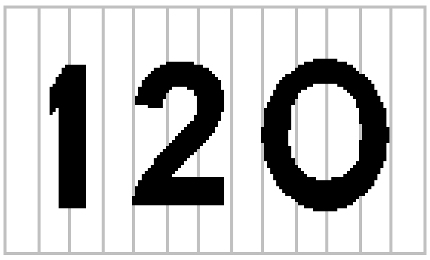
Output for Sample Input 2

```
I like ice-cream.
```

Problem D House Numbers 3 points

As the manufacturer of holders for door numbers, you have to know how wide to make them based on the house number each of your customers supplies. Fortunately, all house numbers in your area use exactly the same style of digit, so the calculation is quite easy. Most digits are exactly 3 cm wide, but a 0 is 4 cm wide and a 1 is only 2 cm wide, so that adds a slight complication. Also, you have to remember to leave a 1 cm gap between digits, and a 1 cm border at the start and end of the number.

As you can see from the diagram, a customer who lives at number 120 will need a 13 cm wide holder – 4 cm for the border and gaps, 2 cm for the 1, 3 cm for the 2 and 4 cm for the 0. 4 + 2 + 3 + 4 = 13.



Input

Input for this problem is a series of street numbers (integers between 1 and 9999 inclusive) each on a line of its own. The last number will be 0 which should not be processed.

Output

For each input line, output a single integer, the width in cm of the required holder for that street number.

Sample Input

```
120
5611
100
0
```

Output for Sample Input

```
13
15
14
```

Becs and Cas are best friends. Cas left school when she was 16 as she has total fashion shopping skills, and became a manager at Supre. So, she and Becs get their clothes from Supre - and in fact they have always bought exactly what the other has (they are that shallow), but they would simply die if they wore the same thing on the same day.

The two friends are so into copying each other that they have exactly the same outfits in their wardrobes hanging in exactly the same order. However, sometimes one of them just hates an outfit, so throws it out. (For the sake of friendship she doesn't tell the other and if she's ever asked about it by text she pretends it is at the cleaners).

One difference between the girls is that Becs numbers her outfits from left to right, so outfit 1 is the left most outfit, whereas Cas numbers them from right to left, so 1 is the right most outfit. Their wardrobes look the same, though, so, for example, the left most outfit in each wardrobe is the same.

Your task in this problem is to write a program that would alert Becs and Cas in advance if they choose the same outfit for a particular day.

Input

Input starts with two integers, n and d. n (5 < n <= 50) represents the number of outfits in each girl's wardrobe before any are thrown out. d represents the number of days to be considered.

The next 2 lines show outfits that have been removed from the girl's wardrobes. Each line will contain a single integer, r (0 <= r <= n). A value of 0 means no outfit has been removed. Any other value means the girl has removed that numbered outfit (according to her numbering system) from her wardrobe. The first line will refer to Becs' wardrobe, the second to Cas'.

There then follow d lines, each representing the outfits chosen by each girl on a particular day, separated by a space with Becs' choice first. For example:

```
3 12
```

means that Becs chose outfit 3 counting from the left of her wardrobe, Cas chose outfit 12 counting from the right of her wardrobe.

Output

Output consists of one line for each day. Each day will have a day number followed by OK if the girls have chosen different outfits, ALERT if they have chosen the same one.

Turn over for sample input and output.

Sample Input 1

```
20 4
0
0
1 3
6 15
20 1
10 10
```

Output for Sample Input 1

```
Day 1 OK
Day 2 ALERT
Day 3 ALERT
Day 4 OK
```

Sample Input 2

```
12 3
7
0
7 5
3 10
7 6
```

Output for Sample Input 2

```
Day 1 ALERT
Day 2 ALERT
Day 3 OK
```

You probably know about HTML, the mark up language used to create Web pages. HTML code contains a number of tags which are expected to follow certain rules.

In this problem we will be concerned with two of these rules:

1 Every opening tag has to have a corresponding closing tag

2 All tags must be properly nested.

Tags are marked by angled brackets which contain a keyword, such as `<body>` or `<strong>`. These are opening tags, the corresponding closing tags having `/` before the keyword, ie `</body>` and `</strong>`. It is possible for a tag to be both opening and closing, such as `<br />`, which complies with rule 1.

A keyword is a single lower case word with no spaces.

To be properly nested, if a tag is opened inside another tag, it must be closed before the other tag closes. For example

`<body> <strong> </strong> </body>` is properly nested

`<body> <strong> </body> </strong>` is not, and breaks rule 2.

Notes

If there are no tags present, the text complies with both rules.

Attributes may be present within an opening tag, such as

```
<a href="http://www.nzprogcontest.org.nz">This is a link</a>
```

The closing tag has only to match the keyword, not the attributes.

Input

Input will consist of a number of lines of HTML code, each line containing from 0 to 255 characters. The last line will contain a single # character – do not process this line. Within the text of each line will be zero or more tags. No angle bracket will be present unless it is part of a properly formed tag.

Determine whether or not the HTML meets the rules specified above.

Output

Output will consist of a single line for each line of input. The line will contain either the word `legal`, or the word `illegal.`

[**Turn over for sample data**]

Sample Input

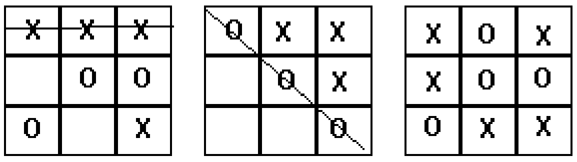
```
<body> <strong>Oops, this is</body> naughty </strong>
<body> <strong>Hello</strong> <br /> </body>
Just text, no tags.
<p> Oh dear, we are missing something.
<a href="http://www.nzprogcontest.org.nz">This is a link</a>
#
```

Output for Sample Input

```
illegal
legal
legal
illegal
legal
```

PROBLEM G **NOUGHTS AND CROSSES** **10 POINTS**

Noughts and crosses is a simple game played by 2 people. The aim is to get a row of 3 of your own symbols (vertically, horizontally or diagonally) before your opponent can do so. For example:



1 2 3

Game 1 shows X winning with a horizontal row. Game 2 shows O winning with a diagonal row. Game 3 is a draw as neither side has completed a row.

In this problem you will be given one or more games to follow and must determine who has won.

Input

Each game occupies a single line and begins with a letter, X or O, which shows who takes the first turn. Input is terminated by a line which contains just the character # - this line should not be processed.

A number of turns follow the initial letter, on the same line, all separated from each other by a space. A turn is designated by one of the numbers from 1 to 9, numbers representing board squares as shown below:

1	2	3
4	5	6
7	8	9

No number will be repeated. Games will stop if one player wins.

Game 1 above could have been represented as

```
X 1 5 9 7 3 6 2
```

Output

For each game in the input, 1 line of output should be given. If there is a winner, the output should be an upper case X or an upper case O as appropriate. If there is no winner, the output should be the word Draw.

[**Turn over for sample input and output**]

Sample Input

```
X 1 5 9 7 3 6 2
X 3 5 6 9 2 1
O 1 5 9 2 8 7 3 6 4
#
```

Output for Sample Input

```
X
O
Draw
```

The numbers 220 and 284 are called Amicable Numbers because they are not equal, and for each, the sum of its divisors (not counting itself) equals the other number. That is, the sum of the divisors of 220,

1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 equals 284,

and the sum of the divisors of 284,

1 + 2 + 4 + 71 + 142 equals 220.

In this problem, you have to find pairs of amicable numbers in a given range.

Input

Input begins with a single positive integer, N, the number of ranges to evaluate. This is followed by N lines each containing two positive integers, L and U ($0 < L < U < 20000$).

Output

For each input number pair, output a line with the text `Amicable numbers between L and U`, giving values for L and U. Follow that with a line for each distinct amicable pair discovered, giving the pair of numbers, separated with a space. Amicable pairs should be output with the lower of the two numbers displayed first. Pairs should be in ascending order of their lower values. If there are no amicable pairs, display 'None'. Remember, a number cannot form an amicable pair with itself.

Sample Input

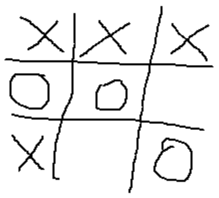
```
2
1 100
150 300
```

Output for Sample Input

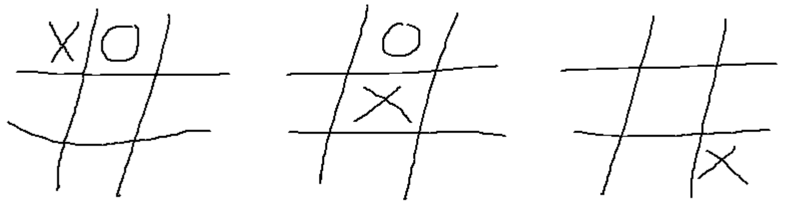
```
Amicable numbers between 1 and 100
None
Amicable numbers between 150 and 300
220 284
```

Problem I **Extreme TTT** **30 Points**

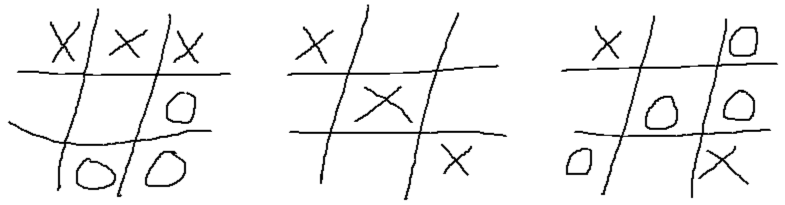
Tic Tac Toe, (Naughts and Crosses) is a simple pencil and paper game played by children. The idea is that players take turns at drawing their symbol (O or X) in squares on a 3 by 3 grid. The first to get three of their symbols in a row (horizontal, vertical or diagonal) wins the game. If the grid is filled without either player making a line, the game is a draw. Here is a game that the X player has just won.



Unfortunately the game is rather limited. An interesting extension is to make it three dimensional – using 3 layers of 3 by 3. Real 3D requires some construction – plastic sets are available commercially. However it is possible to play a 3D game on paper, drawing three 2D grids and imagining them piled above each other. X wins again:



What about playing in 4, 5 or 6 dimensions? This is the basic idea of the game of Extreme Tic Tac Toe (ETTT), a game played by AI computers. ETTT scoring differs from normal Tic Tac Toe. Rather than stop as soon as one player achieves a line of their symbols, ETTT players play until the grid is full, or until they agree to stop. The winner is the one with the greatest number of lines of symbols. By ETTT rules, X would score 4 and O would score 1 in the following 3D game:



Your task is to write a program to read N-Dimensional ETTT game grids and work out the scores.

Input Specification

The input consists of a series of input ETTT board configurations. Each board configuration starts with a line holding N, the dimension of this game (1 <= N <= 7). End of input is signaled by an N value of zero. That is followed by lines of X's, O'x and ~'s for the cells of the game (~ represents an empty cell). Each line holds at least one, and no more than 40 symbols. To understand the order in which data is input, imagine the board being held in an N dimensional array. With N = 5, for example, the board could accessed as cell[a,b,c,d,e] or (cell[a][b][c][d][e] if your language doesn't support the nicer syntax). The following pseudo-code would read the data in the correct order (ignoring line breaks).

```
for a = 1 to 3
   for b = 1 to 3
      for c = 1 to 3
         for d = 1 to 3
            for e = 1 to 3
               read cell[a,b,c,d,e]
```

Output Specification

For each board configuration, output the scores for X and for O as shown in the sample data.

Sample Input

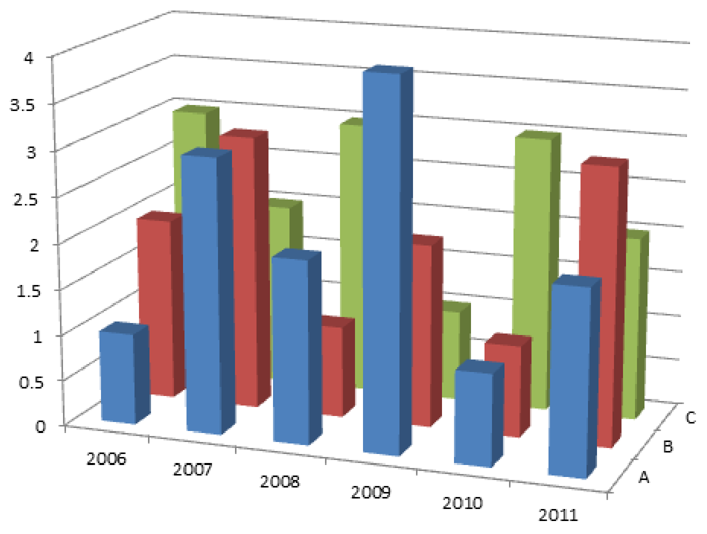
```
2
XXX
OO~
X~O
3
XXX
~~O
~OO
X~~~X~~~X
X~O
~OO
O~X
0
```

Sample Output

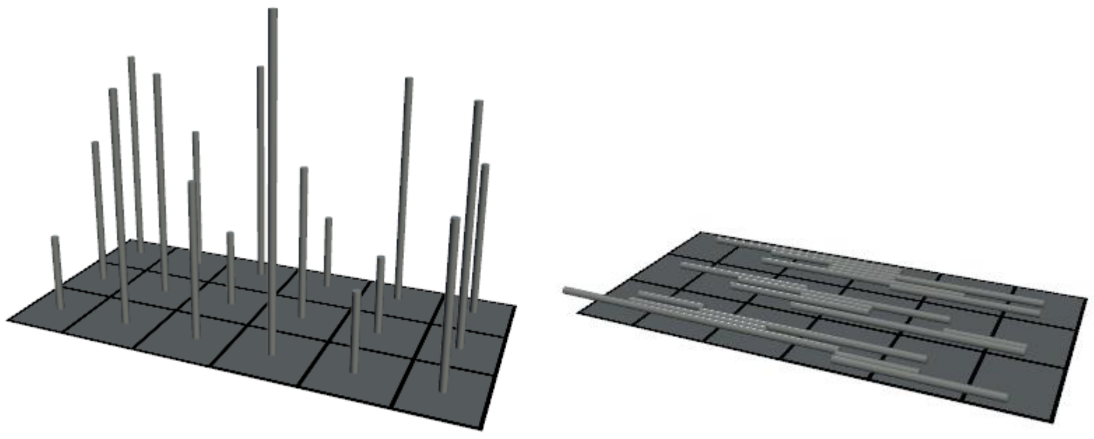
```
X scores 1 and O scores 0
X scores 4 and O scores 1
```

Problem J **Standing Pins** **30 points**

For a special presentation to the Computer Game Developers association Mario made up a two dimensional histogram of showing number of game titles published by year and game genre. As displayed by his spreadsheet software it was quite attractive.

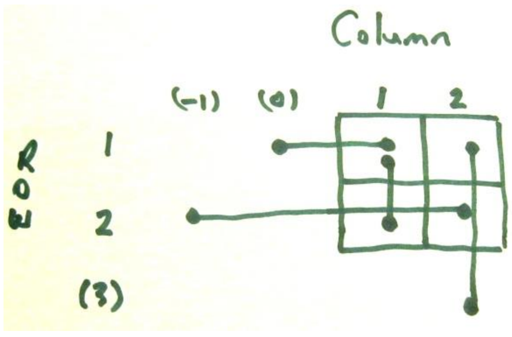


Mario decided that a physical model would be even better and built one from wire and cardboard – see left picture below. When packed for transport to the conference it looked like the right hand picture below. Each wire had been laid down, either to the left or right at random, while keeping its base in the correct square.

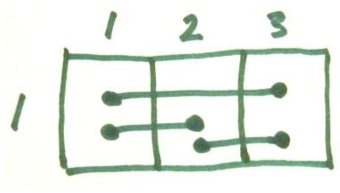


Only when he came to unpack and get his model set up again did he realise that there might be a problem. Whether each wire had been laid down to the left or to the right had not been recorded. Even worse, his assistant had not followed the packing instructions correctly. Not all wires had been laid left or right. Some had been laid forward and backward on the card. Of course the original data was not available. You have been asked to write software to figure out how to stand the wires up. In case the problem occurs again, you have been asked to make the software quite general.

Some problems will have solutions. For example, in the sketch to the right there are four pins of lengths 1, 1, 3 and 2. They can only stand up with the two length 1 pins in column 1; and the length 2 and 3 pins in column2 (rows 1 and 2 respectively).



Other problems will not have unique solutions. For example consider the second sketch. The three pins can be stood up with the length 2 pin either at the right or at the left. When there are multiple solutions in this way we cannot be sure as to which is correct, therefore we must state that there is no solution. Note however that it would not have been a problem if the length 2 pin was only 1 unit long. In that case we might not be sure which cell pins had originally occupied, but we would be sure that each cell had started with a pin of height one. That is ok.



Input

The input consists of a number of problems. Each problem starts with a line holding two numbers R, and C, the number of Rows and the number of columns of the grid. 1 <= R, C <= 100. This will be followed by R * C lines. Each line will hold the grid coordinates of the two ends of a piece of wire as four numbers r1, c1, r2, c2. One end will be in its correct grid cell. The other end will be wherever its length dictates, as it was either horizontally or vertically laid down. Note that the 'other end' coordinates may lie outside the grid (see examples below). Wires always have integer lengths. Wire lengths lie in the range 1 .. 9 (inclusive). Input is terminated by a line with two zeroes.

Output

For each problem output one blank line. Then, for problems for which there is no unique solution, output "No solution". For problems with a unique solution you should output R rows. Each row will have C numbers, giving the heights of the wires in each column. These numbers will be output without spaces.

Sample Input

```
2 2
1 0 1 1
1 1 2 1
2 -1 2 2
1 2 3 2
1 3
1 1 1 3
1 1 1 2
1 2 1 3
0 0
```

Sample Output

```

12
13

No solution
```

Problem K **Dictionary Attack** **30 points**

One of the simplest forms of cypher is the deranged alphabet substitution cypher. The idea is that we rearrange the letters of the alphabet in some arbitrary (deranged) order, and write the alphabet in its normal order underneath.

```
p o n m t s r q w j u z y x d c b a h g f e l k v i
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

To cypher a message we replace each letter in the message with the one written above. So "`all is well`" will become "`pzz wh ltzz`". Note that there is no rule to prevent some letters being substituted as themselves. In the example above j is cyphered as j.

This is not a strong cypher, and there are many approaches to breaking it. A particular weakness in this case is the fact that the spaces between words have been preserved. This allows us to identify words, and know immediately the length of each word. The method of attack you will implement in this problem is based on this fact. It is a kind of dictionary attack.

Your task is to take a dictionary (a list of all possible words that might be used in a message), and use it to find all possible interpretations of an enciphered message. Be aware that the dictionary might be quite large (one supplied with the test set has 58,111 words).

For example, given a dictionary with just the four words: `aab` `bbc` `abb` `cde`
The phrase: `zzq qqt` can be translated as `aab bbc`
The phrase: `prt` can be translated as `cde`
The phrase: `zzr` can be translated either as `aab` or as `bbc`

Input

Input consists of a series of problem sets. Each problem set consists of a dictionary and a number of cypher texts to translate. The first line of the dictionary holds an integer 0 < N < 60000, the number of words in the dictionary. It is followed by N lines, each holding one word. A word is entirely in lower case letters with no special characters. No word is longer than 22 letters. Next is a line with a number C > 0 of texts to be translated, followed by C lines each holding a text. Texts are single spaced sequences of 'words'. No text is longer than 200 characters. End of input is signalled by a line with the number zero.

Output

For each text to be translated, output a line with 'Problem n' where n = 1, 2, 3, 4… Follow this with the possible translations you have found, one per line, lines being in alphabetic order. It is possible that some texts will have no translations. In that case you output the 'Problem n' header, but have no lines following it. The problem numbering should continue across problem sets.

Sample Input

```
4
aab
bbc
abb
cde
3
zzq qqt
prt
zzr
1
soup
1
abcd
0
```

Sample Output

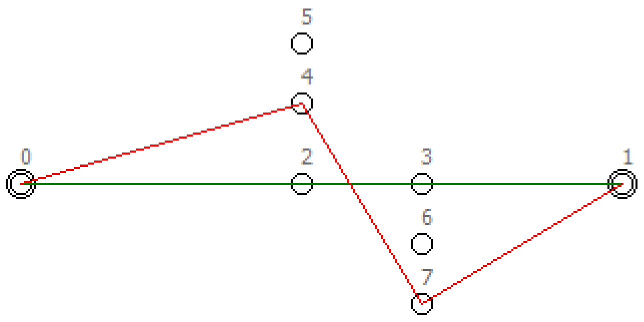
```
Problem 1
aab bbc
Problem 2
cde
Problem 3
aab
bbc
Problem 4
soup
```

Problem L **Frogs With Style** **30 Points**

It is nearly time for the annual NZPC (New Zealand Pond Competition). Frogs from around the country gather to compete at jumping from lily pad to lily pad across competition ponds. The competition is about speed and style. As the time taken per jump is essentially independent of jump distance, speed can be maximised by following paths which involve the smallest number of jumps. Style is about taking only large jumps. Nothing looks worse than a finely trained competition frog taking a short jump. Freddo, last year's champion, has asked you to write a program which, given a lily pad placement map and knowledge of a frog's maximum jump range as input, will determine a best path for that frog to follow from start pad to finish pad.

The best path has a minimum number of hops – for example, every 3 hop path is better than any 4 hop path. Having found the minimum number of hops for a given pond, we must choose the best of the minimum hop paths. Of all minimum hop paths the best one is the one with the largest value for its shortest hop.

Consider the following pond layout. (This is the layout given in sample input, so numeric details are available there.) Pads 0 and 1 are the start and end respectively. Note that the path 0, 2, 3, 1 is the shortest path, but that it involves a particularly short jump from pad 2 to pad 3. The best competition path for this layout turns out to be 0, 4, 7, 1.



Input

Input consists of a sequence of problems. For each problem the first line of input holds two integers P and D. P is the number of lily pads on the pond and D is the maximum distance that the frog can jump. Following are P lines, one for each lily pad, each with two floating point numbers X and Y being the coordinates of the centre of the pad. (0 < D,X,Y < 1000, 2 <= P <= 200) Note that all jumps are from centre to centre – an off centre launch or landing leads to immediate disqualification. The first two pads in the input for each problem are the starting and ending pads for the competition respectively. Input is terminated with a P D line of 0 0.

Output

For each problem, output one line showing the number of hops required and the length of the shortest hop in the best path – this value should be rounded and displayed to exactly one decimal places. The two numbers should be separated by a single space. Note that the problems have been checked to ensure that the optimum result will round cleanly. Note also that the best path might not be unique. In that case, output the values from any best path.

Sample Input

```
8 150
100 100
400 100
240 100
300 100
240 60.0
240 30.0
300 130
300 160
0 0
```

Sample Output

```
3 116.6
```

Problem M **Stars** **100 points**

On a clear moon-less night, you can see millions of stars glimmering in the sky. Faced with this overwhelming number, the Greeks started nearly 2,000 years ago to bring some order to the chaos. They identified groups of stars, called constellations, and gave them names that are still in use today. Examples are "Ursa Minor", "Pisces", "Cancer", etc.

Given a sketch of the constellation, it is not easy for an amateur to actually find the constellation in the sky. Moreover, the shapes of simple constellations, such as "Triangulum" (triangle), which consists of only three stars, may appear several times in the sky. Singling out the "correct" occurrence is not easy.

Traditionally, maps were printed for just this purpose. In this problem, we will see how the computer can help us find constellations in the sky.

You will be given a star map; for simplicity this will be a collection of points in the plane, each having a certain brightness associated with it. Then, given a “constellation”, also as a set of points in the plane, you are to determine:

- the number of occurrences of the constellation in the star map, and
- the position of the brightest occurrence, if one exists. (The rationale behind this is as follows: if a constellation seems to appear several times in the sky, the brightest one is most likely to be the real one, since it is the most eye-catching one.)

An occurrence is a subset of stars from the map that forms a (possibly) arbitrarily rotated and/or scaled copy of the stars in the constellation. (Note that a given subset of stars may fit a constellation in several orientations – a square will fit 4 ways at 90 degree rotations. These do not count as distinct occurrences.)

The brightness of an occurrence is the average brightness of the stars it consists of, i.e. the sum of individual brightnesses divided by the number of stars in the constellation.

Input Specification

The input file contains the descriptions of several star maps. Each map starts with a line containing a single integer N, specifying the number of stars in the map (1 <= N <= 1000). The following N lines contain three integers each, namely the X- and Y-coordinates and the brightness of every star. The larger the value, the brighter the star shines.

The next line contains a single integer M, the number of constellations to follow (1 <= M < 50). Each constellation description starts with a line containing an integer S, the number of stars in the constellation, and a string C, the name of the constellation. (C will consist of no more than 40 characters and contain no blanks.) The following S lines then contain the coordinates of the constellation, again as X/Y-pairs.

All coordinates lie in the range -1000 to 1000; brightness levels lie in the range 0 to 100.

A blank line separates a star map from the next map. The input file ends with an empty map (having N = 0), which should not be processed.

N.B.: Since all star coordinates are integers, you can easily rule out any rotated or scaled constellation whose points do not fall on integer coordinates.

Output Specification

For each star map first output the number of the map (`Map #1', `Map #2', etc.) on a line of its own.

For each constellation, in the same order as in the input, output first its name and how many times it occurs in the map on one line, as shown in the output sample.

If there is at least one occurrence, output the position of the brightest occurrence by listing the positions of the stars that form the brightest occurrence. The star positions should be printed in ascending x-order. Positions having the same x-coordinates must be sorted in ascending y-order. You may assume that star maps always have a single brightest occurrence for each constellation. Adhere to the format shown in the sample output.

Output a blank line before each constellation and a line of 5 dashes (`-----') after every star map.

Sample Input

```
6
1 2 1
2 1 4
2 4 3
3 2 1
4 1 5
4 3 2
2
3 Triangulum
1 1
3 1
2 4
4 Cancer
1 3
4 3
6 1
7 5

0
```

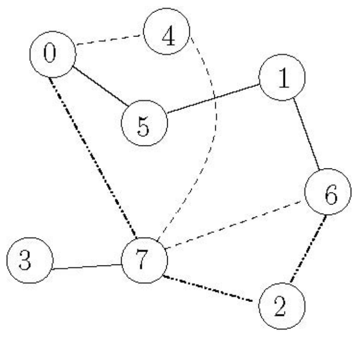
Output for Sample Input

```
Map #1

Triangulum occurs 2 time(s) in the map.
Brightest occurrence: (1,2) (4,1) (4,3)

Cancer occurs 0 time(s) in the map.
-----
```

PROBLEM N CONGESTED NETWORKS 100 POINTS

Given a connected computer network (bidirectional communication) we want to find two different nodes u and v such that we can maximize the congestion between u and v with a continuously sent virus being sent between the pair. We define the congestion level as the maximum number of edge-disjoint paths between nodes u and v.

For example, the network shown in the following figure has three different paths between nodes 0 and 6 such that each edge used is only part of one connecting path. Note that two paths are allowed to go through the same node, such as node 7. No other pair of nodes provides a higher congestion level.



Input Specification

We will be given a sequence of connected computer networks each with n nodes, n ≤ 40, labeled {0, 1,...,n − 1}. The last input case will be followed by a network of n =0 nodes, which should not be processed.

The specification for a computer network will be as follows: the first line contains a single non-negative value n, denoting the number of nodes. This is then followed by n lines of integers, separated by spaces, denoting the neighbors (zero indexed) of each node. Expect up to 2000 test cases.

Output Specification

For each input case output one integer on a line by itself denoting the maximum congestion level possible for some pair of its network nodes.

[Turn over for sample data]

Sample Input

```
8
4 5 7
5 6
6 7
7
0 7
0 1
1 2 7
0 2 3 4 6
4
1 2
0 2
3 0 1
2
0
```

Output for Sample Input

```
3
2
```

Problem O **Jury Compromise** **100 points**

In Frobnia, a far-away country, the verdicts in court trials are determined by a jury consisting of members of the general public. Every time a trial is set to begin, a jury has to be selected, which is done as follows. First, several people are drawn randomly from the public. For each person in this pool, defence and prosecution assign a grade from 0 to 20 indicating their preference for this person. 0 means total dislike, 20 on the other hand means that this person is considered ideally suited for the jury.

Based on the grades of the two parties, the judge selects the jury. In order to ensure a fair trial, the tendencies of the jury to favour either defence or prosecution should be as balanced as possible. The jury therefore has to be chosen in a way that is satisfactory to both parties.

We will now make this more precise: given a pool of n potential jurors and two values d_i (the defence's value) and p_i (the prosecution's value) for each potential juror i, you are to select a jury of m persons. If J is a subset of {1,...,n} with m elements, then $D(J) = \Sigma_{k \text{ in } J}\, d_k$ and $P(J) = \Sigma_{k \text{ in } J}\, p_k$ are the total values of this jury for defence and prosecution.

For an optimal jury J, the value |D(J) - P(J)| must be minimal. If there are several juries with minimal |D(J) - P(J)|, one which maximizes D(J) + P(J) should be selected since the jury should be as ideal as possible for both parties. Finally, if there is more than one optimal jury under this combined condition, the jury who's list of candidate numbers comes first in 'pseudo alphabetic' order should be chosen. By 'pseudo alphabetic' order we mean order as though the candidate numbers were letters. For example 1,5,6,9 comes before 2,3,4,5 because 1 < 2; 1,2,3,5,9 comes before 1,2,3,6,9 because 5 < 6; etc.

You are to write a program that implements this jury selection process and chooses an optimal jury given a set of candidates.

Note: If your solution is based on an inefficient algorithm, it may not execute in the allotted time.

Input Specification

The input file contains several jury selection rounds. Each round starts with a line containing two integers n and m. n is the number of candidates and m the number of jury members. These values will satisfy 1 <= n <= 200, 1 <= m <= 20 and of course m <= n. The following n lines contain the two integers p_i and d_i for i = 1,...,n. A blank line separates each round from the next.

The file ends with a round that has n = m = 0.

Output Specification

For each round output a line containing the number of the jury selection round (`Jury #1', `Jury #2', etc.).

On the next line print the values D(J) and P(J) of your jury as shown below and on another line print the numbers of the m chosen candidates in ascending order. Output a blank before each individual candidate number.

Output an empty line after each test case.

Sample Input

```
4 2

1 2

2 3

4 1

6 2


0 0
```

Output for Sample Input

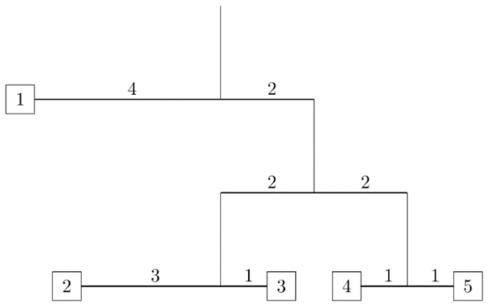
```
Jury #1

Best jury has value 6 for prosecution and value 4 for defence:

 2 3
```

PROBLEM P **MOBILE** **100 POINTS**

Mobile is an oldie from a contest far far away.

You've probably seen mobiles suspended from the ceilings of museums or airports. We'll restrict ourselves to the type suspended from the ceiling by a single wire that is attached to a pivot point on an arm (also made of wire). At each end of the arm is either another wire suspending yet another arm, or a weight (usually in the form of some design). To the right is one example, made by Alexander Calder, the best-known mobile artist.

Some mobiles are simple and some are quite complex. Besides the artistry, these must balance. Recall that from a pivot point distance dL from the left and dR from the right, an arm will balance if the product of the weight at the left end and dL is equal to the product of the weight at the right end and dR. (We ignore the weight of the arm and the wires suspending the arms.)



For example, consider the mobile drawn to the right. If weight 1 weighs 8 units, then weights 2, 3, 4, and 5 must weigh 2, 6, 4, and 4 units respectively. In fact, if you know the structure of the mobile, that is, the arrangement of arms and where the pivot points are on each arm, and the value of one weight, you can determine the values of all the weights. That is your problem here – almost. It seems you only have weights that are integer valued. So, you'll be given the desired minimum value of one weight and determine the value of the other weights, so that those values will also be integers. Thus, it's possible that the specified minimum valued weight must be raised a little bit to accomplish this.

Input

Input for each test case will start with a line containing the positive integer n, indicating the number of arms in the mobile. These arms are numbered 1 through n. The next n lines will describe the arms, in order 1,2,...,n, and will be in the form

```
dL  dR  typeL  typeR  nL  nR
```

where dL and dR are integers ≤ 20 giving the distances from the pivot point to the left end and right end of the arm, typeL and typeR are each either W or A, indicating that a weight or arm hangs from the left or right ends, and nL and nR are the index numbers of the weight or arm on the left and right. The indices for the weights will start at 1 and be consecutive. The mobile will not have an arm that is hanging further down than 6 arms from the top. In our example above the lowest arm is 3 arms from the top. Following the description of the arms is a line of the form m w, indicating that weight m weighs at least w, where $1 \le w \le 20$. A line containing a 0 follows the last test case.

Note: The weight indices are all distinct.

Output

For each test case output one line giving the minimum total weight of the mobile if weight m is at least w. Use the format given in the Sample Output. You may assume all output values will be less than 10^9.

Sample Input

```
4
3 1 W W 2 3
4 2 W A 1 3
2 2 A A 1 4
1 1 W W 4 5
1 8
4
2 2 A W 2 5
3 1 W A 4 3
4 1 W A 3 4
2 1 W W 1 2
3 20
0
```

Output for Sample Input

```
Case 1: 24
Case 2: 280
```