ANZAC 2015

Round 5

(Run in parallel with the New Zealand Programming Contest 2015)

Notes:

1. For all problems, your programs must read the input from stdin/keyboard and output to stdout/console.

2. Problems in this contest have different point values associated with them, as indicated on each problem sheet. The scoreboard and rankings will be based on each problem's points, rather than the normal 1-point-per-problem setting.

PROBLEM A **TALLEST** **3 POINTS**

When I was at school, we were always very competitive; who can run fastest? Who gets best marks in tests? This problem gets you to help with one such challenge – who is tallest?

Input

Input consists of a number of groups of people. The first line for each group contains a single integer, N, which is the number of people in the group. (0 < N <= 50). If N is 0, this marks the end of input.

N lines then follow, each containing data on one person, a single name followed by a height in metres which has 2 decimal places.

Output

For each group of people, output a single line containing the name of the tallest person. Where two or more people in a group have the same height, list each of them on the same line, separated by spaces, in the order they appear in the input.

Sample Input

```
3
John 1.75
Mary 1.64
Sam 1.81
2
Jose 1.62
Miguel 1.58
0
```

Output for Sample Input

```
Sam
Jose
```

NZPC
New Zealand
Programming Contest

PROBLEM B **PASSWORD** **3 POINTS**

Keepsafe is a tool used by KiwiBank to "provide an extra layer of security to help protect customers while using Kiwibank Internet Banking." Customers have to set up a number of security questions and answers. After logging in with a user name and password, a customer is presented with one of their security questions and a space for the answer. Two letters in the answer have to be entered as shown by white boxes. For example:

If the pet's name was "Whisper", the customer would press the P key then the R key. The letters pressed must correspond, in order, to the two spaces in the outline answer to the security question.

For this problem, you have to check customers' answers and verify that they are as expected.

Input

The first line of input contains a single integer, C (0 < C <= 50), which is the number of customers you have to process.

The data for each customer begins with a line containing a single integer, A (2 < A < 10), which is the number of security answers provided. Each answer consists of upper case letters and spaces only, with between 5 and 32 characters. The first answer is to security question 1 and so on.

The next line contains a single integer, L (0 < L <= 50), which is the number of login attempts you must process. A login attempt consists of 3 positive integers followed by 2 upper case letters, all separated by single spaces. The first integer is the number of the security answer required. The second and third integers are numbers of the characters required to fill the two spaces in the outlined answer. They are presented in ascending order. When numbering characters in the answer, the first character is 1, but only letters are assigned numbers, spaces being ignored. The two letters are those entered by the customer into the two spaces. They are expected to correspond to the letters in the appropriate places in the security answer required.

The 3 digits will all be valid ie between 1 and the maximum value possible in each case.

Output

For each customer in the input, output begins with a line containing the text `Customer N`, where N is 1 for the first customer and so on, customers being numbered in the order they appear in the input.

For each transaction for that customer, output one line which says either `correct` or `error`. If the two letters entered match the letters in the required places in the answer, in the order presented, then the output will be `correct`. If one or both letters are not correct, the output is `error`.

Sample Input

```
1
3
WHISPER
GOLDEN APPLE
THE ALL BLACKS
3
1 5 7 P R
3 4 9 A A
1 1 6 W P
```

Output for Sample Input	**Explanation**
`Customer 1`	
`correct`	Letters 5 and 7 in WHISPER are P and R.
`correct`	Ignoring spaces, letters 4 and 9 in THE**A**LLBL**A**CKS are both As.
`error`	Letters 1 and 6 in WHISPER should be W and E, not W and P.

NZPC
New Zealand
Programming Contest

PROBLEM C **NUMBERS** **3 POINTS**

Julia and Jeremy are twins who like to set each other challenges. Both have been learning about number systems, and are aware that we normally use decimal (base 10) numbers, probably because of the number of fingers we have.

The twins decided it would be fun to convert our numbers to number systems that other creatures may use, creatures who have a different number of digits or other appendages. See if you can rise to their challenge!

Input

Each challenge consists of 3 lines. The first line is a description of a creature (not necessarily real life). This will be no more than 20 characters in length. The final challenge begins with # - this challenge is not to be processed.

The second line of a challenge is the number of digits or appendages the twins assume the creature would use as a basis for its numbering system. This will be a number between 2 and 20 inclusive.

The final line is a decimal number which is to be converted into the assumed numbering system of the creature. This will be a positive integer no greater than 5,000.

Output

Each challenge requires a single line of output. It consists of the name of the creature, followed by a comma and a space, followed by the decimal number, followed by another comma and space, followed by the converted number. As is usual, where a base larger than 10 is used, upper case letters of the alphabet (A to J, in alphabetical order) are used as extra digits.

Sample Input

```
Three toed sloth
6
27
Octopus
8
126
Hexapus
16
397
#
```

Output for Sample Input

```
Three toed sloth, 27, 43
Octopus, 126, 176
Hexapus, 397, 18D
```

Explanation for Hexapus

In decimal numbers, each place represents 10x the value of the place to its right, so 1,000s, 100s, 10s and 1s. 397 is 3 x 100 + 9 x 10 + 7. The hexapus (whatever one of those is!) uses 16 digits, 0 to 9 and A to F. With 16 digits, the places are 16x the value of the place to its right, so 3,364s, 256s, 16s and 1s.

18D for a hexapus is 1 x 256 + 8 x 16 + D (which is 13 in decimal) which is 256 + 128 + 13 which is 397 in decimal numbers. In the problem, you have to work backwards and convert 397 decimal to 18D.

NZPC
New Zealand
Programming Contest

PROBLEM D **EXERCISING** **3 POINTS**

In an attempt to get herself fit, a friend of mine has bought herself a pedometer. She wears it when walking and it tells her how far she has walked.

The pedometer is sensitive to movement and is able to count steps reasonably accurately. Before using it, users have to measure the length of a pace and enter it into the pedometer, which can then use it to show how far has been walked. In this problem, you have to do this calculation.

Input

You will be given data for several pedometer users and asked to calculate how far each has walked. The first line of input for a person is a single integer L (10 < L < 100) which gives the stride length, in centimetres, of the person using the pedometer. This will be 0 for the last person whose data is not to be processed.

The next line for each user is a single integer N (1 <= N <=100) which is the number of entries for this person. N lines then follow, each containing a single positive integer no greater than 20,000. This is the number of steps the user has taken for the time period in question.

Output

For each user shown in the input, begin output with a single line of the format

```
User U
```

where U is the user number. Users are numbered from 1 in the order they appear in the input. This is followed by N lines, one for each entry showing the number of steps, which shows the appropriate distance walked in kilometers. This will be in the format

```
d.ddddd
```

ie at least one number before the decimal point and exactly 5 after it.

Sample Input

```
58
3
10653
8540
12305
46
2
8500
7945
0
```

Output for Sample Input

```
User 1
6.17874
4.95320
7.13690
User 2
3.91000
3.65470
```

NZPC
New Zealand
Programming Contest

PROBLEM E **PRICES** **10 POINTS**

The Golden Valley Clothing Warehouse has a lot of winter stock it wants to sell off quickly to make room for spring and summer clothing that is coming in soon. The manager came up with quite a complicated sale policy, which you now have the job of implementing.

Here are the rules the manager put in place:

Items are given a circular sticker with a colour – they are called "dots", hence red dot, yellow dot etc. Each dot represents a particular percentage discount as listed here.

Colour Dot	Discount %
Red	45
Green	30
Blue	20
Yellow	15
Orange	10
White	5

In addition, the manager has given out discount coupons! Any customer who presents one of these receives an extra 5% discount after the "dot" discounts have been calculated.

You have to work out the discounted price for each item presented. As your program has to run on the point of sale terminal, you must round the price to the nearest cent (0.5 is rounded up).

If a customer is paying cash, you must then round the price up or down to the nearest 10 cents. If the right most digit of the cents is from 0 to 5 inclusive, it is rounded down to 0, otherwise the cents are rounded up to the next 10.

Input

The first line of input will be a single positive integer, N, which is the number of purchases to process (0 < N <= 100). There will then follow N lines, each representing a single purchase. Each line will have the following format, items being separated by spaces.

`<original price> <dot> <coupon> <payment>`

`<original price>` The price of the item before any discount. A decimal number with 2 places of decimals.

`<dot>`	The dot colour, using a single upper case letter, the first letter of the colour.
`<coupon>`	C if the customer presented a discount coupon, X if not.
`<payment>`	C if the customer paid cash, P (for plastic) if they did not.

Output

For each purchase in the input, output a single line which is the discounted price. It must be in the format

```
$d.cc
```

i.e. a dollar sign, the dollar amount, a decimal point and the cents with 2 digits. If the dollar amount is 0, then the zero must be displayed.

Sample Input

```
4
29.99 R X P
119.95 W C P
68.50 B X P
69.90 Y C C
```

Output for Sample Input

```
$16.49
$108.25
$54.80
$56.40
```

NZPC
New Zealand
Programming Contest

PROBLEM F **CYPHER** **10 POINTS**

James Pond is a business man who often dreams he is a secret agent! He enjoys encrypting messages to his colleagues, who then have the task of decrypting them in order to read them. Your task is to write the decryption program for his staff to use.

Mr Pond uses the date as a key. He adds the day, month and year together, does a modulo 25 (remainder operator) on the answer and adds one to give him a value from 1 to 25 inclusive. This value, S, becomes the shift in his Caeser cypher.

In a Caesar cypher, each letter of a message is shifted S places forward through the alphabet, with z shifting to a where appropriate. For example, with a shift of 5, a becomes f, h becomes m and x becomes c. White space, punctuation and digits are not changed.

Input

Each message starts with the date as 3 integers on a single line, separated by spaces. A date of 0 0 0 marks the end of input.

The date is followed by a single line of at least 1 and no more than 250 characters; the line will not be just white space. Only lower case letters, spaces, punctuation marks and digits are used.

Output

For each message in the input, output a single line showing the decrypted message.

Sample Input

```
10 7 2015
uism aczm gwc izm zmilg nwz bpm kwvbmab wv 8bp.
12 5 12
n xmfqq gj ts mtqnifd sjcy bjjp xt rnxx rtsjditqqfw bnqq gj ns hmfwlj.
0 0 0
```

Output for Sample Input

```
make sure you are ready for the contest on 8th.
i shall be on holiday next week so miss moneydollar will be in charge.
```

NZPC
New Zealand
Programming Contest

Problem G **GRAPHS** **10 points**

Miss Fotheringale is teaching her maths class about plotting graphs. As a first exercise, she wants them to plot a linear graph **y = ax + b** for values of x from 0 to 10 inclusive.

She wants you to help by writing a program that outputs the correct answers to make it easier for her to mark her pupils' work.

Input

The first line of input is a single number N, 0 < N < 30, a positive integer that tells the number of graphs to be plotted.

N lines then follow, each representing one graph to be plotted. Each line contains 2 positive integers, A and B, 0 <= A, B <10, being the values of a and b for the graph.

Output

For each graph, firstly output the equation being plotted on a line by itself. Under this must be the graph. All points on the axes, plus points on the graph must be represented by the * character. The x axis must contain the points 0 to 10 from left to right, the y axis the points 0 to the maximum value from bottom to top.

Turn over for sample input and output

Sample Input

```
2
1 0
2 2
```

Output for Sample Input

```
y = 1x + 0
*          *
*         *
*        *
*       *
*      *
*     *
*    *
*   *
*  *
* *
**
***********
y = 2x + 2
*          *
*
*         *
*
*        *
*
*       *
*
*      *
*
*     *
*
*    *
*
*   *
*
*  *
*
* *
*
**
*
*
*
***********
```

PROBLEM H **CONTEST** **10 POINTS**

PC^2 (pea sea squared) has been used to manage programming contests for many years. It was written to manage the ACM contest where each problem solved scores 1 point, and time is the tie breaker. To use it in the NZPC, or any other contest where questions have a variable points weighting, a program needs to be written to read a PC^2 file and apply different rules.

That is what you have to do in this problem. You will be given a number of contests to manage. For each you will be given the points each problem is worth, and a PC^2 entry for each team. You have to rank the teams according to the rules of the contest in which they are competing.

Input

Input consists of a number of contests. The first line for each contest contains a single integer, P, which is the number of problems in the contest. (0 < P <= 20). Where P is 0, that marks the end of input.

The second line consists of a series of positive integers, one per problem, separated by spaces. This represents the number of points for each problem, the first number for the first problem and so on.

The third line for each contest contains a single integer, T, which is the number of teams taking part in the contest. (0 < T <= 50).

There then follow T lines, each line being data for one team. The data starts with the team name. This may contain spaces, but is terminated by a comma (with no commas in the name). There then follow one record for each problem, the records being separated by commas. The records are in the same order as were the points values in line 2, problem 1 first and so on.

Each record has an integer, a slash character, /, and either another integer, or a – character. The record should be interpreted as follows:

- The first integer shows how many submissions the team made for the problem. It will be 0 or more.
- If the slash is followed by a -, it means that the team did not solve the problem, so score no points for it.
- If the slash is followed by a number, which will be zero or more, it means that the team did solve the problem. The number shows how many minutes into the contest the successful submission was made.

Output

For each contest, output starts with a line containing the text `Contest N`, where N is the number of the contest. The first contest in the input is 1, and contests are numbered in order.

There then follows a line for each team in the contest showing rank, team name and points scored, separated by spaces. The teams must appear in ascending rank order (ie 1, 2, 3..).

In these contests teams are sorted by points scored, the higher the better. Teams with equal points are considered equal and should be displayed in alphabetical order of name (not case sensitive), but should have equal rank. Make sure any following team has its correct rank (for example after joint first, the next team is rank 3).

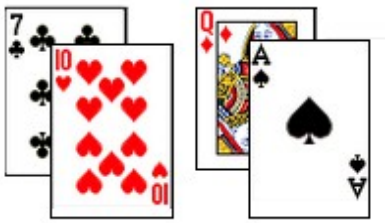
Sample Input

```
5
1 1 2 3 10
4
Happy Fred's team,2/5,3/27,1/106,8/262,3/-
Team 67,0/-,0/-,2/135,4/218,12/-
Astrix,0/-,0/-,0/-,0/-,17/299
apple pie,0/-,0/-,0/-,0/-,0/-
0
```

Output for Sample Input

```
Contest 1
1 Astrix 10
2 Happy Fred's team 7
3 Team 67 5
4 apple pie 0
```

PROBLEM I **BLACK JACK** **30 POINTS**



Black Jack is a card game where the aim is to obtain a hand of cards with a value of no more than 21. The game is played in casinos where players bet money on the result, but this is a friendly version with no money at stake. Instead we are counting the number of wins for each player.

There are 52 cards in a standard pack, 4 of each of 13 values. In Black Jack, cards have the following values:

- Picture cards (K, Q and J) have a value of 10.
- T (the 10) has a value of 10
- 2 to 9 – face value
- A (ace) has a value of 1 or 11.

In this version of Black Jack, these rules must be followed:

1. For each scenario, a sequence of games will be played, dealing cards in the order they are displayed in the input.

2. To start a game, the players are dealt one card each in the order shown in the input, then the dealer (who is not one of the players) is dealt a card. A second card is then dealt in the same way.

3. Players then play in turn. Their play is based at all times on the value of their hand, being the sum of the values of the cards in the hand. At first each A (ace) counts as 11, but later the value of one or more aces may drop to 1. Once dropped to 1, the ace's value will remain at 1.

4. In a turn
   - If a player was dealt 2 aces (A), one ace's value must be dropped to 1 to give a hand value of 12, not 22.
   - If a player has a hand value of 17 or more they must stick – their turn ends.
   - If a player has a hand value of less than 17, they are dealt a new card. If they now have a hand value of 22 or more, they have "busted" and their turn ends, unless they have at least one A which is still valued at 11. In this case one ace's value is then dropped to 1 and their turn continues.
   - A turn continues until a player sticks or busts.

5. Once all the players have played, if any player has not busted, the dealer plays in a similar way.

6. If the dealer busts, all players who have not busted score 1 win.

7. If the dealer sticks, he scores 1 win for every sticking player whose hand value he has beaten, plus 1 win for every busted player. A player who has not busted scores 1 win if he has a higher hand value than the dealer.

8. The game in progress continues until a card must be dealt but there are none left in the pack. The game in progress stops and is not scored. At this point the scenario ends and the number of wins for each player are displayed.

Input

A number of scenarios are presented, each beginning with 52 card values in a random order. The last line contains just # - this line is not to be processed. The second line of a scenario contains the names of from 1 to 5 players, separated by spaces.

Output

For each scenario, present a single line with the names of the players (in the same order as the input) each followed by their number of wins. Follow this with “dealer” and the dealer's number of wins. Names and wins are separated by single spaces.

Sample Input

```
3A9A285TA36T648K52J45J29677984QT5Q77K964TQJKJK32Q83A
John Sally Peter
4Q52Q372A4T78A7TT53T9A8J6962A587KQK95Q6J96J3K8424J3K
Jenny Siobhan
#
```

Output for Sample Input

```
John 2 Sally 2 Peter 1 dealer 6
Jenny 2 Siobhan 2 dealer 6
```

NZPC
New Zealand
Programming Contest

PROBLEM J **MARKLEFT** **30 POINTS**

There are many markup notations – systems for annotating text with formatting instructions. Markleft is yet another such system. Your task is to implement it.

Markleft takes text input in which punctuation characters are used as follows:

- Text between ^ characters should be converted to upper case
  eg: `One^tWo^threE` will be converted to `OneTWOthreE`
- Text between \ characters should have all double quotes prefixed with \ characters
  eg: `Piglet \said "Hello"\` becomes `Piglet said \"Hello\"`
- Text between # characters will have all embedded numbers (there will never be more than 6 digits in a number) converted to hexadecimal
  eg: `#Text with 31 12 and 64#` becomes `Text with 1F C and 40`
- Text between < characters should be reversed
  eg: `This <be reversed<odd` becomes `This desrever ebodd`
- Text between @ characters should be copied verbatim – ie: markup is ignored.
  Note that this means that there is no way of getting @ characters into the output.
  eg: `This is @\<<#@ verbatim` becomes `This is \<<# verbatim`

Markleft sections may be nested, but if so must be nested properly (like matching brackets).

`^aa<bcd<ee^` becomes `AADCBEE`

The following is illegal because the nesting is improper.

```
One #reverse me ^uppercase me # and more^
```

Markleft sections are closed by the first validly nested markleft character. There is no nesting in this example

`abc^def^ghi^jkl^mno` becomes `abcDEFghiJKLmno`

But nested marklefts of the same kind are possible sometimes. Usually they have little effect, but:

`ab<cd^ef<gh<ij^kl<mn` becomes `ablkJIGHFEdcmn`

The markleft application processes innermost nested sections first

`<#31#<` becomes `F1`

Input

Input consists of a number of lines of "marked left" text to be formatted. Lines of input are no longer than 100 characters. Each line should be formatted independently. You may assume that lines have valid markup. The end of input is signalled by a line with just a # character.

Output

Lines of formatted text.

Sample Input

```
One^tWo^threE
Piglet \said "Hello"\
#Text with 31 12 and 64#
This <be reversed<odd
This is @\<<#@ verbatim
^aa<bcd<ee^
abc^def^ghi^jkl^mno
ab<cd^ef<gh<ij^kl<mn
<#31#<
#
```

Output for Sample Input

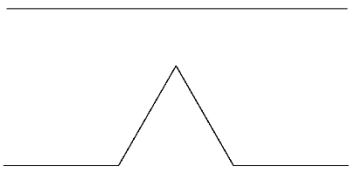
```
OneTWOthreE
Piglet said \"Hello\"
Text with 1F C and 40
This desrever ebodd
This is \<<# verbatim
AADCBEE
abcDEFghiJKLmno
ablkJIGHFEdcmn
F1
```

NZPC
New Zealand
Programming Contest

PROBLEM K **KOCH** **30 POINTS**

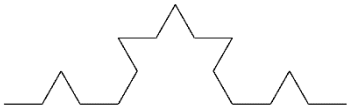
The Koch curve or snowflake is drawn in the XY plane. It is defined recursively.

Start with a line of length 1 from (0,0) to (1,0). We call this a level 0 Koch curve.

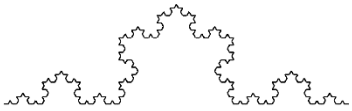


Divide the line into thirds. Make an equilateral triangle on the centre third, then delete the bottom edge of the triangle. This is a level 1 Koch curve.

Divide and put triangles on each of the sections of the level 1 curve. This results in a level 2 curve.



And so on. Here is a level 4 curve.



Consider the strip of the XY plane from X=0 to X=1. The Koch curve divides this strip into an upper and lower part. Your task is: given a point (x, y): $0 < x < 1$ decide whether it is in the upper or lower part of the strip – where upper and lower are defined by a Koch curve of a given level.

Input

You are given a number N ($N > 0$) of cases to check. The first line of input holds the number N. The following N lines each specify a case. Each line holds three values L, X, Y. Where L is an integer: $0 <= L <= 10$. X and Y are floating point numbers: $0 < X < 1$ and $-10 < Y < 10$.

Output

For each input case output one line with either Above or Below. The input has been arranged so that test points are not exactly on the curve. They should be roughly 0.0000001 distant at least from the nearest line segment.

Sample Input

```
8
1 0.2 0.1
1 0.5 0.1
1 0.8 0.1
1 0.5 -0.1
2 0.1 0.05
2 0.166 0.05
2 0.25 0.05
2 0.33 0.1
```

Output for Sample Input

```
Above
Below
Above
Below
Above
Below
Above
Above
```

NZPC
New Zealand
Programming Contest

PROBLEM L

30 POINTS

This is a placeholder – there is no Problem L at the moment.

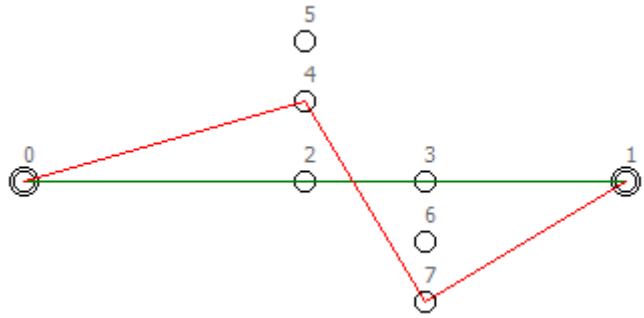
NZPC
New Zealand
Programming Contest

PROBLEM M **FROGS WITH STYLE** **100 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
```

Output for Sample Input

```
3 116.6
```

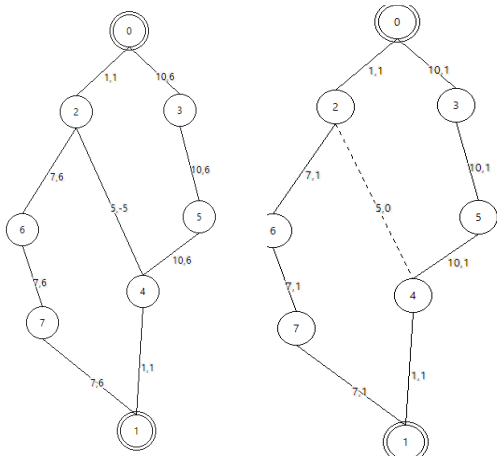
PROBLEM N **SV FILTERS** **100 POINTS**

NZPC Ltd (New Zealand Pump Control) designs and builds water filters. Their production system is fairly simple. A filter contains a sponge like material. We can think of it as having cavities and channels between the cavities through which water can flow. An NZPC customer requires filters which limit the amount of water flowing to be of particular specified values (SV filters). Each filter made by the company does have a particular maximum flow rate, determined by size, number and connection pattern of cavities and channels, however this rate is difficult to control during manufacture. NZPC come up with an ingenious method of making SV filters. Filters are manufactured by the normal method. If their flow rate is too low they are discarded. If the flow rate is too high, particles are added to the inlet of the filter. These particles block the flow through some channels (depending on size and connection pattern). The manufacturing process is to experiment with adding particles of different sizes in different sequences (flushing all particles out and starting again if flow becomes too low). Not all filters will be adjustable to the required flow rate, so some must still be discarded. Filters that are adjusted correctly are heat treated to lock the particles in place, and can be delivered to the customer.

In practice NZPC finds that their experimental process is not completely satisfactory. In particular, flushing particles after failure is difficult and time consuming, and the amount of water used is problematic (it seems that all the fresh water available is needed by Auckland (to cool their housing market (but that is another story))). To streamline their process NZPC installs a CT scanner (seems that a filter is much the same size and texture as a human brain). The scanner gives an accurate map of the cavities and channels in a filter. You are assigned the task of writing software to determine the effect of a particle addition to a filter.

To make the task more manageable you can make the following assumptions. All sizes and flow rates can be represented as integer values. Cavities are always large enough to allow the largest particle to pass through freely. Particles block a channel by getting stuck in the channel. This only happens when the particle is of the correct size for the channel. Large particles cannot pass through small channels, and do not block them. Small particles can pass freely through large channels. Particles whose size matches a channel block that channel, and do not pass through it. The map obtained from the CT scanner takes the form of a graph with cavities as nodes and channels as edges. One node is the inlet and one is the outlet. For each channel there is a single integer 'capacity' value which is the maximum flow rate through that channel. Particles are of integer size also and by careful choice of units their sizes match the channel capacities, in the sense that a particle of size S exactly blocks a channel of capacity S. Note that flow can be in either direction through a channel.

Consider the following example. The numbers on the cavities are for identification. In each graph the inlet is to cavity 0 and the outlet is from cavity 1. The numbers on each the edges show capacity and flow. In the left graph the total flow from inlet to outlet (maximal) is 7. For clarity in the diagram nodes are laid out so that most flows are from high to low. Note the negative flow in the edge between nodes 2 and 4 – in the diagram it is 'uphill'. The right hand diagram shows the system after addition of size 5 particles. The edge between nodes 2 and 4 is now blocked (shown as dashed line) and total flow has dropped to 2.



Input

Input consists of a series of problems, each concerning one filter. The input for a problem starts with a line holding three integers N, E and P. N (3 <= N <= 1000) is the number of cavities and E (3 <= E <= 2000) is the number of channels of a filter. P (1 <= P <= 6) is the size of the particles that will be put into the filter. Following are E lines: one per channel, with three numbers on each: the indices of the two cavities connected by the channel, and the capacity C of the channel. Cavities are indexed from 0 to N-1. Cavity 0 is the inlet and cavity 1 is the outlet. Input is terminate by a line with three zeroes.

Output

For each problem output two numbers (integer values) separated by a space. The first number should be the maximum flow through the unmodified filter. The second should be the flow after particles of the given size have been added. Particle addition can be assumed to be complete – all reachable channels of the correct size will be blocked by the addition of particles.

Sample Input

```
8 9 5
0 2 1
0 3 10
2 4 5
2 6 7
6 7 7
7 1 7
3 5 10
5 4 10
4 1 1
0 0 0
```

Output for Sample Input

```
7 2
```

NZPC
New Zealand Programming Contest

PROBLEM O **DEPACTING** **100 POINTS**

The Pact system enables programmers to encode arbitrary data structures built of records, arrays, strings and integers. An encoded structure takes the form of a string of printable characters. It can be written to a file, or transmitted over a network. Pact is designed to be hardware, operating system and programming language independent. Encoding software has been built. Your task is to build and test a decoder.

Records in Pact are sets of named components. Two records in Pact have the same type if they have components of the same types with the same names. A record must have at least one component. Arrays are one dimensional, indexed from 0, and all elements of an array must be of the same type. Arrays must have at least one element. Two arrays are of the same type if their elements are of the same type. *Note: two arrays with the same element type, but different lengths, are considered to be of the same type.*

In Pact, data is basically encoded as follows (see sample data for examples):

- Integers are encoded as digit strings optionally preceded by minus signs. Integers may have up to 100 digits.

`12`	is encoded as	`12`
`123456543254321`	is encoded as	`123456543254321`
`-23`	is encoded as	`-23`

- Strings may consist only of printable characters (letters, digits, punctuation). They are encoded by enclosure in double quotes. Double quote characters within the string are escaped with a backslash character. Backslash characters within the string are also escaped with a backslash.

`Hello`	is encoded as	`"Hello"`
`Say "Hello"`	is encoded as	`"Say \"Hello\""`
`"\"Hello\""`	is encoded as	`"\"\\\"Hello\\\"\""`

- Arrays are one dimensional and are indexed from zero. They are encoded as comma separated encodings of their elements (in sequence, from element 0 to element length–1), all enclosed by semicolons.

`[5,7,4,3]`	is encoded as	`;5,7,4,3;`
`[red,green,blue]`	is encoded as	`;"red","green","blue";`
`[[1,2,3],[4,5,6],[7,8]]`	is encoded as	`;;1,2,3;,;4,5,6;,;7,8;;`

- Records are tuples with named fields. Field names are alphanumeric; must start with a letter and are case sensitive. Records are encoded as comma separated sequences of fields, enclosed in semicolons. Each field is encoded as the field name followed by a colon followed by the encoding its value. No two fields in a record have the same name. Encodings of records may have the fields in arbitrary order.

`(x:5,y:7,n5:Fred)`	is encoded as	`;x:5,y:7,n5:"Fred";`

This basic encoding is modified by a compression feature in Pact that works as follows.

- When two or more successive array elements have the same value the value is entered once followed by an asterisk and an integer giving the number of repetitions.

`[5,5,5,7,6,6]`	is encoded as	`;5*3,7,6*2;`
`[(x:5,y:6),(x:5,y:6)]`	is encoded as	`;;x:5,y:6;*2;`

- In an array of records, element field entries are omitted for fields whose values are the same as the corresponding values in the predecessor array element. *Note that this only applies to top level record elements of an array. It does not allow selective omission of values in nested records. Note also that this never results in a record encoding with no fields.*

```
[(x:5,y:6),(x:5,y:7)]   is encoded as    ;;x:7,y:6;,;y:7;;
[(x:5,y:6),(x:5,y:6)]   is encoded as    ;;x:5,y:6;*2;
[(a:5,b:(x:6,y:7)),(a:5,b:(x:6,y:8))
    is encoded as   ;;a:5,b:;x:6,y:7;;,;b:;x:6,y:8;;;
```

Input

Input consists of a sequence of problems. Each problem starts with a line with a positive integer N. This is followed by a line with a Pact encoded string of length no greater than 1,000 characters. There are then N lines, each with a test. Tests are data accesses – a 'v' character followed by zero or more indices or field selectors. For example:

```
v          v[0].x          v.x.name          v.x.arr[5].str
```

Index values are always integer literals `i: 0<=i<=100`. Test data accesses always reach strings or integers – they never refer to entire arrays or records. You may assume that test data accesses always correctly match the structure encoded. You may also assume that the total number of string and integer values in a data structure is no greater than 1000. End of input is signalled by an N value of zero.

Output

For each problem, output a line with the text "Problem n" where n is the problem number 1, 2, …
This should be followed by N lines giving the string or integer accessed by the test.

Sample Input

```
1
"Say \"Hello\""
v
3
;"red","green","blue";
v[0]
v[1]
v[2]
4
;;x:5,y:6;*2;
v[0].x
v[0].y
v[1].x
v[1].y
0
```

Output for Sample Input

```
Problem 1
Say "Hello"
Problem 2
red
green
blue
Problem 3
5
6
5
6
```

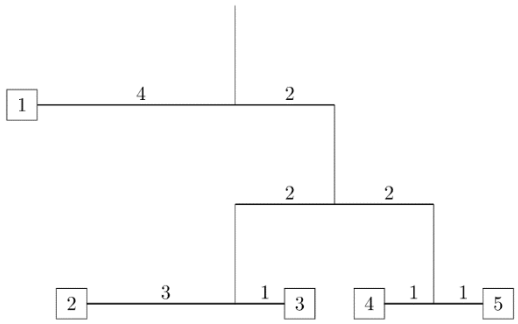
NZPC
New Zealand
Programming Contest

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 ≤ w ≤ 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
```