NEW ZEALAND PROGRAMMING CONTEST 2008

PREAMBLE

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

- Read all input from the keyboard, i.e., use stdin, System.in, std::cin, or equivalent. Input will be redirected from a file to form the input to your submission.

- Write all output to the screen, i.e., use stdout, System.out, std::cout, 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, ncurses 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 an otherwise 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 white space, i.e., spaces or tabs.

- An uppercase letter is a character in the sequence 'A' to 'Z'. A lowercase letter is a character in the sequence 'a' to 'z'. A letter is either a lowercase letter or an uppercase letter.

- Unless otherwise stated, a word or a name is a continuous sequence of letters, and 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 (or characters) specifying the end of line.

- All input files are terminated by a 'sentinel' line, followed by an end of file marker. This line should not be processed.

Please also note that the filenames of your submitted programs may need to follow a particular naming convention, as specified by the contest organisers at your site.

The following messages will be issued as a result of an incorrect submission. The feedback will be provided only on the first of the errors encountered by the judges.

1. Compilation Error

Any error that prevents the program from compiling and linking successfully.

2. Run-Time Error

Issued as a result of any error which occurs at execution time.

3. Time Limit Exceeded

Programs exceeding two minutes of execution time will be issued this message. It will also be issued when a program enters an "obvious" infinite loop.

4. Wrong Answer

Issued when some or all of the output is incorrect, excluding errors in formatting (see below).

5. Output Format Error

This result is issued when all of the cases in the team's submission are basically correct (all of the technical parts of the problem work correctly) but the output is not in the format requested in the problem statement. If one or more of the test cases fail, the message should be Wrong Answer.

White space at the end of a line, or at the end of the output, is ignored.

6. Other -Contact Staff

This will be used in exceptional circumstances only. It allows for human intervention in a mechanical system!

PROBLEM A — TRIANGLES — 3 POINTS

It is not hard to draw a triangle of stars of any given size. For example, a size 5 triangle would look like this (5 stars high and 5 stars wide):

```
*
**
***
****
*****
```

Your task is to draw triangles in a number of sizes.

Each line of input contains a single positive integer, n, 1 <= n <= 100. The last line of input contains 0. For each non-zero number, draw a triangle of that size.

Output consists of triangles of the appropriate sizes. Each triangle is wider at the bottom. There are no gaps between the triangles.

Sample input

```
5
3
2
7
0
```

Sample output

```
*
**
***
****
*****
*
**
***
*
**
*
**
***
****
*****
******
*******
```

PROBLEM B — MP3 SONGS — 3 POINTS

Sue loves her mp3 player but she hates the fact that her shuffle mode plays her tracks randomly. Because she loves order and patterns she would like the tunes on her mp3 player to be played in alphabetical order. In this problem you have to help Sue by sorting her tunes into alphabetical order of tune name.

Input consists of a number of scenarios, each beginning with a single positive integer, n, the number of music tracks that require sorting ($1 < n <= 200$). The last line of input is a single 0 – this scenario should not be processed.

Each scenario consists of n lines, each line containing a tune name. No line will contain more than 250 characters. Each name will begin with an alphabetic character.

Output will consist of the scenario number, the first being 1, on a line on its own. This will be followed by n lines showing the tune names from the input list, sorted in alphabetical order, one name per line. Case should be ignored.

Sample Input

```
10
Forever
Take A Bow
Always On My Mind
Lollipop
Love In This Club
No Air
Sweet About Me
Party People
Mercy
American Boy
8
Partita
Air on a 'G' string
Sinfonia in D
Jesu, joy of man's desiring
Arioso
Violin Concerto in A Minor
Brandenburg Concerto 2
Concerto for 2 violins
0
```

Sample Output

```
1
Always On My Mind
American Boy
Forever
Lollipop
Love In This Club
Mercy
No Air
Party People
Sweet About Me
Take A Bow
2
Air on a 'G' string
Arioso
Brandenburg Concerto 2
Concerto for 2 violins
Jesu, joy of man's desiring
Partita
Sinfonia in D
Violin Concerto in A Minor
```

PROBLEM C TENNIS 3 POINTS

Do you follow tennis? If so, you will be aware of the strange scoring system, 15, 30, 40, game. Then, of course there is 40-40 or deuce, with one player having to get 2 points ahead to win a game.

This problem asks you to work out the score of a tennis match given the player who wins each point. To keep things simple, players are called A and B. The points are given as a sequence of letters, where A means a point won by player A, B a point won by player B. Here is an example:

```
ABAABBAABABAABABBB
```

Remember, to win a game a player needs to score 4 points, but also to be 2 points ahead. So in the above example, the first game is represented by `ABAABBAA` and player A has won it. It went to deuce then player A won two points. The second game was represented by `BABAABABBB`. This game was won by player B. Again it went to deuce, player A went ahead by 1 but player B caught up. Player B then won the last 2 points. After all that, the score would be 1 game all.

Input will consist of a number of strings of characters, each on a line of its own. No line will contain more than 250 characters. The last line will contain just the # character – this line should not be processed.

Each line represents part of a tennis match, a number of completed games. The line will contain only the upper case letters A and B representing points won by each player. Consider each line as separate, and work out the score, in games, after the points shown.

Output will be one line for each input line in the form of a score, `A x B y` where x is the number of games won by A, y is the number of games won by B.

Sample Input	Sample Output
`ABAABBAABABAABABBB`	`A 1 B 1`
`AABAABABAAAABBAAABABBAB`	`A 3 B 1`

```
#
```

PROBLEM D — BOOKINGS — 3 POINTS

Air NZ (not to be confused with Air New Zealand) operate flights between various smaller cities in New Zealand. They have a number of Aerospatiale ATR72 aircraft, each with 68 seats. Their policy is that passengers must book for a particular flight, and if they fail to show up then they lose their fare. That means that, unlike other airlines, they do not have to overbook flights, so a maximum of 68 seat bookings may be made for any flight.

In this problem you will be given one or more scenarios. Each scenario begins with a flight number and a current number of booked seats, n, (0 <= n <=68). The flight number will be the letters AZ followed by 3 digits, which will be followed by a space.

A number of transactions will follow, each on a single line. If the transaction is to book one or more seats, this will be presented as an upper case B followed by a space followed by an integer, n, which is the number of seats to book (0 < n <=68). Should the number of bookings requested take the number of bookings for the flight to more than 68, then that transaction must be completely ignored.

Customers may also cancel a booking - this will be presented as an upper case C followed by a space followed by an integer, n, which is the number of seats to cancel (0 < n <=68). Should n be greater than the number of seats currently booked, then that transaction must be completely ignored.

The list of transactions will be terminated by a line containing `X 0` (X and zero separated by a space) which should not be processed. A line containing `# 0` (# and zero separated by a space) marks the end of input.

Output consists of one line for each flight. The line contains the flight number, followed by a space, followed by the number of seats booked for that flight after the transactions have been carried out.

Sample input and output is on the next page.

Sample Input

```
AZ001 5
B 10
B 1
B 2
C 1
B 3
C 1
C 4
B 2
B 1
B 1
B 5
C 3
X 0
AZ002 60
C 61
B 9
X 0
# 0
```

Sample Output

```
AZ001 21
AZ002 60
```

PROBLEM E — LIBRARY CODES — 10 POINTS

A local library is redoing its book labels with the Dewey Decimal classification. Where possible, labels are printed horizontally, but if they are too wide, they are printed vertically. A fixed pitch font is used, so all characters are the same width. To make the labels readable, a 1mm gap must be left before and after the number.

Given the width of a book in mm, the width of a character of the font and the text of the label, it is easy to decide if the label could be displayed horizontally or if it would have to be displayed vertically.

Input consists of data about a number of libraries. The final line contains # 0 (separated by a space) – this line should not be processed.

For a library, the first line of input will be the name of the library (up to 20 letters only, no spaces), followed by a single positive integer, f, which is the width in mm of a single character in the font used by the library. $1 < f < 8$. The next line contains a single integer, c, the number of books to be processed. There then follows c lines, each containing data about 1 book. Each line contains a positive integer, w, the width of the book in mm ($5 <= w <= 50$), and the text to be displayed on the label. The text will contain between 3 and 6 characters.

Output for each library begins with the name of the library on a single line, followed by 1 line for every book listed in the input. Each book will begin with "`Book n`", where n is the number of the book in the library's list, the first book being 1, followed with either the word `horizontal`, or the word `vertical` according to how the book is to be displayed.

Sample input and output is on the next page.

Sample input	Sample output
Auckland 3 6 11 811 25 636.7 38 330.94 29 398.2 16 398.6 18 973.05 Wellington 4 3 35 636 12 398.2 9 398.2 # 0	Auckland Library Book 1 horizontal Book 2 horizontal Book 3 horizontal Book 4 horizontal Book 5 vertical Book 6 vertical Wellington Library Book 1 horizontal Book 2 vertical Book 3 vertical

PROBLEM F LETTERS 10 POINTS

"The quick brown fox jumped over the lazy dogs."

This sentence is well known for containing at least one of every letter of the alphabet. We could say, then, that it contains 26 different letters.

In this problem you are given a series of pieces of text. In each case, state how many different letters the text contains. Spaces, digits and punctuation are ignored. Upper case letters are not distinguished from their lower case counterparts, so 'A' and 'a' just count as one letter.

Input consists of a series of lines of text, each one no more than 250 characters long. Each piece of text will contain at least one non-white space character, but not necessarily a letter. The last line will contain just `#` - this line should not be processed.

Output consists of a single integer for each line of input, each integer on a line by itself. The integer will be the number of different letters found in the text.

Sample input

```
The quick brown fox jumped over the lazy dogs.
2 + 2 = 4
New Zealand Programming Contest.
#
```

Sample output

```
26
0
16
```

PROBLEM G — OUTFITS — 10 POINTS

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 leftmost outfit, whereas Cas numbers them from right to left, so 1 is the rightmost outfit. Their wardrobes look the same, though, so, for example, the leftmost 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 consists of a number of scenarios. Each scenario 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 last line of input is a scenario containing 0 0 – this is not to be processed.

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 will be a scenario number, followed by one line for each day in that scenario. 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.

Sample input and output is on the next page.

Sample input

```
20 4
0
0
1 3
6 15
20 1
10 10
12 3
7
0
7 5
3 10
7 6
0 0
```

Sample output

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

PROBLEM H EXERCISE 10 POINTS

Our local gym has a number of exercise machines, each of which has a number of "levels". Some levels are harder than others, and exercising at those levels uses more energy per minute than at the easier levels. In this problem you have to work out how much energy a person would use up during an exercise session.

Input will consists of a number of scenarios, each representing a machine and a number of people using the machine for exercise. Each scenario will begin with an integer, `n`, the number of levels available on the machine (`0 < n < 10`). The next `n` lines define the energy expended per minute at one level of the machine. The first line represents level 1, the second level 2 and so on.

There then follows data about a number of people who are exercising on the machine described. The first line of data gives the name of the person who is exercising (a single series of between 2 and 10 letters, lower case except for the first) followed by a space, followed by `e`, the number of exercises the person carried out (`0 < e <= 50`).

The next `e` lines contain the exercises carried out by the person. These consist of an integer giving the level, followed by a space, followed by d, the duration of the exercise in minutes.

The list of people is terminated by a line containing `# 0`. This line should not be processed.

Input is terminated by a scenario where n is 0 – this scenario should not be processed.

Output is in sections, one section for each machine. Machines must be numbered, starting at 1. For each machine, one line is output for each person using the machine. This output is the name of the person, followed by a space, followed by the total amount of energy they expended carrying out their set of exercises. The people should be output in the order they appear in the input.

Sample input and output is on the next page.

Sample Input

```
3
10
20
30
Bill 3
1 5
2 20
1 5
Jason 2
1 10
3 30
# 0
2
50
25
Susan 4
1 5
2 20
1 5
2 15
Li 2
2 20
1 60
# 0
0
```

Sample Output

```
Machine 1
Bill 500
Jason 1000
Machine 2
Susan 1375
Li 3500
```

PROBLEM I — QUEEN KINGDOM — 30 POINTS

A game designer wants to build a modified chess board of size n by n. At certain squares there will be placed pillars that prevent placement of pieces and also prevent attacks across them.

Before shipping the specifications to a manufacturer, the designer wants to know what is the maximum number of non-attacking queens that can be placed on each of his designs. Recall that queens can attack vertically, horizontally and diagonally as far as possible until the end of board (or, in our case, until a pillar is reached). Also of interest is how many different ways this maximum number can be achieved.

Input

Input consists of scenarios. Each scenario starts with an integer n (n <= 10). Input ends when n = 0. The next n lines, each containing n characters represent the rows of the chess board. Here, a '0' represents an open square and a '1' represents a pillar.

Output

For each case output the maximum number queens that can be placed, followed by the number of different ways possible (separated by a single space).

Sample Input

```
3
010
111
000
6
000000
000000
000000
000000
000000
000000
6
000010
001110
111011
010110
011010
010010
0
```

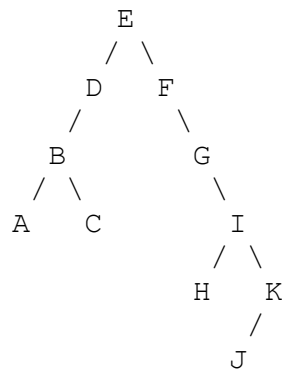
Sample Output

```
3 3
6 4
7 270
```

PROBLEM J RECONSTRUCTING TREES 30 POINTS

It is very easy to produce the various traversals of a binary tree, however this problem requires you to produce the tree from the traversals. Specifically, given the pre-order and in-order traversals of a binary tree (not necessarily a binary search tree), reconstruct, if possible, the original tree.

For instance, the following tree will produce the traversals shown



```
Preorder : EDBACFGIHKJ
Inorder  : ABCDEFGHIJK

Postorder: ACBDHJKIGFE
```

Input will consist of representations of several trees, each on a single line, and terminated by a `#`. Each line will consist of two strings of up to 26 unique uppercase letters in the form

<pre-order> <space> <in-order>.

You can assume that the two strings will be permutations of each other.

Output will consist of the post-order traversal of the reconstructed tree if possible, otherwise the words 'Invalid tree'.

Sample Input

```
EDBACFGIHKJ ABCDEFGHIJK
#
```

Sample output

```
ACBDHJKIGFE
```

PROBLEM K — TRAFFIC ENGINEERING — 30 POINTS

ISPs live on very thin margins, so saving money in selecting the optimum route for network traffic is important for the survival of the company. Your job is to find the cheapest route between two hosts on the Internet for the person sending the data. Network traffic that passes over the providers own network is essentially free and therefore costs $0 to send traffic through. Network traffic that passes over other peoples network costs $1 per node traversed.

Input is a set of scenarios. Each scenario has three sections. Each section begins with a number on one line followed by that many lines of text.

The first section lists unidirectional links between networks. Two networks are specified per line separated by a space. A line that contains "Alice Bob" states that there is a link from Alice to Bob, but not the other way around.

The second section lists networks that you own. One network is specified per line.

The third section lists the packet source and destinations you must provide an answer for. Two networks are specified per line separated by a space. These specify the source and destination of a packet.

A network of 0 nodes will terminate the input and should not be processed. A network will have no more than 100 nodes.

Output will consist of a number representing the lowest possible cost of sending that packet. Nodes that are owned by the sender cost zero, nodes that aren't owned cost 1. The source and destination nodes count towards the cost.

Sample Input

```
5
Alice Bob
Bob Charlie
Alice Charlie
Charlie Dave
Dave Alice
2
Alice
Bob
2
Alice Bob
Bob Alice
0
```

Sample Output

```
0
2
```

PROBLEM L THE RACE 100 POINTS

During the Annual Interstellar Competition for Tuned Spaceships, N spaceships will be competing. Each spaceship i is tuned in such a way that it can accelerate in zero time to its maximum speed Vi and remain cruising at that speed. Due to past achievements, each spaceship starts at a starting position Xi , specifying how many kilometers the spaceship is away from the starting line.

The race course is infinitely long. Because of the high speeds of the spaceships, the race course goes straight all the time. On that straight course, spaceships can pass one another very easily, without interfering with each other.

Many people in the audience have not realized yet that the outcome of the race can be determined in advance. It is your task to show this to them, by telling them how many times spaceships will pass one another, and by predicting the first 10 000 times that spaceships pass in chronological order.

You may assume that each spaceship starts at a different position. Furthermore, there will never be more than two spaceships at the same position of the course at any time.

Input

The first line of the input specifies the number of spaceships n ($0 < n \leq 250\ 000$) that are competing. Each of the next n lines describes the properties of one spaceship. The i + 1th line describes the ith ship with two integers Xi and Vi , representing the starting position and the velocity of the ith spaceship ($0 \leq Xi \leq 1\ 000\ 000$, $0 < Vi < 100$). The spaceships are ordered according to the starting position, i.e. $X1 < X2 < \ldots < XN$. The starting position is the number of kilometers past the starting line where the spaceship starts, and the velocity is given in kilometers per second. Input is terminated with an input of 0 spaceships, no output should be generated for this race.

Output

The first line of output should contain the number of times that spaceships pass one another during the race modulo 1,000,000. By publishing the number of passes only modulo 1,000,000, you can at the same time prove your knowledge of it and don't spoil the party for the less intelligent people in the audience.

Each of the subsequent lines should represent one passing, in chronological order. If there would be more than 10,000 passings, only output the first 10,000 passings. If there are less than 10,000 passings, output all passings. Each line should consist of two integers i and j, specifying that spaceship i passes spaceship j. If multiple passings occur at the same time, they have to be sorted by their position on the course. This means that passings taking place closer to the starting line must be listed first. The time of a passing is the time when the two spaceships are at the same position.

Sample input and output is on the next page.

Sample Input

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

Sample Output

```
2
3 4
1 2
```

PROBLEM M LIST CALCULATOR 100 POINTS

Your job, should you choose to accept it, is to write an interpreter for a small list calculator. Input will consist of various expressions, which you evaluate.

Operators (from most tightly bound to least tightly bound). Brackets (), Array Slicing, Unary operators, Binary operators, List concatenation.

The Unary operators (`*`, `/`, `+`, `-`) continuously remove the first two items of the list, and replace them with the result of the operator being applied. For example `+(1:2:4)` takes 1 and 2 off the front of the list, applies "+" to them, to get 3 which is placed on the front of the list leaving `+(3:4)`, again we take the first two items, remove them, apply +, and insert the result into the front of the list `+(7)`. Now the list has fewer than two items so we return the list 7. `*`, `/`, `+`, `-` are multiplication, division, addition and subtraction respectively. There is no unary minus on numbers.

Array slicing is done with the [begin:end] operator applied as a postfix to an expression. Items are 0 indexed and include the beginning item but don't include the end item. Thus `(1:2:3:4:5)[1:3]` is `2:3`. The begin or end may be omitted and is expected to mean all items from the beginning or all items until the end (respectively).

Binary operators are applied to the first element of both lists, then the second element and so on. If one list is shorter than the other, then the shorter list is padded with its last item until both lists are the same length. Thus `(1:2:3)+(4:5)` is `(5:7:8)`. Note that applying a binary operator to an empty list produces an empty list.

List concatenation is provided with the : operator. Standalone numbers are assumed to be a list of a single item. All numbers are assumed to be integers. None of the expressions will cause a division by zero.

The `print` command will output the result of an expression on a line by itself to standard out. Lists have items separated with a : with no spaces. The maximum number of items to be output will be no more than 15.

Assignment is provided with the `=` operator. Variables are always single letters, case sensitive, and are never redefined.

Input will be terminated with a line starting with a "`#`".

Sample input and output is on the next page.

Sample Input

```
print +(1:2:4)
print (1:2:3:4:5)[1:3]
print (1:2:3)+(4:5)
N=1:(N+1)
E=2*N
print E[:5]
print +E[:10]
F=1:1:(F+F[1:])
print F[:10]
#
```

Sample Output

```
7
2:3
5:7:8
2:4:6:8:10
110
1:1:2:3:5:8:13:21:34:55
```

PROBLEM N — BARGAIN OR NO BARGAIN — 100 POINTS

The NZPC (inc) Entertainment Division has planned a new TV gameshow called "Bargain or no Bargain". In this gameshow, there are a number of briefcases which each contain a cheque for a set amount of prize money. The values of the prizes (one for each briefcase) are given in advance, but it is unknown which briefcase contains which prize. The lone contestant is given one case, without knowing its contents.

The game consists of a series of rounds of play. In each round (including the first), the contestant is first given an offer of a certain amount of money by the NZPC Bank (the precise amount is specified below), which the contestant may choose to accept and thereby quit the game, forfeiting the contents of their own briefcase. Otherwise, the contestant then selects one of the remaining closed cases to open, revealing its contents and thus eliminating the possibility that the revealed prize is in the contestant's case. Play continues until the contestant either accepts a bank offer, or all cases except the contestant's are opened. In this event, the prize in the contestant's case is revealed and the contestant goes home with the cheque from his or her own case.

However, NZPC is unsure of how much money the prizes should be worth in order to make the game exciting without breaking the bank. They want a program that will determine, given a set of prize values, whether a contestant would win more than a certain amount of money on average, assuming that the contestant plays optimally. Fortunately, the NZPC has found a cheaper option than outsourcing - somehow they have convinced New Zealand's most talented programmers to write the software by paying them with only pizza and coke!

Game Details

- Let `S` be the sum of the remaining prizes, and `N` the number of remaining prizes. The bank offer is always $S / (N^2)$.
- Each case is equally likely to contain each prize.
- The contestant is assumed to play optimally in order to maximize the expected value of u(x), where u(x) is the utility of winning x dollars, and is equal to ln(x), the natural logarithm of x. This utility function is designed to model the fact that a fixed increase in wealth becomes less useful when one has more money to start with.

Input

The input is a series of game scenarios, each of which is on two lines. The first line of each scenario contains the dollar values of the set of prizes, each separated by a space. The prize values are all positive integers. The second line contains a positive integer M, the maximum dollar value of expected prize money per game (under optimal play, as defined above) which the NZPC can afford. Each game scenario will contain a maximum of 25 prizes. Prize values may be duplicated.

Problem description continues on the next page.

Output

For each game scenario, if the expected prize money per game under optimal play is greater than M, output the string "`UNACCEPTABLE`" on a line by itself, otherwise output the string "`OK`" on a line by itself.

Input is terminated by a line containing only "-1", which should not be processed.

Sample Input

```
1 10 100 1000
200
1 3000 50 750 10000
1500
-1
```

Sample Output

```
OK
UNACCEPTABLE
```