A. Campus Speeding

Author: Sean Falconer, March 12, 2004

UNB Security has noticed a problem with campus drivers speeding. In an effort to fine more of these violators of the campus speeding regulations, security has setup radar detectors to collect data about the speeds of vehicles around campus. You've been hired to analyze this data.

Input

On the first line of input there will be one integer C which represents how many cases there will be. Each test case begins with one integer, $R \leq 20$, which represents how many radar detection stations have been setup. The next line of input will consist of R positive integers, each representing the speed limit for the section of campus that this radar station is located in. The next line of input will contain an integer, $V \leq 100$, which represents how many vehicles we have data for. Following this, there will be V lines, each with R positive integers. Each line represents a separate vehicle and each positive integer is the speed recorded for that vehicle by the given radar station.

Output

For each test case, output "Case T:", were T is replaced by the current test case number, this should be on its own line of output. Then, for each vehicle in this test case, you must output "Vehicle X: ", where X is replaced by a number representing which vehicle this line represents. After this, you must output either "is a speeder", if this vehicle broke the speed limit, or "is not a speeder", if this vehicle did not break the speed limit. Check sample output for examples of how output should be formatted.

Sample Input

```
2
3
15 16 17
2
13 13 14
22 16 23
4
25 30 25 40
3
30 30 25 40
25 30 25 40
20 20 20 20
```

Sample Output

```
Case 1:
Vehicle 1: is not a speeder
Vehicle 2: is a speeder
Case 2:
Vehicle 1: is a speeder
Vehicle 2: is not a speeder
Vehicle 3: is not a speeder
```

B. So Who Won?

Author: Nathan Scott, March 12, 2004

Yeah, we were going to hold this UNB Programming Competition, but we need some way to determine who wins. Of course, we COULD just keep track of these things ourselves, but we're programmers so it goes without saying that we are too lazy to do that. We'll just write a program to do it instead. However, being the executive of the UNB Programming Club means we are among the laziest of the lazy! So rather than even bothering to write a program ourselves to do it, we're going to get the competitors themselves to write one.

(Let's hope one of you solves this one: We'd hate to have to like, figure this stuff out ourselves!)

The winner of the competition is the competitor who solves the most problems. If there is a tie, the competitor who solved these problems in the least amount of time wins. The competition may have a penalty value for submissions. (In case you are wondering, the competition you are in right now in real life doesn't!) The total time taken to solve a problem is equal to the time until the problem was accepted, plus the penalty value for each submission except for the accepted one. For example, with a penalty value of 20 minutes, if you solve a problem after 45 minutes, and it took you 3 submissions, your total time for that problem would be 85 minutes (45 + 2 * 20 for the 2 prior submissions).

Input

There may be multiple test cases. Each test case begins with a number, N, indicating the number of competitors in the competition ($0 < N \leq 20$), followed by a number P ($P \geq 0$) that specifies the penalty for wrong submissions ($P = 0$ means there is no penalty.)

The following N lines contain the results of a single competitor in the competition, in the following format:

name t s t s ... -1

"name" is the user name of a competitor, and will consist of only upper or lower case letters (no numbers or symbols), and will be ≤ 10 characters long. "t" and "s" will always come in pairs. "t" [$t > 0$] is the time taken to solve the problem specified by the pair, and "s" [$s > 0$] is the number of submissions it took to solve it. -1 signifies that its the end of this competitor's entry. The input only lists the submission details for accepted solutions.

Output

For each test case, print out the winner's name followed by a single space and then their total time. Each test case should have its own single line of output.

Sample Input

```
4 25
Nathan 30 1 45 2 95 2 -1
Sean 25 9 -1
Oleg 10 1 15 1 25 2 -1
Tom 50 2 90 1 -1
```

Sample Output

```
Oleg 75
```

C. Data Classification

Author: Sean Falconer, March 12, 2004

The area of data mining is concerned with extracting "knowledge" from a given domain or corpus of data. One approach to knowledge extraction is data classification, which attempts to determine which class or category a given piece of data belongs to. One of the simplest approaches is to use a **Naive Bayes Classifier**, which is based on Bayes Theorem from statistics,

$$P(B|A) = \frac{P(B)P(A|B)}{P(A)}. \tag{1}$$

The reason this approach is called "naive" is because it makes the assumption that all the attributes are independent, however, most of the time this is not the case. Although this assumption is obviously wrong, the Naive Bayes Classifier performs incredibly well.

With the simplest implementation of a Naive Bayes Classifier, the model determines, given an example represented as a set of attributes, the probabilities that for each class, C_i, this example belongs to that class. This is done with the following formula,

$$f_i(E) = P(C_i)\prod_{j=1}^{a} P(A_j = v_{jk}|C_i), \tag{2}$$

where v_{jk} is the value of attribute A_j in the example and $\prod_{j=1}^{a}$ means the product of $P(A_j)$ from $P(A_1) \times P(A_2) \ldots \times P(A_a)$. The class that yields the largest probability value $f_i(E)$ for example E is the class the model determines this example belongs to. The original probabilities for $P(C_i)$ and $P(A_j = v_{jk}|C_i)$ are pre-computed using a training set.

For this problem, you will be given a list of class probabilities and attribute value probabilities which correspond to the values for the given example with respect to the given class. You must determine which class the example belongs to using the following formula, which is a modification of equation (2),

$$h = \max_i^{|C|}\{P(C_i)\prod_{j=1}^{|A|} P(A_j = v_{jk}|C_i)\}, \tag{3}$$

where h is the correct hypothesis, ie. the class that corresponds to the set of probabilities that maximizes the overall probability calculation.

Consider the following set of numbers:

```
0.4 0.5 0.1 0.89 0.2
0.6 0.2 0.8 0.11 0.3
```

The first line represents the probabilities for class 1 and the second line is the probabilities for class 2. The first number, 0.4, represents the class probability, and the following numbers correspond to the attribute value probabilities with respect to this class. To calculate the probability for this example for class 1 we compute $P(C_1) \times \prod_{i=0}^{|A|} P(A_i) = 0.00356$. For class 2, we compute $P(C_2) \times \prod_{i=0}^{|A|} P(A_i) = 0.00317$. Now we must choose the maximum overall probability, which is obviously 0.00356, therefore, the hypothesis will be that this example belongs to class 1 rather than class 2.

Input

On the first line of input there will be an integer N. This number represents how many examples are in the input. Each example begins with integers $C \geq 1$ and $A \geq 1$. The C represents how many classes there are and A represents how many attribute values there are for each class. Next, there will be C lines, each with $A + 1$ floating point numbers. The first number of each line represents the class probability and the next A numbers are the attribute value probabilities.

Output

The output should consist of N lines, one line for each example. For each example, you must output "Class X", where X is replaced by the the class number with the largest overall probability followed by a space and then the probability you calculated. The probability value should be a floating point number, consisting of 5 digits beyond the decimal point, rounded to the nearest decimal.

Sample Input

```
3
2 4
0.4 0.5 0.1 0.89 0.2
0.6 0.2 0.8 0.11 0.3
1 8
0.1 0.1 0.9 0.9 0.2 0.7 0.34 0.44 0.111
5 1
0.2 0.45
0.3 0.55
0.15 0.67
0.25 0.01
0.05 0.1
```

Sample Output

```
Class 1 0.00356
Class 1 0.00002
Class 2 0.16500
```

D. Balloon Collecting

Author: Sean Falconer, March 12, 2004

The UNB Engineering Society has recently held a contest to develop the world's best balloon floatation compound. The team that won the competition has managed to develop a new gas compound that is able to float a conventional balloon well beyond that of helium gas. The team called their invention the "Super Hyper-Mega Global Anti-gravitational Liquid Thermamine Compound", however, they use the term SHMGALTC for short. Due to the excitement of winning the competition, the winning team released a large number of balloons into the air, however, these balloons are beginning to interfere with the normal operations of the airport.

UNB has been ordered to collect as many of the balloons as possible. They need your help. UNB only has one large balloon capable of carrying human passengers. Also, they only have enough gas to lift the balloon to any given height once, and then begin their descent to the ground. Given the list of balloon heights, they want you to determine what is the maximum number of balloons that they can catch. The balloons can only be captured during the descent of the human passenger balloon. That is, UNB's collecting balloon must lift to a certain height and then begin its descent and the balloons must be captured in order, so a balloon in the sequence can be skipped but once skipped, it cannot be collected later.

Input

The input contains one or more test cases, each test case will consist of one or more non-negative integers representing the list of incoming balloons. These balloons must be processed in the order that they are given in, that is, you cannot rearrange the position of a balloon in the sequence, but it is acceptable to miss or pass a balloon. The end of a test case is terminated by a -1 (this does not represent one of the balloons), and the end of all the test cases is terminated by the end of file.

Output

The output should consists of a single line stating "The maximum number of balloons possible to catch is: X", where X is replaced by the maximum number of balloons that you should be able to collect. Each test case should be on its own line.

Sample Input

```
389 400 100 56 78 34 1 100 300 -1
1 2 3 3 2 1 -1
```

Sample Output

```
The maximum number of balloons possible to catch is: 5
The maximum number of balloons possible to catch is: 3
```

NOTE: There are no constraints on the number of balloons in the input, so your solution must be efficient enough to handle a large number of balloons.

E. Selbat H. Turt

Author: Nathan Scott, March 12, 2004

Professor Selbat H. Turt is researching in the area of reversible logic design. He wants to design circuits that perform different functions on a binary input. He is concerned mainly with reversible circuits that are made up of a cascade of gates known as “Toffoli” and “Fredkin” gates. Such circuits are guaranteed to be reversible, you can trust Prof. Turt on this one, so don’t worry about it.

A Toffoli gate is a type of “controlled NOT” gate, and looks like these:

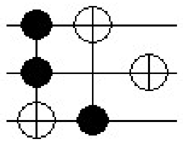

Figure 1: Toffoli Gate.

The dark circles are the “control bits” and the hollow circle is the “target bit”, of which there will always be exactly 1. Given an input value of either 0 or 1 on every line, the Toffoli gate will invert the target bit’s value if and only if all of the control bits are set to 1. If there are no control bits, the target bit will always be inverted, so in that case the Toffoli gate acts as an ordinary NOT gate.

A Fredkin gate is a type of “controlled SWAP” gate, and looks like these:

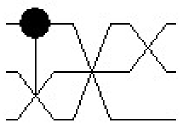

Figure 2: Fredkin Gate.

Again, the dark circles are the “control bits”. The two criss-crossing lines are the “target bits”, of which there will always be exactly 2. Given an input value of either 0 or 1 on every line, the Fredkin gate will swap the two target bits’ values with eachother if and only if all of the control bits are set to 1. If there are no control bits, the two target bits will always be swapped.

A circuit is made up of a series of these two gates, each of which is applied one after the other. The order most definitely matters in this, since the output after one gate has been applied becomes the input for the next gate. Here is an example of a circuit with 4 bits, along with one possible input (on the left) and the corresponding output (on the right) with the values of each bit at each intermediate step in between.

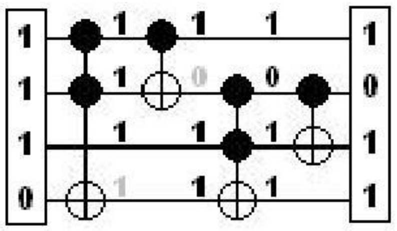

Figure 3: Example Circuit.

Write a program to save Prof. Turt some time by reading in a description of a reversible circuit, and then generating the truth table for it. In other words, for every possible input to the circuit, calculate the corresponding output for the circuit.

Input

The first line of the input contains a single number, N, indicating the number of test cases to follow $(0 < N \leq 50)$.

Each test case begins with two numbers, S and G, separated by a single space. S is the "size" of the circuit, the number of bits in total for this circuit $(0 < S \leq 10)$. G is the "gate count", the number of gates in this circuit $(0 \leq G \leq 20)$.

Following these two numbers will be G gate descriptions. These are the gates in the circuit, in the order they should be applied. In these descriptions, bits will be described by lower case letters from 'a' to 'j' inclusive. At all times, 'a' will mean the least significant bit (the rightmost column in the truth table, see sample output), 'b' the next least significant, and so on. This means that if $S = 4$, the bits that you can expect to appear are 'a', 'b', 'c', and 'd', no others.

Toffoli gates are specified as follows: $T(c1c2 \ldots cn, t1)$ where $c1, \ldots, cn$ are the control bits, and $t1$ is the target bit. $\{c1, \ldots, cn\} \bigcap \{t1\} = \emptyset$ (no bit is both a control and target) and $c1 \neq \ldots \neq cn$ (no control bit is ever listed twice). There will always be a target bit, and it will always be the bit immediately after the ',' character, but there may or may not be any number of control bits. (Thus $T(, a)$ is a valid gate.)

Fredkin gates are specified as follows: $F(c1c2 \ldots cn, t1t2)$ where $c1, \ldots, cn$ are the control bits, and $t1$ and $t2$ are the target bits. $\{c1, \ldots, cn\} \bigcap \{t1, t2\} = \emptyset$ (no bit is both a control and target) and $c1 \neq \ldots \neq cn$ (no control bit is ever listed twice). There will always be two target bits, and they will always be the two bits immediately following the ',' character, but there may or may not be any number of control bits. (Thus $F(, ab)$ is a valid gate).

Output

The first line of the output should say "Circuit #n:" where n is the current test case number (beginning with 1).

The next line should contain the column headers, which are simply the variable names (the characters used to specify each bit) repeated twice (once for the input table, once for the output table). On the line beneath them, '-' characters should be printed repeatedly to "underline" them. See the sample output.

The rest of the output consists of the truth table for the circuit. Each line should first print the input values of each bit (1 or 0), under the "input" table, underneath the corresponding column. The output values (again, 1 or 0) of the circuit should then similarly be printed under the "output" table. The entries should be listed in ascending order of input values, treating each input value as a binary number (0000 = 0, 0001 = 1, 0010 = 2, etc.).

At all times the "input" table should be separated from the "output" table by precisely 4 blank space characters. See the sample output.

Each test case should be separated by a blank line.

Sample Input

```
3
3 2
T(ab,c)T(,b)
4 4
T(d,c)F(ab,cd)F(,ac)T(,a)
4 4
T(ab,d)T(a,b)T(bc,d)T(b,c)
```

Sample Output

```
Circuit #1:
cba    cba
---    ---
000    010
001    011
010    000
011    101
100    110
101    111
110    100
111    001

Circuit #2:
dcba    dcba
----    ----
0000    0001
0001    0101
0010    0011
0011    0111
0100    0000
0101    0100
0110    0010
0111    1111
1000    1000
1001    1100
1010    1010
1011    1110
1100    1001
1101    1101
1110    1011
1111    0110

Circuit #3:
dcba    dcba
----    ----
0000    0000
0001    0111
0010    0110
0011    1001
0100    0100
0101    1011
0110    1010
0111    1101
1000    1000
1001    1111
1010    1110
1011    0001
1100    1100
1101    0011
1110    0010
1111    0101
```

NOTE: Output text looks different in order to ensure spacing matches the specifications of the output format for this problem.

F. The Kings of Mesopotamia

Author: Sean Falconer, March 12, 2004

Mesopotamia is an area of land located between the Tigris and Euphrates rivers in contemporary Iraq. This area was conquered and settled by many groups of people, also, this area is where most experts agree that human civilization began. The ancient Mesopotamians developed written language, arithmetic, the calendar, and basic medicine. They also kept accurate records of various astronomical events. Archeologists are interested in studying the ancient Mesopotamians because they are our "cradle of civilization".

The various groups that settled the area of Mesopotamia were generally ruled by a King. They believed that the King was appointed by the Gods, and his word was the Gods word. When their King would die, they'd have extravagant burial ceremonies in order to honour his life. For instance, the Mesopotamians would often bury their King's possessions throughout Mesopotamia. They would then use the locations of the possessions to determine the location of where to bury the King. Archeologists had mathematicians analyze their findings, and the mathematicians found that the Kings were always located within a triangle formed by considering 3 different possession locations as the vertices. Furthermore, the triangle always has the largest area possible out of all the triangles that can be constructed by considering each possession as a vertex. Finally, this special triangle, never contains any other possessions within it or on its edges.

Given a list of possession locations, you must determine the triangle in which the King was buried.

Consider three points, (x_1, y_1), (x_2, y_2), and (x_3, y_3). You may find the following formula for calculating the area of a triangle connecting these points useful:

$$A = 0.5 \times [(y_3 - y_1) \times (x_2 - x_1) - (y_2 - y_1) \times (x_3 - x_1)]. \tag{4}$$

HINT: In order to figure out if a point is inside a triangle, first draw on paper a triangle with a point inside it. Connect this point with lines to each of the vertices. What do you notice about the three triangles formed by this point joined to the vertices of the original triangle? Now do the same thing, but with a point outside the triangle. What do you notice about the triangles formed for the point outside?

Input

For each King, there are several lines of input. The first line contains an integer N, $3 \leq N \leq 15$, which is the number of possession locations that follow. After this line, there will be N lines, each line begins with a single character which represents the name of the possession, then where will be two integers, x and y, which represent the coordinates of the given possession. The end of input is indicated by a 0 for the number of possessions. This case should not be processed.

Output

For each King, output the possession names that make up the triangle in which the King is located followed by a space, and then the triangle area. The triangle area should be a floating point number, rounded to 2 decimal points. Each test case should be on its own line.

Sample Input

```
6
A 1 0
B 4 0
C 0 3
D 1 3
E 4 4
F 0 6
4
A 0 0
B 1 0
C 99 0
D 99 99
0
```

Sample Output

```
BEF 8.00
BCD 4851.00
```

G. Princess Penelope

Author: Sean Falconer, March 12, 2004

Princess Penelope is the prettiest princess in all the land and she has many many suitors bidding for her attention. Princess Penelope has promised her father that she would choose to marry one of her suitors by the end of this year, but, being a spoiled princess, she is accustomed to a rather expensive lifestyle. This means, she will only marry the richest suitor.

As the court programmer, she has ordered you to help her choose the richest suitor. Each suitor is bidding their largest piece of connected land as a representation of their wealth. However, all the suitors want to win Princess Penelope's affections, so they've been sneaky, and have only submitted a map of their entire property rather than the size of their largest piece of land. It is up to you to find each suitors largest piece of connected land, and then recommend to Princess Penelope the suitor with the largest bid.

Input

On the first line of input there will be one integer, C, which represents how many test cases will follow. Each test case begins with two numbers, N and M, separated by a single space. N represents the number of suitor land descriptions that follow and M represents the width and height of the suitor land maps, $N \leq 5$ and $M \leq 20$. After this line, there will be N suitor maps consisting of M rows and M columns. Each map is represented by 0s and 1s, where 0 represents water, and 1 represents land. Two map coordinates are connected if and only if they are touching horizontally, vertically, or diagonally.

Output

You must output the suitors input rank with the largest bid (ie. the largest connected land component) and the size of his bid. The suitor ranks are determined by their order in the input, that is, the first suitor map has a rank of 1, the second has rank 2, etc. If there is a bidding tie, break the tie by recommending the suitor with the lowest rank. Each test case should be on its own line of output.

Sample Input

```
1
3 5
1 0 0 0 1
1 1 1 0 0
0 0 0 1 0
0 0 0 0 0
0 0 0 0 0
1 1 1 0 0
0 0 0 0 1
0 0 1 0 0
1 0 0 0 0
0 0 0 1 0
1 1 1 0 0
0 0 0 0 1
0 0 1 0 0
1 0 0 0 0
1 1 1 1 1
```

Sample Output

```
3 6
```