Problema 0: Numeros Grandes.

Your task is to write a program which calculates a factorial of a given number not exceeding 1000. (warning: the problem is the size: 1000! has 2568 digits)

Input specifications

Any number of lines, each containing value "n" for which you should provide value of $n!$.

Output specifications

2 lines for each input case. First should contain value "n" followed by character '!'. The second should contain calculated value $n!$.

Sample input

10
30
50
100

Output for sample input

10!
3628800
30!
265252859812191058636308480000000
50!
30414093201713378043612608166064768844377641568960512000000000000
100!
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862530

Problema 1: Tablas de Multiplicar.

There are several definitions of semigroups. In some books, a semigroup is like a group, except that the inverse element may not exist. However, in other books, a semigroup is just a set with a binary operation that is associative. (i.e., it is not required that there exists a neutral elemental). We will use this second definition here. We will only deal with finite semigroups, in fact, our sets will have at most 26 elements. The binary operation may be specified simply by listing for each pair (x, y), the result $x.y$. One way to do this is by means of the multiplication table, which is a matrix indexed by the elements of the set, and in the entry in row x and column y is the element $x.y$. For example, the table for the operation + mod 4 is:

+	0	1	2	3
.	–	–	–	–
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

In this particular case, it is true that the semigroup is commutative, i.e., $a+b = b+a$, but this is not always true.

Input specifications

Write a program that will read the elements of sets together with corresponding "multiplication" tables which denote possible binary operations #. Your program should then determine if the set S with the defined operation constitutes a semigroup. If the set and corresponding table do not form a semigroup, your program should report that the pair do not form a semigroup and state why. If the set and operation pair do form a semigroup, your program should check to see if the semigroup is also a commutative semigroup. Thus, for each set and corresponding table one of the following four results is possible:

NOT A SEMIGROUP: x#y = z WHICH IS NOT AN ELEMENT OF THE SET
NOT A SEMIGROUP: (x#y)#z IS NOT EQUAL TO x#(y#z)
SEMIGROUP BUT NOT COMMUTATIVE (x#y IS NOT EQUAL TO y#x)
COMMUTATIVE SEMIGROUP

In the first three results you should substitute actual elements of the set that yield a counter-example to the definitions for a semigroup and a commutative operation. If more than one counter-example exist, simply use one of your choice.

The first line of the input file contains an integer, n with $1 \leq n \leq 26$. The next line of the input file will contain n unique, lower case letters of the alphabet. These letters represent the elements of the set. Although each letter is unique (no duplicates), they are not necessarily arranged in alphabetical order. The next n lines contain the "multiplication" table that corresponds to the elements in the previous line. Each of these lines will contain n lower case letters. For example, the first such line corresponds to the first row of the body of the table. We will assume that the ordering of the rows and columns of the table coincide with the ordering in the line that defines the elements of the set. After the table, there will be another integer, n whith $0 \leq n \leq 26$. If $n > 0$, another set and corresponding table follows. If $n = 0$ it's EOF.

Output specifications

The output file should contain the following for each set and table found in the input file:

1. List of the elements of S in same order as found in the input file using the following format: $S = \{a, b, c, d\}$

2.A line that starts with a space followed by the characters "#| followed by the n elements of the set (no spaces or commas). For example: #|abcd

3.A line that begins with a space followed by the characters '-+ ' followed by n more dashes '-'. For example: `-+----`

4. List of the n rows and columns of the "multiplication" table in the same order as found in the input file. The ith line of the table should begin with a space followed by the ith element of the set followed by the '|' character followed by the n characters in the ith row of the body of the table (no spaces). For example: *a|abcd*

5. One blank line.

6. One line that reports what your program found to be true. This must be one of the four possible results listed above.

7. A line of 30 dashes.

8. One blank line to separate this report from subsequent reports.

Sample input

```
3
abc
abc
bca
cab
3
abc
abc
bca
cad
4
acdb
aaaa
aaca
aada
aaab
5
abcde
aaaaa
bbabb
cccbc
ddddd
eeeee
0
```

Output for sample input

```
S = {a,b,c}
 #|abc
 -+---
 a|abc
 b|bca
 c|cab

COMMUTATIVE SEMIGROUP
------------------------------

S = {a,b,c}
 #|abc
 -+---
 a|abc
 b|bca
 c|cad

NOT A SEMIGROUP: c#c = d WHICH IS NOT AN ELEMENT OF THE SET
------------------------------

S = {a,c,d,b}
 #|acdb
 -+----
 a|aaaa
 c|aaca
 d|aada
 b|aaab

SEMIGROUP BUT NOT COMMUTATIVE (c#d IS NOT EQUAL TO d#c)
------------------------------

S = {a,b,c,d,e}
 #|abcde
 -+-----
 a|aaaaa
 b|bbabb
 c|cccbc
 d|ddddd
 e|eeeee

NOT A SEMIGROUP: (b#a)#c IS NOT EQUAL TO b#(a#c)
------------------------------
```

Problema 2: Conversion No Religiosa.

Almost all you need to do in this problem is convert numbers from one base to another.

Input specifications

The input for your program will consist of one base conversion per line. There will be three numbers per line. The first number will be the number in the base you are converting from. The second number is the base you are converting from. The third number is the base you are converting to. There will be one or more blanks surrounding (on either side of) the numbers. There are several lines of input and your program should continue to read until the end of file is reached. Bases are 2 through 16, digits are 0,...9,A,...,F. All numbers will be at most 7 digits long (in the approriate base)

Output specifications

The output will only be the converted number. The number should be right justified as a 7-digit number. If the number in the new base has more than 7 digits, then print "ERROR" (without the quotes) right justified in the display.

Sample input

```
1111000  2 10
1111000  2 16
2102101  3 10
2102101  3 15
  12312  4  2
     1A 15  2
1234567 10 16
   ABCD 16 15
```

Output for sample input

```
    120
     78
   1765
    7CA
  ERROR
  11001
 12D687
   D071
```

Problema 3: Algun problema con juegos tenia que haber.

A surprisingly complex game can be played between two players on a simple one-dimensional board with up to 60 holes; each player has counters of one colour (indicated here by letters like W or R) which go into the holes. A player may move a counter anywhere on the board, as long as it does not land on or jump over any other counter. The object of the game is to block the other player so he or she has no allowable moves. In a game with one counter each, the first player can force a win on the first move by moving their counter next to the other counter. Whenever the second player tries to move away, the first player moves next to them. Eventually the second player will have no moves left. If the first player does not take this first move, then the second player can force a win. Convince yourself that these statements are true in this example:

$$\| \| \quad \| \| \quad \| \| \quad \|W\| \quad \| \| \quad \| \| \quad \|R\| \quad \| \| \quad \| \| \quad \| \| \quad \| \| \quad \| \|$$

With two counters each, the game is more complex. For example, in the following situation, if W moves first they can force a win by moving the leftmost counter one square to the right, or the rightmost counter one square to the left. Any other move (for example, moving one of the W counters next to an R counter) will mean than R can force a win. Try it and convince yourself this is true.

$$\| \| \quad \| \| \quad \| \| \quad \|W\| \quad \| \| \quad \| \| \quad \|R\| \quad \| \| \quad \| \| \quad \|W\| \quad \| \| \quad \|R\|$$

Even more complex situations arise with more counters; for example, draws can occur (ie neither player can force a win) as in the following case:

$$\| \| \quad \|W\| \quad \| \| \quad \|W\| \quad \| \| \quad \| \| \quad \|R\| \quad \| \| \quad \|W\| \quad \|R\| \quad \|R\| \quad \| \|$$

It is embarrassing that it is difficult to see how to win such a simple game. Please write a program so that I can play this safely by always knowing the winning first move, or, when there is no winning first move, whether the game is lost or can be drawn. Assume that a player will always win the game if it is in their power, and will always try to avoid defeat.

Input specifications

Each line of input will be a game description, which is a string of up to 60 digits. Each digit represents a hole on the board: '0' for an empty hole, '1' for a counter belonging to the first player, or '2' for a counter belonging to the second player. Both players will have the same number of counters, and there will be at least two empty holes on the board. The input will be terminated by a line consisting of a single zero.

Output specifications

For each input game description, one line of output must be produced. This line must be:

.- '0' if the game cannot be won but can be drawn;

.- '2' if the first player will lose whatever move is made;

.- '1' if there is a winning first move. In this case, the '1' must be followed by a description of the winning move. The winning move must be given by two numbers, the first giving the hole in which the move begins, and the second giving the hole to which the piece moves. The holes are assumed to be numbered from left to right with the leftmost being number 1. If there is more than one winning move, choose the move starting from the smallest numbered hole. If there is more than one of these, choose whichever moves to the smallest numbered hole.

Sample input

```
000200100000
000100200102
000010201002
000010200102
001020020001100002000
010122010122
0
```

Output for sample input

```
1 7 5
1 4 5
1 5 4
2
1 13 15
1 2 1
```

Problema 4: Vamo, vamooo, Argentiiina. . ..

The World Cup has a first round out of which 16 teams classify to the single elimination round: in each match one team is eliminated and the other advances. They are seeded following a complicated and sometimes random process, and then team 1 faces team 2, team 3 faces team 4, etc. The winner of the match between teams 1 and 2 will face the winner of the match between team 3 and 4, etc. For example, the seed could be:

1 Germany 2 Morocco 3 Mexico 4 Bulgaria 5 Italy 6 France 7 Brazil 8 Polland

9 Russia 10 Belgium 11 Denmark 12 Spain 13 Argentina 14 Uruguay 15 England 16 Paraguay

The winner of the first game (say, Germany) faces the winner of the second (say, Mexico). The winner of the third game (say, France) will face the winner of the fourth game (say, Brazil) The winner of the fifth game (say, Belgium) will face the winner of the sixth game (say, Spain). The winner of the seventh game (say, Argentina) will face the winner of the eight (say, England). Then, the winner of the game between Germany and Mexico (say, Germany) will face the winner of the game between France and Brazil (say, France), and the winner of the game between Belgium and Spain (say, Belgium) will face the winner of the game between Argentina and England (say, Argentina). Finally, the winners of Germany-France (say, Germany) and Argentina-England (say, Argentina), will face in the final, the winner of which (say, Argentina) is the World Champion.

For each possible match A vs. B between these 16 nations, you are given the probability that team A wins against B. This (together with the tournament mode explained above) is sufficient to compute the probability that a given nation wins the World Cup. For example, if Argentina wins against Uruguay with 80%, England against Paraguay with 60%, Argentina against England with 70% and Argentina against

Paraguay with 90%, then the probability that Argentina reaches the semi-finals is:

$$80\% * (70\% * 60\% + 90\% * 40\%) = 62.4\%$$

Your task is to write a program that computes the chances of the 16 nations to become the World Champion

Input specifications

The input file will contain just one test case. The first 16 lines of the input file give the names of the 16 countries, from top to bottom according to the seed. Next, there will follow a 16×16 integer matrix P where element $p_{i,j}$ gives the probability in percent that country $\#i$ defeats country $\#j$ in a direct match. Country $\#i$ means the i-th country from top to bottom given in the list of countries. In the example above Germany is #1 and Argentina is #13, so $p_{1,13} = 45$ would mean that in a match between Argentina and Germany, Germany wins with a probability of 45%. Note that matches may not end with a draw, i.e. $p_{i,j} + p_{j,i} = 100$ for all i, j, so that Argentina wins againts Germany with a probability of 55%.

Output specifications

Output 16 lines of the form " XXXXXXXXXX p=Y.YY%", where XXXXXXXXXX is the country's name, left-justified in a field of 10 characters, and Y.YY is their chance in percent to win the cup, written to two decimal places. Use the same order of countries like in the input file.

Sample input

```
Germany
Morocco
Mexico
Bulgaria
Italy
France
Brazil
Polland
Russia
Belgium
Denmark
Spain
Argentina
Uruguay
England
Paraguay
50 65 50 60 55 50 50 65 45 55 40 55 40 55 50 50
35 50 35 45 40 35 35 50 30 40 25 40 25 40 35 35
50 65 50 60 55 50 50 65 45 55 40 55 40 55 50 50
40 55 40 50 45 40 40 55 35 45 30 45 30 45 40 40
45 60 45 55 50 45 45 60 40 50 35 50 35 50 45 45
50 65 50 60 55 50 50 65 45 55 40 55 40 55 50 50
50 65 50 60 55 50 50 65 45 55 40 55 40 55 50 50
35 50 35 45 40 35 35 50 30 40 25 40 25 40 35 35
55 70 55 65 60 55 55 70 50 60 45 60 45 60 55 55
45 60 45 55 50 45 45 60 40 50 35 50 35 50 45 45
60 75 60 70 65 60 60 75 55 65 50 65 50 65 60 60
45 60 45 55 50 45 45 60 40 50 35 50 35 50 45 45
60 75 60 70 65 60 60 75 55 65 50 65 50 65 60 60
45 60 45 55 50 45 45 60 40 50 35 50 35 50 45 45
50 65 50 60 55 50 50 65 45 55 40 55 40 55 50 50
50 65 50 60 55 50 50 65 45 55 40 55 40 55 50 50
```

Output for sample input

```
Germany    p=8.54%
Morocco    p=1.60%
Mexico     p=8.06%
Bulgaria   p=2.79%
Italy      p=4.51%
France     p=7.50%
Brazil     p=8.38%
Polland    p=1.56%
Russia     p=9.05%
Belgium    p=3.23%
Denmark    p=13.72%
Spain      p=3.09%
Argentina  p=13.79%
Uruguay    p=3.11%
England    p=5.53%
Paraguay   p=5.53%
```

Problema 5: Power index.

In an assembly in which every member has one vote, each member has the same power. In some cases, however, members have different number of votes. For example, in the International Monetary Fund, member countries do not have the same numbers of votes: countries which contribute more to the fund have more votes. Another example: in the Mercosur, Brazil and Argentina have more votes than Paraguay or Uruguay. (and Brazil more than Argentina).

However, sometimes the numbers do not reflect the actual voting power of the members. For example, suppose a three-member council, with members A and B having 1000 votes each, and member C having only 1 vote. It would seem that the power of C is infinitesimal, compared with the power of A or B. However, if you need a majority of votes to approve something, then the power of A,B and C are equal!!. This is because in this case the only way a member can lose a vote is if that member is alone supporting the vote: given a proposal P, if any two members support the proposal, then the proposal will be approved, so the relative power of A,B and C are the same.

Another typical situation is in parliaments with political parties with strong disciplines. Although each member has one vote, under the political party system, each party has some measure of voting strenght, assuming that all its members vote together. People tend to believe that parties with the more votes have more power, but as the above example show, this is not neccesarily true. (It is still true that if party A has more votes than party B, then the power of party A cannot be less than the one of B, but they may be equal)

Several theories have been developed to try to measure the "true" power of members. One possibilty could be, given a member, to count, of the total number of possible coalitions which have it as a member, how many of those coalitions are winning coalitions (i.e., coalitions that have a majority of the vote). This however do not reflect the true power: suppose that party A has 51 votes out of a hundred, the other 49 divided among the other parties. (say, 4 other parties). Then A would be in 16 winning coalitions, and each of the other parties will be in 8 winning coalitions. This would seem to imply that the power of A is twice that power of the other parties, but this is clearly not so: the power of A is absolute. So John F. Banzhaf III proposed a slight refinement of this idea. He proposed that a party's power is determined by the number of minority (losing) coalitions that it can join and turn into a (winning) majority coalition. This number is called the Banzhaf power index, or simply the power index. Note that the empty coalition is also a minority coalition and that a coalition only forms a majority when it has more than half of the total number of votes.

In the example given above, the power index of A is 16, while the power index of any other member is 0. In the first example, a coalition needs 1001 votes, so the power index of A is 2 (it turns coalitions B and C alone , which are losing coalitions, into coalitions A,B and A,C, which are winning. The same is true obviously of B, but also of C, sinve it turns the losing coalitions A and B into the winning coalitions A,C and B,C.

Another example: if A has 7 votes, B 4, C 2 and D and E 6 votes each, then the total of votes is 25, so a winning coalition needs 13 votes. A,B is a losing coalition, but adding C to the coalition makes it a winning coalition. D,E is also a losing coalition that turns into a winning coalition with C on it. However, any other coalition without C is either already winning or it is losing, adding C to it does not make it winning. So the power index of C is 2. However, B, with twice as many votes, also has power index 2. D and E have power index 6, but A with only one more vote, has power index 10.

Your task is to write a program that computes for each party its power index.

Input specifications

The first line contains a single integer which equals the number of test cases that follow. Each of the following lines contains one test case. The first number on a line contains an integer P in [1,...,20] which equals the number of parties for that test case. This integer is followed by P positive integers, separated by spaces. Each of these integers represents the number of members of a party in the electoral system. The$i$-th number represents party number i. No electoral system has more than 1000 votes.

Output specifications

For each test case, you must generate P lines of output, followed by one empty line. P is the number of parties for the test case in question. The i-th line (i in [1,...,P]) contains the sentence:

party i has power index I

where I is the power index of party i.

Sample input

```
3
5 7 4 2 6 6
6 12 9 7 3 1 1
3 2 1 1
```

Output for sample input

```
party 1 has power index 10
party 2 has power index 2
party 3 has power index 2
party 4 has power index 6
party 5 has power index 6

party 1 has power index 18
party 2 has power index 14
party 3 has power index 14
party 4 has power index 2
party 5 has power index 2
party 6 has power index 2

party 1 has power index 3
party 2 has power index 1
party 3 has power index 1
```

Problema 6: Ayudemos a los físicos!.

While studying certain extragalactical phenomena, a group of phycisists developed a new theory on star evolution.

The key to the theory lies in a certain infinite matrix A.

They discovered that if, for each n, they take the submatrix $A[n]$ formed by the first n rows and n columns of A, then the determinant of $A[n]$ is of some importance to the theory, specifically, certain term in the expansion of the determinant.

Let's recall that the determinant can be expanded by the formula:

$$det(A) = \sum_{\sigma \in S_n} sg(\sigma) \prod_{i=1}^{n} A_{i,\sigma(i)}$$

where S_n is the group of permutations of $\{1, ..., n\}$ and $sg(\sigma)$ is the sign (± 1) of the permutation, in which we are not interested, since the phycisists in this case only care about the absolute value of the term.

They think that the term with the greatest absolute value is a number that is of key importance in one of the theorems of their theory. You need to help them calculate it.

The matrix A has entry i, j equal to:

$$A_{i,j} = e^{-|(i+6)(j-5)+28|}$$

Input specifications

In each line, an integer $n \geq 0$ will be presented. $n = 0$ will mean the end of the input file. No n will be greater that 409.

Output specifications

For each line n of the input file with $n > 0$ you have to output one line with "n :" and to its right, leaving one space, the natural logarithm of the term with greatest absolute value in the expansion of the determinant of $A[n]$ as explained above. The result should always be an integer.

Sample input

2
10
15
22
100
397
0

Output for sample input

2: -4
10: -327
15: -931
22: -2520
100: -178396
397: -10593394

Problema 7: Y bueno, otro juego tenia que haber (pero sin estrategia esta vez).

Stan and Ollie will play a game. Except that in this game it is not tested at all their intelectual abilities.

Each is given a number of balls (the same number to each), and an unlimited number of (numbered) baskets, Stan having the blue baskets and Ollie the red baskets. Both red and blue baskets are numbered 1,2,3, etc. They have to put the balls in the basket, subject to the following conditions:

1) The number of balls in basket i must be greater than the number of balls in basket $i + 1$.

2) Stan must use an odd number of baskets, and Ollie an even number of baskets.

They each start, and fill the baskets. Once both finished, the results are written down, the baskets are emptied, and they start again, the numbers are written down, etc.

The loser is the first one to repeat a previous pattern of filling, or the one who cannot play anymore without violating this condition. Assume that in fact they never violate the condition, i.e., they play perfectly until they cannot do so.

For example, with 6 balls, Stan can put 6 balls in one basket, or else 3 in the first, 2 in the second and 1 on the third, and then he cannot play anymore. Meanwhile, Ollie can put 5 in the first and 1 in the second, or else 4 in the first and 2 in the second, and then he cannot play anymore. In this case the game is a draw.

With 5 balls however, Stan can put 5 in the first, and cannot play anymore, while Ollie can put 4 in the first and 1 in the second, or 3 in the first and 2 in the second, so Ollie wins.

With 2 balls, Stan can put 2 balls in the first, but Ollie cannot play at all, so Stan wins.

Input specifications

A number of lines, each with a single integer n with $0 \leq n \leq 1000000$. $n = 0$ terminates the input.

Output specifications

For each line of the input file with $n > 0$, output a line "Stan wins", "Ollie wins" or "Draw", depending on the result of the game.

Input specifications

2
5
6
15
1000
1001
996745
1000000
0

Output for sample input

Stan wins
Ollie wins
Draw
Stan wins
Draw
Ollie wins
Stan wins
Draw

Problema 8: Genetic Algorithms.

In certain problems a strategy is required. The best strategy may not be clear or they may not be any way to develop a theory to find one. Sometimes the use of genetic algorithms can solve the problem.

An application of genetic algorithms to this problem will consist in creating individuals with some strategy, let them use their strategy some time, and then find out which strategies were "better" with respect to some measure. The "best" strategies are given more chances of reproducing, and you also need some way to "mix" strategies between individuals, and some chances of "mutation". This way, you create new individuals, presumably "better adapted". You repeat the process through several generations, until you get the "best" stategies, according to your model of evolution.

In our case, we will have the following problem: i items will be available. (e.g, food) and they must be consumed in exactly t units of time (eg, days).

In each unit of time at least one item must be consumed. (so we must have $i \geq t$). A "strategy" is needed as to how many items will be consumed every unit of time.

In this context a "strategy" is simply a FIXED amount of items that will be consumed each unit of time.

This strategy will be subject to several "market conditions" that are not known in advance. Some strategies will be better in some situations, others in others. For example, a strategy may be to consume all items the first unit of time, except for enough items to consume one in each subsequent unit of time. In an inflationary environment, for example, this wil be the best strategy,but it will not be if there is deflation.

Your problem is not to determine the best strategy or even to write the genetic algorithm, but to help the person in charge of that.

The problem that person faces is that there would be several is and ts at some moment, and in part of the genetic algorithm that will compute the rate of mutation, a certain number has to be calculated. That number will consists of a quotient. If $S(i,t)$ denotes the total number of strategies for i and t fixed, in part of the genetic program some parameters $(i_1,t_1),(i_2,t_2),...(i_n,t_n)$ will have been selected at random as well as some other parameters $(\hat{i}_1,\hat{t}_1),(\hat{i}_2,\hat{t}_2),...(\hat{i}_m,\hat{t}_m)$.

The program call for the calculation of the number:

$$X = \frac{S(i_1,t_1)S(i_2,t_2)...S(i_n,t_n)}{S(\hat{i}_1,\hat{t}_1)S(\hat{i}_2,\hat{t}_2)...S(\hat{i}_m,\hat{t}_m)} . \frac{i_1 i_2 ... i_n \hat{t}_1 \hat{t}_2 ... \hat{t}_m}{\hat{i}_1 \hat{i}_2 ... \hat{i}_m t_1 t_2 ... t_n}$$

and it is your task to do it.

However, if the result is not an integer,or is too big, the statistic cannot be used, and a different number is computed.

Also, sometimes a parameter (i,t) with $t = 0$ will be selected. In that case, i, t and $S(i,t)$ are not used in the above computation.

Input specifications

There will be several test cases. Each test case will start with two positive integer, n and m, both less than 101. n lines will follow, each with two integers,$(i_1,t_1),(i_2,t_2),...(i_n,t_n)$. The integers will be separeated with a space (no parentesis or commas). After that there will be m more lines, each with two integers, $(\hat{i}_1,\hat{t}_1),(\hat{i}_2,\hat{t}_2),...(\hat{i}_m,\hat{t}_m)$. In each case, $0 \leq t \leq i \leq 5000$. Input will be terminated by EOF.

Output specifications

For each test case show X as specified above. If the result is not an integer, show 0 instead. If the result is an integer but has more than 100 digits, then print -1 instead.

Sample input

```
4 1
10 5
10 5
10 5
10 5
100 100
4 4
547 256
547 349
547 333
547 168
547 198
547 214
547 291
547 379
3 3
10 5
10 4
10 3
10 7
10 6
10 1
3 3
10 5
10 4
10 3
10 7
10 6
10 10
```

Output for sample input

```
4032758016
1
0
252
```

Problema 9: Donde hay que deducir que tienen que hacer.

Your task in this problem is first, deduce what you have to do from the sample input/output, and second, to do it.

Input specifications

standard input. (terminated by EOF)

Output specifications

standard output.

Sample input

The judges from the programming contests are known to be very mean and very lazy. We, judges, want less work and more Wrong Answers! So, we'd like you to help us and write an automated judge script to judge solution runs from teams all over the world. All you have to do is write a program which receives the standard solution and a team output and gives as answer one of the following messages: "Accepted", "Presentation Error" or "Wrong Answer". We define each one as:

Accepted: As we are very mean judges, we only want you to give "Accepted" as answer if the team output matches the standard solution integrally. That is, ALL characters must match and must be in the same order.

Presentation Error: We want you to give "Presentation Error" if all NUMERIC characters match (and in the same order) but there is at least one non-numeric character wrong (or in wrong order). For instance, "15 0" and "150" would receive a "Presentation Error", whereas "15 0" and "1 0" would not (it would receive "Wrong Answer", see bellow).

Wrong Answer: If the team output could not be classified as any of the two above, then you have no option but to give "Wrong Answer" as an answer!

Input Specifications

The input will consist of an arbitrary number of input sets. Each input set begins with a positive integer n< 100, alone in a line, which describes the number of lines of the standard solution. The next n lines contain the standard solution. Then there is a positive integer m < 100, alone in a line, which describes the number of lines of the team output. The next m lines contain the team output. The input is terminated by a value of n = 0, and should not be processed. No line will have more than 120 characters.

Output Specifications For each set you should output one of the following lines: (x is the number of the input set, starting at 1).

Output for sample input

Th jdgs frm th prgrmmng cntsts r knwn t b vry mn nd vry lzy. W, jdgs, wnt lss wrk nd mr Wrng nswrs! S, w'd lk y t hlp s nd wrt n tmtd jdg scrpt t jdg sltn rns frm tms ll vr th wrld. ll y hv t d s wrt prgrm whch rcvs th stndrd sltn nd tm tpt nd gvs s nswr n f th fllwng mssgs: "ccptd", "Prsnttn rrr" r "Wrng nswr". W dfn ch n s:

ccptd: s w r vry mn jdgs, w nly wnt y t gv "ccptd" s nswr f th tm tpt mtchs th stndrd sltn ntgrlly. Tht s, LL chrctrs mst mtch nd mst b n th sm rdr.

Prsnttn rrr: W wnt y t gv "Prsnttn rrr" f ll NMRC chrctrs mtch (nd n th sm rdr) bt thr s t lst n nn-nmrc chrctr wrng (r n wrng rdr). Fr nstnc, "15 0" nd "150" wld rcv "Prsnttn rrr", whrs "15 0" nd "1 0" wld nt (t wld rcv "Wrng nswr", s bllw).

Wrng nswr: f th tm tpt cld nt b clssfd s ny f th tw bv, thn y hv n ptn bt t gv "Wrng nswr" s n nswr!

npt Spcfctns

Th npt wll cnsst f n rbtrry nmbr f npt sts. ch npt st bgns wth pstv ntgr n < 100, ln n ln, whch dscrbs th nmbr f lns f th stndrd sltn. Th nxt n lns cntn th stndrd sltn. Thn thr s pstv ntgr m < 100, ln n ln, whch dscrbs th nmbr f lns f th tm tpt. Th nxt m lns cntn th tm tpt. Th npt s trmntd by vl f n = 0, nd shld nt b prcssd. N ln wll hv mr thn 120 chrctrs.

tpt Spcfctns Fr ch st y shld tpt n f th fllwng lns: (x s th nmbr f th npt st, strtng t 1).