11th México and Central America
ACM International Collegiate Programming Contest 2006
Problem No. 1 – Encrypting Passwords.
Input file: encrypting.in **Output file:** Standard Output

Chip and Dale have devised an encryption method to hide their passwords. They first agree secretly on a number that will be used as the number of columns in a matrix. The sender prepares an intermediate format using the following rules:

1. - The password is formed with lowercase letters only.
2. - Each letter will be represented in binary code according to ASCII table.

The sender then enters the letters of the intermediate format along the diagonals of the matrix and pads with extra characters '0' (zero) so as to make a rectangular array of letters. These additional characters do not comprise the Password

For example, if the password is: "iteso" and there are five columns, Dale would write down



For example, for the letter 'i':

Letter = i
ASCII Code = 105
Binary Code = 1 1 0 1 0 0 1

Note how Dale includes only the binary code of each letter and writes them all in lower case. Dale then sends the password to Chip by writing the binary code in each row. So, the password in its intermediate format would be encrypted as

```
11110001110100110010111100111001100110010000
```

Your job is to recover the password that was sent from Chip to Dale.

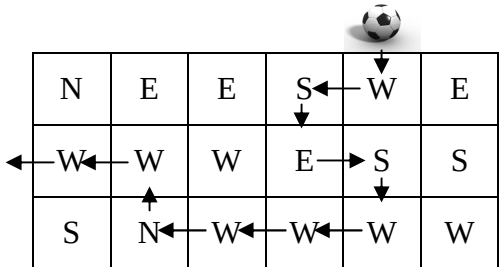
Input
There will be multiple input sets. Input for each set will consist of two lines. The first line will contain an integer in the range 2 . . . 20 indicating the number of columns used. The next line is a string of up to 400 characters (1's and 0's). The last input set is followed by a line containing a single zero (0). This line should not be processed.

Output
Each input set should generate one line of output, giving the password.

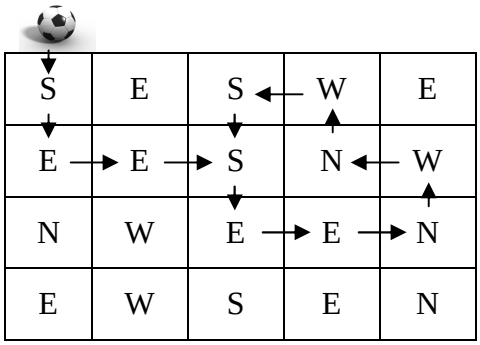
Sample Input
```
5
11110001110100110010111100111001100110010000
3
11000101010101101101
0
```

Sample Output
```
iteso
acm
```

11th México and Central America
ACM International Collegiate Programming Contest 2006
Problem No. 2 – Football Foundation (FOFO).
Input file: fofo.in **Output file:** Standard Output



GRID 1



GRID 2

The football foundation (FOFO) has been researching on soccer; they created a set of sensors to describe the ball behavior based on a grid uniformly distributed on the field. They found that they could predict the ball movements based on historical analysis. Each square sensor of the grid can detect the following patterns:

`N` north (up the field)
`S` south (south the field)
`E` east (to the right on the field)
`W` west (to the left on the field)

For example, in grid 1, suppose the ball was thrown into the field from north side into the field. The path the sensors detected for this movement follows as shown. The ball went through 10 sensors before leaving the field.

Comparing with what happened on grid 2, the ball went through 3 sensors only once, and started a loop through 8 instructions and never exits the field.

You are selected to write a program in order to evaluate line judges job, with the following out put based on each grid of sensors, the program needs to determine how long it takes to the ball to get out of the grid or how the ball loops around.

Input
There will be one or more grids of sensors for the same game. The data for each is in the following form. On the first line are three integers separated by blanks: The number of rows in the grid, the number of columns in the grid, and the number of the column in which the ball enters from the north. The grid column's number starts with one at the left. Then come the rows of direction instructions. The lines of instructions contain only the characters `N`, `S`, `E` or `W`, with no blanks. The end of input is indicated by a grid containing `0 0 0` as limits.

Output
For each grid in the input there is one line of output. Either the ball follows a certain number of sensors and exits the field on any one of the four sides or else the ball follows the behavior on some number of sensors repeatedly. The sample input below corresponds to the two grids above and illustrates the two forms of output. The word "`step`" is always immediately followed by "`(s)`" whether or not the number before is 1.

Sample Input

```
3 6 5
NEESWE
WWWESS
SNWWWW
4 5 1
SESWE
EESNW
NWEEN
EWSEN
0 0 0
```

Sample Output

```
10 step(s) to exit
3 step(s) before a loop of 8 step(s)
```

11th México and Central America
ACM International Collegiate Programming Contest 2006
Problem No. 3 – Similar Structures.
Input file: simi.in **Output file:** Standard Output

After September 11, 2001, improve the security has become a crucial challenge for intelligence departments around the world. A crucial issue about this task is to identify terrorism patterns represented by attacks in different locations (dots), relations between terrorism activities (connections) and, attributes (dots and connections labels) that describes the acts committed by political groups or persons. A natural representation for this information is a graph, and the ability to discover and test known patterns in such information could lead to significant improvement in our ability to identify potential threats.
Based on the above problem description, our task consists on developing a tool that identifies a terrorism patterns into a set of potential terrorism activities using a graph-based representation where all graphs are undirected and labeled (thus, all vertices and edges has a label). The task of this problem consists on finding if there exist a subgraph S of G where S have the same number of vertices and edges of G' and there exist one to one association between vertices and edges from S and G' with the same label (isomorphism).

Input
The input will be a file called simi.in, where different terrorism activities are represented by different graphs. Consider that each graph have a name, where graph "XS GX" represents a recognized terrorism pattern associated to a terrorism group and "XP GX" represents a set of attacked locations. By notation, "X" is an integer number between 1 to 5, where each of one is a test. Consider that "XS G1" is the graph to search for in "XP G1" and so on. Moreover, "XS G1" appears in simi.in immediately before of "XP G1".

For all cases, #vertices of "XS GX" is lower than #vertices of "XP GX" and 0 < #vertices <= 150 for each "XP GX" and for all graphs, 0 <= #edges <= 5000. Also, vertex and edge labels contains 500 printable ASCII characters (or less) without single spaces.

Both "XS GX" and "XP GX" are represented based on the following format. First, there is a line with the graph name. As an example, the line for "XS G1" is:

```
XS G1
```

Next, there are *k* lines (each line represents a vertex), 0 < *k* <= 150, each one using the following format:

```
v id label
```

Consider that id is a positive integer, 0 < id <= 150, and label is a string (vertex' label). There are not two lines with the same id value and all lines are sorted in an increased order based on id.

The next *l* lines represent all edges into the graph, each one using the following format:

```
e idx idy label
```

Consider that idx and idy are positive integers associated to a vertex into the graph. On the other hand, label is a string (edge' label).

Output
For each pair of graphs ("XS GX", "XP GX"), X = 1, ..., 5, if it does not exist a subgraph S of "XS Gi" where "XP Gi" is isomorphic to S, print the message:

```
XS_Gi_<-->_XP_Gi
There are not instances!
```

Otherwise, each instance is reported based on the following format. First, print the message:

```
XS_Gi_<-->_XP_Gi
Vertices_(G')_<-->_Vertices_(G)
```

The next lines are the vertices and edges from G' associated to vertices and edges from G. The format is as follows:

{vX,_vY}_<-->_{vA,_vB}

where X, Y, A and B are positive integers and each of one represents a vertex valid id, where X < Y. All lines are sorted based on the id's of G' in an increasing order:

Line n: {vX,_vY}_<-->_{vA,_vB}
Line n+1: {vX',_vY'}_<-->_{vA',_vB'}
Exactly when X < X' or X = X' and Y < Y'

Restrictions: If there exist a isomorphic subgraph, it is unique. There are not empty lines for inputs and output files. There are not two different edges between a pair of vertices.

Sample Input

```
XS G1
v 1 a
v 2 b
v 3 a
v 4 b
e 1 2 d
e 1 3 d
e 1 4 c
e 2 3 d
XP G1
v 1 b
v 2 a
v 3 b
v 4 a
e 1 2 d
e 1 3 d
e 1 4 d
e 2 3 c
e 2 4 d
e 3 4 e
XS G2
v 1 house
v 2 attacked
e 1 2 yesterday
XP G2
v 1 house
v 2 attacked
e 1 2 yesterday
XS G3
v 1 WTC
v 2 AlQaeda
e 1 2 atacked_by
XP G3
v 1 WTC
v 2 ETA
e 1 2 atacked_by
```

Sample Output

```
XS G1 <--> XP G1
Vertices_(G’)_<-->_Vertices_(G)
{v1,v2} <--> {v2,v1}
{v1,v3} <--> {v2,v4}
{v1,v4} <--> {v2,v3}
{v2,v3} <--> {v1,v4}
XS G2 <--> XP G2
Vertices_(G’)_<-->_Vertices_(G)
{v1,v2} <--> {v1,v2}
XS G3 <--> XP G3
There are not instances!
```

11th México and Central America
ACM International Collegiate Programming Contest 2006
Problem No. 4 – Medium Size Summations (R2D2).
Input file: r2d2.in **Output file:** Standard Output

R2-D2 (our well known friendly robot) needs to perform some operations quickly to save his space ship. These operations require computing long summations and a division. Moreover, he needs to find the exact solution and he is required to present a report with the results. For that, he needs to simplify his solution as much as possible.

We assume that there is an array available (X1, X2 , ….) of 99999999 elements.
The array has the peculiar property that the average of the first K numbers is equal to the average of the index K and the number 1.

R2-D2 needs to do the following: Given a natural number N less than 99999999, his assignment is to compute the function:

F(N)= F1(N) / F2(N) where:
$F1(N) = N \times (\Sigma_{\{0 \le k \le N\}} k^4 / X_k)$
$F2(N) = (\Sigma_{\{0 \le k \le N\}} k^3 / X_k) \times (\Sigma_{\{0 \le k \le N\}} k^2 / X_k)$
That is, $F1(N)= N \times (1/X_1 + 16/X_2 + \ldots + N^4 / X_N)$ and
$F2(N)= (1/ X_1 + 8/X_2 + ... + N^3/X_N) \times (1/X_1 + 4/X_2 + ... + N^2 / X_N)$

Since R2-D2 needs an exact solution, we ask him to report the following:
The solution needs to be given as a pair of relative prime numbers a,b such that f(N)= a/b if the solution is not an exact integer. Otherwise just give the exact integer. The numbers processed by R2-D2 were of eight digits (99999999). Remember that R2-D2 was built long long time ago. His circuits are not that fast but he is clever. R2-D2 was able to perform one of these operations in less than one second. Can you do this assignment as fast as R2-D2 did it???

Input
You will receive an input line with natural numbers separated by comma. Each number is less than 99999999. You will receive no more than 20 numbers.

Output
You need to give a sequence of lines each one with the solution of the corresponding input case. The solution is either a pair of natural numbers separated by the symbol / representing the pair a,b mentioned above (when the division is not exact) or just one natural number (when the division is exact). Notice that these numbers could require more than 8 digits.

Sample Input
```
1,2
```

Sample Output
```
1
6/5
```

NOTE: The computing time expected of your algorithm is 10 seconds.

11th México and Central America
ACM International Collegiate Programming Contest 2006
Problem No. 5 – The Inheritance
Input file: inheritance.in **Output file:** Standard Output

Mr. Smyth wants to leave his son, Peter, a part of his land as inheritance. This is a very big piece of land which is in the shape of a simple polygon. The only condition is that his son has to start building walls within his father's land in such a way that he has an area of land that is also in the shape of a simple polygon.

Peter is a very lazy person and prefers to pay someone to build the walls. However, the people contracted by Peter forgot the boundaries that limit Mr. Smyth's land. But they did not worry about this and went ahead and built the walls. On the contrary, if they told Peter that they had forgotten the boundaries of Mr. Smyth´s land Peter could get angry and he may fired his employee.

Mr. Smyth knows his son very well and he will only leave land to his son if he builds walls that do not invade the land of his neighbors', (that is, no wall must be outside Mr. Smyth's land), and the walls built by his son must form an enclosed area.

Finally, Mr. Smyth wants to know the area of the land left for himself.

Input
The input begins with a single positive integer on a line by itself indicating the number of the cases (a integer between 1 and 20) following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

- The next line defines Mr. Smyth's land which is a simple polygon (without crossing edges or touching borders), defined by a list of vertexes (there are at most 100 vertices in a polygon); each vertex is defined by its coordinates (x,y) in decimal format (positive real numbers less than 1000). The vertexes are ordered according to their sequence in the polygon.

- The next line contains the number NB which defines the number of walls (there are at most 100 walls).

- Next, NB lines are defined, each line having 4 integer numbers (positive integers less than 1000) X_0 Y_0 X_1 Y_1 (separated by a single space), which represent the starting points and end points of each wall (the walls don't have any order)

Output
For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

- If Peter doesn't invade the neighbors' land and he obtained an enclosing area, then the output has a line which states "The area free for your father is ***totalarea"*** where ***totalarea*** is the total area of Mr. Smyth's land minus the area of Peter's land. ***totalarea*** must be printed exact to three digits to the right of the decimal point (round up).
  Followed by the Peter's inheritance in next line, this part must be represented by coordinates (x,y) ordered according to their sequence in the polygon, beginning with P0 and P1 of first wall and so on.
- If Peter invades the neighbors' land , output should contain a line stating "Trying to steal neighbors' land".
- If Peter does not invades the neighbors' land and does not succeed in enclosing an area with the walls built, then the output should contain a line skating "Sorry, your assignment do not deserve an inheritance"

Sample Input

```
3

2 2 16 2 14 12 4 12
7
5 7 8 10
13 6 11 3
5 5 8 5
5 5 7 5
8 10 12 9
12 9 13 6
11 3 8 5

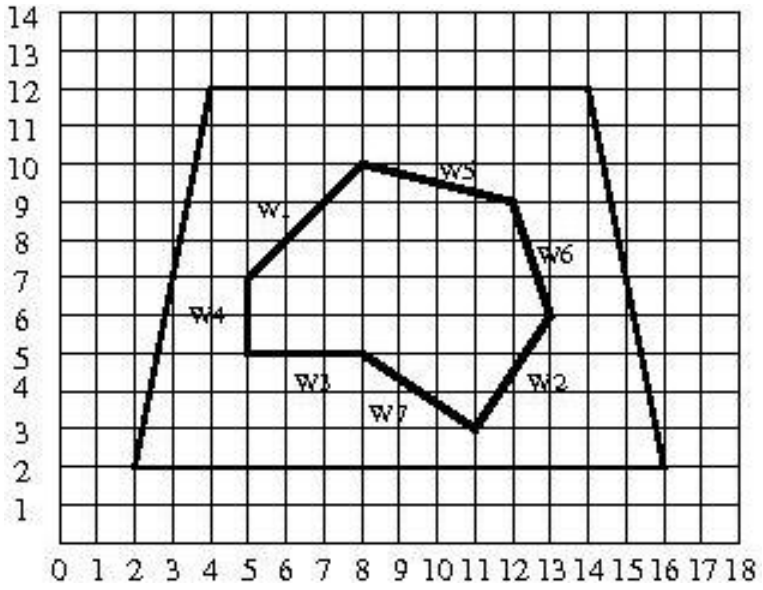
2 2 16 2 14 12 4 12
7
5 7 8 10
13 6 11 3
5 5 8 5
5 5 7 5
8 10 16 11
16 11 13 6
11 3 8 5

10 10 20 10 20 20 10 20
5
15 15 15 19
25 19 25 17
17 17 17 15
15 19 25 19
17 17 25 17
```

Sample Output

```
The area free for your father is: 85.000 and your land is: 5 7 8 10 12 9 13 6 11 3
8 5 5 5
Trying to steal neighbours' land
Trying to steal neighbours' land
```

The picture corresponds to the first case in the sample input



11th México and Central America

ACM International Collegiate Programming Contest 2006

Problem No. 6 – The Explorer.

Input file: explorer.in **Output file:** Standard Output

An explorer must decide where to construct a way to arrive from a point A to point B. In order to help him, the explorer has made a map with the obstacles that exist. He drew into squares his map and he wants a way that passed through the smaller number of squares. The way only can go from a square to another one if they have a side in common, that is to say, cannot advance in diagonal, and it cannot happen through a square that contains an obstacle. Each square of the map is identified by its coordinates, column and row. The columns are numbered of left to right initiating with the 0. The rows are numbered from top to down initiating with the 0.

Input

There will be multiple input sets. The first line contains the incoming number of input to evaluate. Each input will be formed of the following way: first, a line containing two integer numbers, N and M, separated by spaces, indicate the number of columns and lines, where $1 \leq N, M \leq 100$. In each one of the following M lines there is N numbers, separated by spaces, that can be 0 or 1, 0 if there is not obstacle in corresponding square and 1 if there is it. In the penultimate line the column and row of point A. In the last line the column and row of point B. All the cases of test will have at least a way to arrive from A at B.

Output

The number of squares through which it passes a minimum way between A and B.

Sample Input

```
2
3 3
1 0 0
1 0 0
0 0 1
2 0
0 2
5 4
0 1 0 0 0
0 0 1 1 0
0 1 0 0 0
0 0 0 0 0
1 3
2 0
```

Sample Output

```
5
9
```