Do NOT read the problems before contest starts.

Contest URL: http://www.link.cs.cmu.edu/contest/current/.

acm International Collegiate Programming Contest

IBM event sponsor

2014 Carnegie Mellon University Programming Contest

and

ACM ICPC Team Selection Qualification

Round 1

6:35pm-10:05pm, Sept 10, 2014

PROBLEMS

A:	Binary Tree
B:	Strange Billboard
C:	Phone Cell
D:	Key Task
E:	Weird Numbers
F:	Rectangular Polygons
G:	Minimizing Maximizer
H:	Hexagonal Parcels
I:	House Cleaning

Hints:

1. Problems are not sorted from easiest to hardest. It is important to solve the easiest problems first to minimize your penalty score. Usually, you can guess the difficulty of the problems by looking at the scoreboard.
2. Be careful about the efficiency of input/output. If the input size is larger than 10^5 bytes, try to use "`scanf / printf`" in C++ and "`BufferedReader / BufferedWriter`" in Java.
3. Do not access the internet, except for language support and our online judge for submissions.

Binary Tree

Binary Tree is a tree data structure where each node has at most two children, usually they are distinguished as **left** and **right** child. And the node having the children are called parent of those children.

An instruction string is a string consisting of the letters *L*, *R* and *U*. *L* stands for *Left*, *R* for *Right* and *U* for *Up*. Meaning of these will be clear shortly.

One day I have drawn an infinitely large Binary Tree. In this tree each node has exactly two children (*left child* and *right child*) and each of them has a parent. For this problem, we will consider **the parent of the root is root itself**. I put a pen in the root and follow the instruction string **S**. That is, we look at the first character if it says *L* we go to *left child*, if it says *R* we go to *right child* and if it says *U* then to *the parent*. If we receive a *U* instruction at root we just end up at root since we assumed parent of the root is root itself.

Now we have another instruction string **T**. Starting from the node where we are after following the instruction string **S**, we will follow the instruction string **T**. But this time, if we wish we may skip any instruction in the string **T** (possibly discarding all of them). You have to tell me how many different nodes I can end up after following instruction string T (skipping as many instructions as I wish).

For example:

Suppose: **S** = **L** and **T** = **LU**. Our answer is 3. Following **S** we will end up at the left child of the root. Now, when we follow **T**, there may be 4 cases:

i Skipping all letters: we will be at the **same node** where we are.

ii Skipping L and following U: we will be at the **root**.

iii Following L and Skipping U: we will be at the **left child** of current node.

iv Following both L and U: we will be at the **same** node as in case *i*.

Since 3 different nodes we can end up after following **T**, the answer is 3.

INPUT

First line of the test file contains an integer **N** (<= 15) denoting number of test cases. Hence follow **N** test cases. Each test case consists of two non empty strings. First line will contain instruction string **S** and the second line will contain the instruction string **T**. You may assume that there will not be any letter other than L, R or U in these strings. Length of the strings will not be greater than 100000.

OUTPUT

For each test case print the case number followed by the number of nodes we can end up finally. Since the answer may be large, you have to give the answer modulo **21092013**.

SAMPLE INPUT

```
2
L
LU
L
L
```

SAMPLE OUTPUT

```
Case 1: 3
Case 2: 2
```

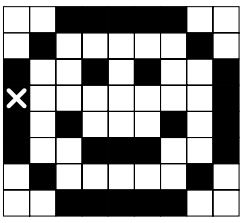
(This page is intentionally left blank)

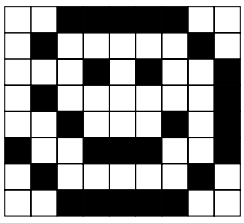
Strange Billboard

The marketing and public-relations department of the Czech Technical University has designed a new reconfigurable mechanical Flip-Flop Bill-Board (FFBB). The billboard is a regular two-dimensional grid of $R \times C$ square tiles made of plastic. Each plastic tile is white on one side and black on the other. The idea of the billboard is that you can create various pictures by flipping individual tiles over. Such billboards will hang above all entrances to the university and will be used to display simple pictures and advertise upcoming academic events.

To change pictures, each billboard is equipped with a "reconfiguration device". The device is just an ordinary long wooden stick that is used to tap the tiles. If you tap a tile, it flips over to the other side, i.e., it changes from white to black or vice versa. Do you agree this idea is very clever?

Unfortunately, the billboard makers did not realize one thing. The tiles are very close to each other and their sides touch. Whenever a tile is tapped, it takes all neighboring tiles with it and all of them flip over together. Therefore, if you want to change the color of a tile, all neighboring tiles change their color too. Neighboring tiles are those that touch each other with the whole side. All inner tiles have 4 neighbors, which means 5 tiles are flipped over when tapped. Border tiles have less neighbors, of course.





For example, if you have the billboard configuration shown in the left picture above and tap the tile marked with the cross, you will get the picture on the right. As you can see, the billboard reconfiguration is not so easy under these conditions. Your task is to find the fastest way to "clear" the billboard, i.e., to flip all tiles to their white side.

Input Specification

The input consists of several billboard descriptions. Each description begins with a line containing two integer numbers R and C ($1 \leq R, C \leq 16$) specifying the billboard size. Then there are R lines, each containing C characters. The characters can be either an uppercase letter "`X`" (black) or a dot "`.`" (white). There is one empty line after each map. The input is terminated by two zeros in place of the board size.

Output Specification

For each billboard, print one line containing the sentence "`You have to tap` T `tiles.`", where T is the minimal possible number of taps needed to make all squares white. If the situation cannot be solved, output the string "`Damaged billboard.`" instead.

Sample Input

```
5 5
XX.XX
X.X.X
.XXX.
X.X.X
XX.XX

5 5
.XX.X
.....
..XXX
..X.X
..X..

1 5
...XX

5 5
...X.
...XX
.XX..
..X..
.....

8 9
..XXXXX..
.X.....X.
X..X.X..X
X.......X
X.X...X.X
X..XXX..X
.X.....X.
..XXXXX..

0 0
```

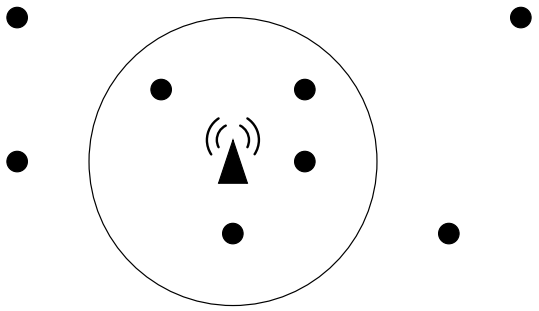
Output for Sample Input

```
You have to tap 5 tiles.
Damaged billboard.
You have to tap 1 tiles.
You have to tap 2 tiles.
You have to tap 25 tiles.
```

Phone Cell

Nowadays, everyone has a cellphone, or even two or three. You probably know where their name comes from. Do you? Cellphones can be moved (they are "mobile") and they use wireless connection to static stations called BTS (Base Transceiver Station). Each BTS covers an area around it and that area is called a *cell*.

The Czech Technical University runs an experimental private GSM network with a BTS right on top of the building you are in just now. Since the placement of base stations is very important for the network coverage, your task is to create a program that will find the optimal position for a BTS. The program will be given coordinates of "points of interest". The goal is to find a position that will cover the maximal number of these points. It is supposed that a BTS can cover all points that are no further than some given distance R. Therefore, the cell has a circular shape.



The picture above shows eight points of interest (little circles) and one of the possible optimal BTS positions (small triangle). For the given distance R, it is not possible to cover more than four points. Notice that the BTS does not need to be placed in an existing point of interest.

Input Specification

The input consists of several scenarios. Each scenario begins with a line containing two integer numbers N and R. N is the number of points of interest, $1 \leq N \leq 2\,000$. R is the maximal distance the BTS is able to cover, $0 \leq R < 10\,000$. Then there are N lines, each containing two integer numbers X_i, Y_i giving coordinates of the i-th point, $|X_i|, |Y_i| < 10\,000$. All points are distinct, i.e., no two of them will have the same coordinates.

The scenario is followed by one empty line and then the next scenario begins. The last one is followed by a line containing two zeros.

A point lying at the circle boundary (exactly in the distance R) is considered covered. To avoid floating-point inaccuracies, the input points will be selected in such a way that for any possible subset of points S that can be covered by a circle with the radius $R + 0.001$, there will always exist a circle with the radius R that also covers them.

Output Specification

For each scenario, print one line containing the sentence "`It is possible to cover` M `points.`", where M is the maximal number of points of interest that may be covered by a single BTS.

Sample Input

```
8 2
1 2
5 3
5 4
1 4
8 2
4 5
7 5
3 3

2 100
0 100
0 -100

0 0
```

Output for Sample Input

```
It is possible to cover 4 points.
It is possible to cover 2 points.
```

The first sample input scenario corresponds to the picture, providing that the X axis aims right and Y axis down.

Key Task

The Czech Technical University is rather old — you already know that it celebrates 300 years of its existence in 2007. Some of the university buildings are old as well. And the navigation in old buildings can sometimes be a little bit tricky, because of strange long corridors that fork and join at absolutely unexpected places.

The result is that some first-graders have often difficulties finding the right way to their classes. Therefore, the Student Union has developed a computer game to help the students to practice their orientation skills. The goal of the game is to find the way out of a labyrinth. Your task is to write a verification software that solves this game.

The labyrinth is a 2-dimensional grid of squares, each square is either free or filled with a wall. Some of the free squares may contain doors or keys. There are four different types of keys and doors: blue, yellow, red, and green. Each key can open only doors of the same color.

You can move between adjacent free squares vertically or horizontally, diagonal movement is not allowed. You may not go across walls and you cannot leave the labyrinth area. If a square contains a door, you may go there only if you have stepped on a square with an appropriate key before.

Input Specification

The input consists of several maps. Each map begins with a line containing two integer numbers R and C ($1 \leq R, C \leq 100$) specifying the map size. Then there are R lines each containing C characters. Each character is one of the following:

Character		**Meaning**
Hash mark	`#`	Wall
Dot	`.`	Free square
Asterisk	`*`	Your position
Uppercase letter	`B Y R G`	Blue, yellow, red, or green door
Lowercase letter	`b y r g`	Blue, yellow, red, or green key
Uppercase X	`X`	Exit

Note that it is allowed to have

- more than one exit,
- no exit at all,
- more doors and/or keys of the same color, and
- keys without corresponding doors and vice versa.

You may assume that the marker of your position ("`*`") will appear exactly once in every map.

There is one blank line after each map. The input is terminated by two zeros in place of the map size.

Output Specification

For each map, print one line containing the sentence "`Escape possible in` S `steps.`", where S is the smallest possible number of step to reach any of the exits. If no exit can be reached, output the string "`The poor student is trapped!`" instead.

One step is defined as a movement between two adjacent cells. Grabbing a key or unlocking a door does not count as a step.

Sample Input

```
1 10
*........X

1 3
*#X

3 20
####################
#XY.gBr.*.Rb.G.GG.y#
####################

0 0
```

Output for Sample Input

```
Escape possible in 9 steps.
The poor student is trapped!
Escape possible in 45 steps.
```

Weird Numbers

Binary numbers form the principal basis of computer science. Most of you have heard of other systems, such as ternary, octal, or hexadecimal. You probably know how to use these systems and how to convert numbers between them. But did you know that the system base (radix) could also be negative? One assistant professor at the Czech Technical University has recently met *negabinary* numbers and other systems with a negative base. Will you help him to convert numbers to and from these systems?

A number N written in the system with a positive base R will always appear as a string of digits between 0 and $R-1$, inclusive. A digit at the position P (positions are counted from right to left and starting with zero) represents a value of R^P. This means the value of the digit is multiplied by R^P and values of all positions are summed together. For example, if we use the octal system (radix $R=8$), a number written as 17024 has the following value:

$$1.8^4 + 7.8^3 + 0.8^2 + 2.8^1 + 4.8^0 = 1.4096 + 7.512 + 2.8 + 4.1 = 7700$$

With a negative radix $-R$, the principle remains the same: each digit will have a value of $(-R)^P$. For example, a negaoctal (radix $R=-8$) number 17024 counts as:

$$1.(-8)^4 + 7.(-8)^3 + 0.(-8)^2 + 2.(-8)^1 + 4.(-8)^0 = 1.4096 - 7.512 - 2.8 + 4.1 = 500$$

One big advantage of systems with a negative base is that we do not need a minus sign to express negative numbers. A couple of examples for the negabinary system ($R=-2$):

decimal	negabinary	decimal	negabinary	decimal	negabinary
-10	1010	-3	1101	4	100
-9	1011	-2	10	5	101
-8	1000	-1	11	6	11010
-7	1001	0	0	7	11011
-6	1110	1	1	8	11000
-5	1111	2	110	9	11001
-4	1100	3	111	10	11110

You may notice that the negabinary representation of any integer number is unique, if no "leading zeros" are allowed. The only number that can start with the digit "0", is the zero itself.

Input Specification

The input will contain several conversions, each of them specified on one line. A conversion from the decimal system to some negative-base system will start with a lowercase word "`to`" followed by a minus sign (with no space before it), the requested base (radix) R, one space, and a decimal number N.

A conversion to the decimal system will start with a lowercase word "`from`", followed by a minus sign, radix R, one space, and a number written in the system with a base of $-R$.

The input will be terminated by a line containing a lowercase word "`end`". All numbers will satisfy the following conditions: $2 \leq R \leq 10$, $-1\,000\,000 \leq N \leq 1\,000\,000$ (decimal).

Output Specification

For each conversion, print one number on a separate line. If the input used a decimal format, output the same number written in the system with a base $-R$. If the input contained such a number, output its decimal value.

Both input and output numbers must not contain any leading zeros. The minus sign "`-`" may only be present with negative numbers written in the decimal system. Any non-negative number or a number written in a negative-base system must not start with it.

Sample Input

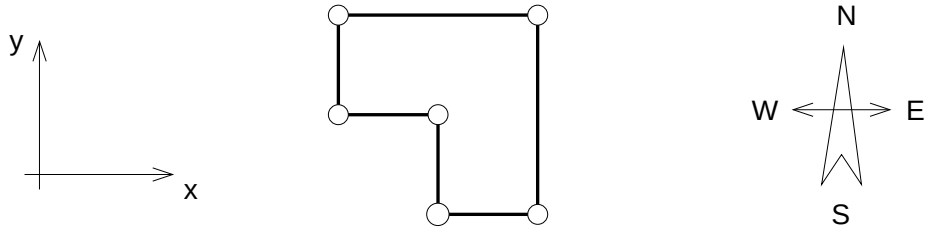
```
to-2 10
from-2 1010
to-10 10
to-10 -10
from-10 10
end
```

Output for Sample Input

```
11110
-10
190
10
-10
```

Rectangular Polygons

In this problem, we will help the Faculty of Civil Engineering. They need a software to analyze ground plans of buildings. Specifically, your task is to detect outlines of a building when all of its corners are given.



You may assume that each building is a rectangular polygon with each of its sides being parallel either with X or Y axis. Therefore, each of its vertex angles is exactly either 90 or 270 degrees.

Input Specification

The input contains several buildings. The description of each building starts with a single positive integer N, the number of corners (polygon vertices), $1 \leq N \leq 1000$. Then there are N pairs of integer numbers X_i, Y_i giving coordinates of individual corners, $|X_i|, |Y_i| \leq 10\,000$.

You may assume that all corners are listed and no two of them have the same coordinates. The polygon does always exist, it is closed, its sides do not intersect or touch (except neighboring sides, of course), and it contains no "holes" inside. In other words, the outline is formed by one closed line. The order of corners in the input file may be arbitrary.

There is an empty line after each building, then the next one is described. After the last building, there is a single zero that signals the end of input.

Output Specification

For each building, output one line containing N characters without any whitespace between them. The characters should be uppercase letters that specify directions of individual walls (sides) when the building outline is followed. "`N`" stands for North (the positive direction of the Y axis), "`E`" for East (the positive direction of the X axis), "`W`" for West, and "`S`" for South. The "walk" should start in the vertex that has been given first in the input and always proceed in the *clockwise* direction.

Sample Input

```
4
0 0
2 2
0 2
2 0

6
1 1
2 2
0 1
1 0
0 2
2 0

0
```

Output for Sample Input

```
NESW
WNESWN
```

The second sample input corresponds to the picture.

Minimizing Maximizer

The company Chris Ltd. is preparing a new sorting hardware called Maximizer. Maximizer has n inputs numbered from 1 to n. Each input represents one integer. Maximizer has one output which represents the maximum value present on Maximizer's inputs.

Maximizer is implemented as a pipeline of sorters $Sorter(i_1, j_1)$, ..., $Sorter(i_k, j_k)$. Each sorter has n inputs and n outputs. $Sorter(i, j)$ sorts values on inputs $i, i+1, \ldots, j$ in non-decreasing order and lets the other inputs pass through unchanged. The n-th output of the last sorter is the output of the Maximizer.

An intern (a former ACM contestant) observed that some sorters could be excluded from the pipeline and Maximizer would still produce the correct result. What is the length of the shortest subsequence of the given sequence of sorters in the pipeline still producing correct results for all possible combinations of input values?

Task

Write a program that:

- reads the description of a Maximizer, i.e. the initial sequence of sorters in the pipeline,
- computes the length of the shortest subsequence of the initial sequence of sorters still producing correct results for all possible input data,
- writes the result.

Input

The first line of the input contains two integers n and m ($2 \leq n \leq 50\,000$, $1 \leq m \leq 500\,000$) separated by a single space. Integer n is the number of inputs and integer m is the number of sorters in the pipeline. The initial sequence of sorters is described in the next m lines. The k-th of these lines contains the parameters of the k-th sorter: two integers i_k and j_k ($1 \leq i_k < j_k \leq n$) separated by a single space.

Output

The output consists of only one line containing an integer equal to the length of the shortest subsequence of the initial sequence of sorters still producing correct results for all possible data.

Example

For the input:

```
40 6
20 30
1 10
10 20
20 30
15 25
30 40
```

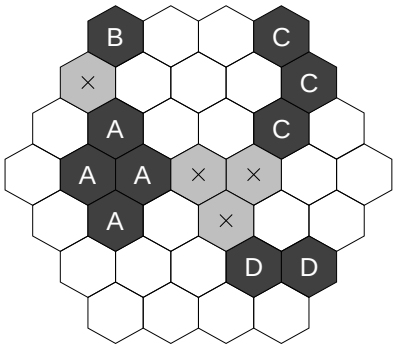
the correct answer is:

```
4
```

Hexagonal Parcels

A civil engineer that has recently graduated from the Czech Technical University encountered an interesting problem and asked us for a help. The problem is more of economical than engineering nature. The engineer needs to connect several buildings with an infrastructure. Unfortunately, the investor is not the owner of all the land between these places. Therefore, some properties have to be bought first.

The land is divided into a regular "grid" of hexagonal parcels, each of them forms an independent unit and has the same value. Some of the parcels belong to the investor. These parcels form four connected areas, each containing one building to be connected with the others. Your task is to find the minimal number of parcels that must be acquired to connect the four given areas.



The whole land also has a hexagonal shape with six sides, each consisting of exactly H parcels. The above picture shows a land with $H = 4$, parcels with letters represent the four areas to be connected. In this case, it is necessary to buy four additional parcels. One of the possible solutions is marked by crosses.

Input Specification

The input contains several scenarios. Each scenario begins with an integer number H, which specifies the size of the land, $2 \leq H \leq 20$. Then there are $2.H - 1$ lines representing individual "rows" of the land (always oriented as in the picture). The lines contain one non-space character for each parcel. It means the first line will contain H characters, the second line $H + 1$, and so on. The longest line will be the middle one, with $2.H - 1$ characters. Then the "length" descends and the last line contains H parcels, again.

The character representing a parcel will be either a dot ("`.`") for the land that is not owned by the investor, or one of the uppercase letters "`A`", "`B`", "`C`", or "`D`". The areas of parcels occupied by the same letter will always be connected. It means that between any two parcels in the same area, there exists a path leading only through that area.

Beside the characters representing parcels, the lines may contain any number of spaces at any positions to improve "human readability" of the input. There is always at least one space between two letters (or the dots). After the land description, there will be one empty line and then the next scenario begins. The last scenario is followed by a line containing zero.

Output Specification

For each scenario, output one line with the sentence "`You have to buy` P `parcels.`", where P is the minimal number of parcels that must be acquired to make all four areas connected together.

Areas are considered *connected*, if it is possible to find a path between them that leads only through parcels that have been bought.

Sample Input

```
4
    B . . C
   . . . . C
  . A . . C .
 . A A . . . .
  . A . . . .
   . . . D D
    . . . .

0
```

Output for Sample Input

```
You have to buy 4 parcels.
```

House Cleaning

Maomao decides to clean her house (which hasn't been cleaned for almost 20 years!) with water in the MRR River.

The rooms in the house are in precise grid pattern, with a door between every two neighboring rooms. Once water runs into one room, it will flush into neighboring room if the door between them is open. Maomao plans to open some of the doors so that all the rooms will be flushed. The only problem is that, rooms have different heights.

Doors can be opened in both directions. However, the energy Maomao should use is the height of the room that the door pushed forward to. Maomao wants to know how much energy she needs to complete the task.

Input

The first line contains two integers: the length l and width w of the numbers of the rooms ($3 \leq w, h \leq 40$). The next w lines describe the height of each room. The heights are in the range of 1 to 100.

Output

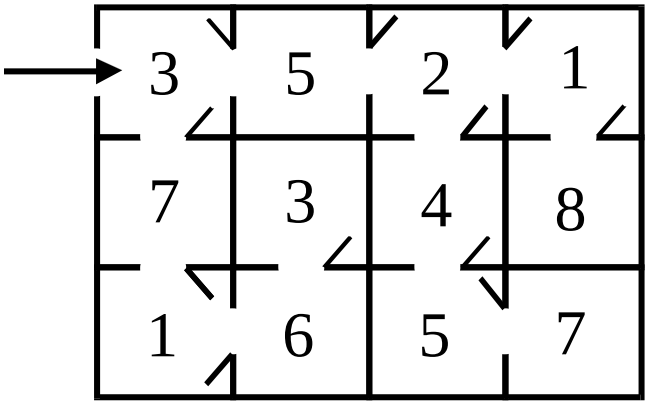
The minimum energy Maomao needs to use to make all rooms to be flushed. The water goes in the house in the room in the top left corner.

Sample Input

```
4 3
3 5 2 1
7 3 4 8
1 6 5 7
```

Sample Output

```
26
```



Notes:

See the picture on the right as an illustration of the sample.