539 The Settlers of Catan

Within *Settlers of Catan*, the 1995 German game of the year, players attempt to dominate an island by building roads, settlements and cities across its uncharted wilderness.

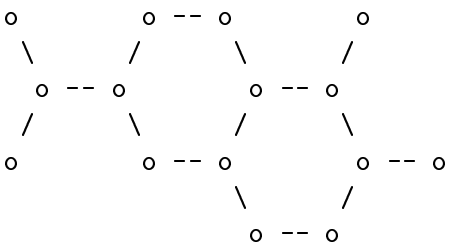
You are employed by a software company that just has decided to develop a computer version of this game, and you are chosen to implement one of the game's special rules:

When the game ends, the player who built the longest road gains two extra victory points.

The problem here is that the players usually build complex road networks and not just one linear path. Therefore, determining the longest road is not trivial (although human players usually see it immediately).

Compared to the original game, we will solve a simplified problem here: You are given a set of nodes (cities) and a set of edges (road segments) of length 1 connecting the nodes. The longest road is defined as the longest path within the network that doesn't use an edge twice. Nodes may be visited more than once, though.

Example: The following network contains a road of length 12.



Input

The input file will contain one or more test cases.

The first line of each test case contains two integers: the number of nodes n ($2 \leq n \leq 25$) and the number of edges m ($1 \leq m \leq 25$). The next m lines describe the m edges. Each edge is given by the numbers of the two nodes connected by it. Nodes are numbered from 0 to $n-1$. Edges are undirected. Nodes have degrees of three or less. The network is not neccessarily connected.

Input will be terminated by two values of 0 for n and m.

Output

For each test case, print the length of the longest road on a single line.

Sample Input

```
3 2
0 1
1 2
15 16
0 2
1 2
2 3
3 4
```

```
3 5
4 6
5 7
6 8
7 8
7 9
8 10
9 11
10 12
11 12
10 13
12 14
0 0
```

Sample Output

```
2
12
```

183 Bit Maps

The bitmap is a data structure that arises in many areas of computing. In the area of graphics, for example, a bitmap can represent an image by having a 1 represent a black pixel and a 0 represent a white pixel.

Consider the following two ways of representing a rectangular bit map. In the first, it is simply represented as a two dimensional array of `1`s and `0`s. The second is based on a decomposition technique. First, the entire bit map is considered. If all bits within it are 1, a `1` is output. If all bits within it are 0, a `0` is output. Otherwise, a `D` is output, the bit map is divided into quarters (as described below), and each of those is processed in the same way as the original bit map. The quarters are processed in top left, top right, bottom left, bottom right order. Where a bit map being divided has an even number of rows and an even number of columns, all quarters have the same dimensions. Where the number of columns is odd, the left quarters have one more column than the right. Where the number of rows is odd the top quarters have one more row than the bottom. Note that if a region having only one row or one column is divided then two halves result, with the top half processed before the bottom where a single column is divided, and the left half before the right if a single row is divided.

Write a program that will read in bitmaps of either form and transform them to the other form.

Input

Input will consist of a series of bit maps. Each bit map begins with a line giving its format ("`B`" or "`D`") and its dimensions (rows and columns). Neither dimension will be greater than 200. There will be at least one space between each of the items of information. Following this line will be one or more lines containing the sequence of "`1`", "`0`" and "`D`" characters that represent the bit map, with no intervening spaces. Each line (except the last, which may be shorter) will contain 50 characters. A "`B`" type bitmap will be written left to right, top to bottom. The file will be terminated by a line consisting of a single `#`.

Output

Output will consist of a series of bitmaps, one for each bit map of the input. Output of each bit map begins on a new line and will be in the same format as the input. The width and height are to be output right justified in fields of width four.

Sample input

```
B 3  4
001000011011
D  2   3
DD10111
#
```

Sample output

```
D   3   4
D0D1001D101
B   2   3
101111
```

Problem F
Solve It

Input: standard input
Output: standard output
Time Limit: 1 second
Memory Limit: 32 MB

Solve the equation:

$p*e^{-x} + q*\sin(x) + r*\cos(x) + s*\tan(x) + t*x^2 + u = 0$

where $0 <= x <= 1$.

Input

Input consists of multiple test cases and terminated by an EOF. Each test case consists of 6 integers in a single line: p, q, r, s, t and u (where $0 <= p,r <= 20$ and $-20 <= q,s,t <= 0$). There will be maximum 2100 lines in the input file.

Output

For each set of input, there should be a line containing the value of x, correct upto 4 decimal places, or the string "`No solution`", whichever is applicable.

Sample Input

```
0 0 0 0 -2 1
1 0 0 0 -1 2
1 -1 1 -1 -1 1
```

Sample Output

```
0.7071
No solution
0.7554
```

Mustaq Ahmed

679 Dropping Balls

A number of K balls are dropped one by one from the root of a fully binary tree structure FBT. Each time the ball being dropped first visits a non-terminal node. It then keeps moving down, either follows the path of the left subtree, or follows the path of the right subtree, until it stops at one of the leaf nodes of FBT. To determine a ball's moving direction a flag is set up in every non-terminal node with two values, either **false** or **true**. Initially, all of the flags are **false**. When visiting a non-terminal node if the flag's current value at this node is **false**, then the ball will first switch this flag's value, i.e., from the **false** to the **true**, and then follow the left subtree of this node to keep moving down. Otherwise, it will also switch this flag's value, i.e., from the **true** to the **false**, but will follow the right subtree of this node to keep moving down. Furthermore, all nodes of FBT are sequentially numbered, starting at 1 with nodes on depth 1, and then those on depth 2, and so on. Nodes on any depth are numbered from left to right.

For example, Fig. 1 represents a fully binary tree of maximum depth 4 with the node numbers 1, 2, 3, ..., 15. Since all of the flags are initially set to be **false**, the first ball being dropped will switch flag's values at node 1, node 2, and node 4 before it finally stops at position 8. The second ball being dropped will switch flag's values at node 1, node 3, and node 6, and stop at position 12. Obviously, the third ball being dropped will switch flag's values at node 1, node 2, and node 5 before it stops at position 10.

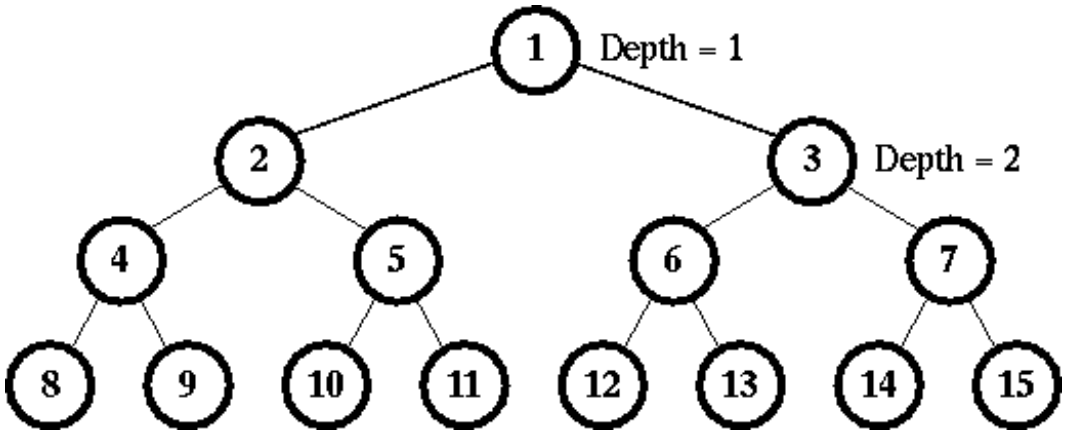


Fig. 1: An example of FBT with the maximum depth 4 and sequential node numbers.

Now consider a number of test cases where two values will be given for each test. The first value is D, the maximum depth of FBT, and the second one is I, the I^{th} ball being dropped. You may assume the value of I will not exceed the total number of leaf nodes for the given FBT.

Please write a program to determine the stop position P for each test case.

For each test cases the range of two parameters D and I is as below:

$$2 \leq D \leq 20, \text{ and } 1 \leq I \leq 524288.$$

Input

Contains $l + 2$ lines.

Line 1	I	the number of test cases
Line 2	D_1 I_1	test case #1, two decimal numbers that are separated by one blank
...		
Line $k + 1$	D_k I_k	test case #k
Line $l + 1$	D_l I_l	test case #l
Line $l + 2$	`-1`	a constant −1 representing the end of the input file

Output

Contains l lines.

Line 1	the stop position P for the test case #1
...	
Line k	the stop position P for the test case #k
...	
Line l	the stop position P for the test case #l

Sample Input

```
5
4 2
3 4
10 1
2 2
8 128
-1
```

Sample Output

```
12
7
512
3
255
```

I	N + NOD (N)	
	Input	Standard Input
	Output	Standard Output

Consider an integer sequence **N** where,

$\mathbf{N_0 = 1}$

$\mathbf{N_i = N_{i-1} + NOD(N_{i-1})}$ **- for i > 0**

Here, NOD(**x**) = number of divisors of **x**.

So the first few terms of this sequence are *1 2 4 7 9 12 18…*

Given two integers **A** and B, find out the number of integers in the above sequence that lies within the range [**A, B**].

Input

The first line of input is an integer **`T (T < 100000),`** that indicates the number of test cases. Each case contains two integers, **`A`** followed by **`B (1 ≤ A ≤ B ≤ 1000000)`**.

Output

For each case, output the case number first followed by the required result.

Sample Input	Sample Output
3 1 18 1 100 3000 4000	Case 1: 7 Case 2: 20 Case 3: 87

Problemsetter: Sohel Hafiz, Special Thanks: Shamim Hafiz

Problem J
The Closest Pair Problem

Input: standard input
Output: standard output
Time Limit: 8 seconds
Memory Limit: 32 MB

Given a set of points in a two dimensional space, you will have to find the distance between the closest two points.

Input

The input file contains several sets of input. Each set of input starts with an integer **N (0<=N<=10000)**, which denotes the number of points in this set. The next **N** line contains the coordinates of **N** two-dimensional points. The first of the two numbers denotes the **X-coordinate** and the latter denotes the **Y-coordinate**. The input is terminated by a set whose **N=0**. This set should not be processed. The value of the coordinates will be less than **40000** and non-negative.

Output

For each set of input produce a single line of output containing a floating point number (with four digits after the decimal point) which denotes the distance between the closest two points. If there is no such two points in the input whose distance is less than **10000**, print the line **INFINITY**.

Sample Input

```
3
0 0
10000 10000
20000 20000
5
0 2
6 67
43 71
39 107
189 140
0
```

Sample Output

```
INFINITY
36.2215
```

(World Final Warm-up Contest, Problem setter: Shahriar Manzoor)

“Generally, a brute force method has only two kinds of reply, a) Accepted b) Time Limit Exceeded.”