Problem 1. Balanced Parentheses

When programming, we use parentheses `()`, brackets `[]`, and curly braces `{}` a lot. Sometimes, we forget a parenthesis or add an extra one by mistake. Many modern coding environments will automatically check to see if a set of these characters are "balanced" to save time in debugging. A set of parentheses, brackets, and braces is called "balanced" if for every open parenthesis `(`, there is a matching closing parenthesis `)` paired with it and likewise for brackets and braces. Everything between the paired characters must also recursively be a balanced string. For example, these sets of parentheses, brackets, and braces are balanced:

```
([{}])      (){}[()]      (((()()())(())))
```

While these are not:

```
(())(      [(])      {}()[()]}
```

Your task is to write a program that will check whether or not a given set of parentheses is balanced.

Input

The input consists of a number n, with $1 \leq n \leq 1000$, followed by n lines. Each line will contain a string containing only the characters "`()[]{}`". The length of each string will not exceed 1000 characters.

Output

Write n lines. On each line, write "Balanced" if the corresponding set of parentheses, brackets, and braces is balanced and "Not Balanced" if it is not.

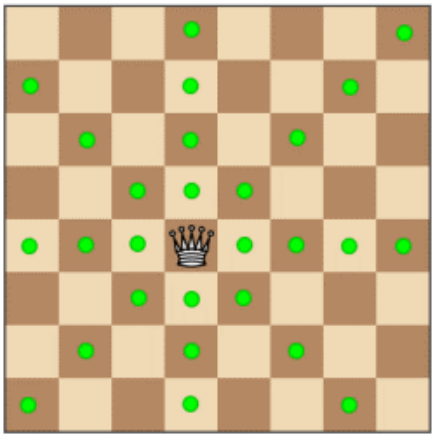
Example

standard input	standard output
6 ([{}]) (){}[()] (((()()())(()))) (())([(]) {}()[()]}	Balanced Balanced Balanced Not Balanced Not Balanced Not Balanced

Problem 2. Too Many Queens in the Kitchen

The n-queens problem is a chess puzzle where the player must try to place n queens onto an $n \times n$ chessboard without any two queens "attacking" each other. There are solutions for all values of n.

In chess, a queen is piece that can move any number of squares vertically, horizontally, or diagonally. In the image below, the queen can move to any square with a green dot.



Source: learnchessrules.com

A queen is "attacking" another piece if it can immediately move to the square the other piece is currently occupying. Your task, should you choose to accept it, is to check, given an $n \times n$ board with k queens placed on it, whether or not any of the specified queens attack each other.

Input

The input begins with a line containing a pair of numbers n and k, where n is the size of the board, and k is the number of queens. Here $1 \leq n \leq 1000$ and $1 \leq k \leq n$. The following k lines contain a pair of numbers (i, j) with $1 \leq i \leq n$ and $1 \leq j \leq n$. Each (i, j) represents a queen that is placed on the ith column (from the left) and jth row (from the bottom). No two of the specified queens will occupy the same square. See the samples below.

Output

Print one line one line consisting of the word "Valid" if no queens are attacking each other, and "Invalid" otherwise.

Examples

standard input	standard output
4 4 1 2 2 4 3 1 4 3	Valid
4 2 1 1 4 4	Invalid

Problem 3. SUDOKU

Sudoku is a popular newspaper puzzle consisting of a 9×9 grid. The grid is divided into nine 3×3 "blocks". Each cell in the grid contains either a blank spot or a digit from 1-9. The goal is to fill in the blank spots so that each row, column, and block contain one of each digit. An example puzzle and solution is presented below.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Source: Wikimedia Commons

Your task is to write a program that will solve any given sudoku puzzle.

Input

The input will always consist of nine lines, each representing a row of the sudoku puzzle. Each line will contain nine integers, each between 0 and 9 (inclusive). The 0s represent the blank cells in the puzzle. It is guaranteed that there is a way to complete this partially filled grid satisfying the constraints. Unlike the puzzles you find in the newspaper, the puzzles here are randomly generated, and may have more than one solution.

Output

Output nine lines, with each line containing the corresponding row of the solution to the puzzle and each cell separated by a space. If multiple solutions exist, you should output the one which fills the first blank with a smaller number. If the first blank is the same number for both, then you should choose the solution which fills the second blank with a smaller number and so on. Note: The "first" blank refers to the topmost and leftmost blank. Thus, in the example above, the 4 is in the first blank, the 6 in the second, the 8 in the third, and so on.

Example

standard input	standard output
0 0 0 0 0 7 0 0 3	1 2 4 5 6 7 8 9 3
0 0 0 0 0 0 0 0 0	3 5 6 1 8 9 2 7 4
0 0 0 0 0 0 0 0 0	7 8 9 2 3 4 5 1 6
0 0 0 0 0 0 1 4 0	2 3 5 6 7 8 1 4 9
0 0 0 0 0 1 7 0 0	4 6 8 9 2 1 7 3 5
0 1 0 0 4 0 0 0 0	9 1 7 3 4 5 6 2 8
0 0 0 0 0 0 3 0 0	5 4 1 7 9 6 3 8 2
0 0 0 0 5 2 9 0 0	8 7 3 4 5 2 9 6 1
0 0 0 0 0 0 0 0 0	6 9 2 8 1 3 4 5 7

Problem 4. Use the Force, Lu

Steven is training to become a jedi while riding on the ship the Mellon-ian Tartan. He is currently using a training remote, which sends low-power lasers in different directions, which Steven must then deflect with his lightsaber. However, the force is not strong with this one, and Steven is not doing very well. Thus, he decides to use his programmable robotic arm to help him out. He needs you to write a program for his arm that will help him decide the direction to move his lightsaber.

Input

The input consists of a number n, with $1 \leq n \leq 1000$, followed by n lines. Each line will contain six integers whose absolute value is at most 1000, in the order x_1, y_1, x_2, y_2, x_3, y_3. The first four numbers represent the lightsaber, a line with equation $(y - y_1) * (x_2 - x_1) = (x - x_1) * (y_2 - y_1)$. (For the purposes of this problem, the lightsaber is an infinite line in both directions.) The last two numbers represent the point (x_3, y_3) where the next laser will be. (x_1, y_1) and (x_2, y_2) will never be the same point. Study the sample input and output carefully to make sure you understand the problem.

Output

Write n lines. On each line, write "Right" if (x_3, y_3) is to the right of the line, "Left" if (x_3, y_3) is to the left of the line, or "Stay" if (x_3, y_3) is on the line.

Example

standard input	standard output
5 0 1 1 3 2 5 0 1 1 3 -1 -1 0 0 1 1 5 0 1 1 0 0 5 0 0 0 0 1 3 4	Stay Stay Right Left Right

Note

Lines with the same equation can be "facing different directions" if the order of the points is different. For example, the point (1,1) is on the right side of the line with $(x_1, y_1) = (0, 0)$ and $(x_2, y_2) = (0, 1)$ but on the left side of the line with $(x_1, y_1) = (0, 1)$ and $(x_2, y_2) = (0, 0)$

Problem 5. Repeating Decimals

Recall that when a decimal repeats, we write it with a bar over the repeating digits. For example, $\frac{1}{11}$ is written as $1.\overline{09}$ and $\frac{1}{6}$ is written as $1.1\overline{6}$. We call the portion of the decimal under the bar the *repetend*. So, in the previous examples, the repetend for $\frac{1}{11}$ is 09 while the repetend for $\frac{1}{6}$ is 6.

Danny is doing research on repeated decimals, and wants to know what the repetend is for any given rational number. However, calculating these has become very tedious, so he needs your help. He has asked you to write a program to find the repeating portion of a decimal.

Input

The input consists of a number k, with $1 \leq k \leq 1000$, followed by k lines. Each line will contain two integers, n and d, which represent the numerator and denominator of the fraction. These numbers are in the range $[1, 1000]$, that is, they satisfy $1 \leq n, d \leq 1000$.

Output

Write k lines. On each line, write the repeating portion of the decimal form of $\frac{n}{d}$. If the decimal terminates, write "Terminates".

Example

standard input	standard output
4 1 6 4 9 1 7 1 25	6 4 142857 Terminates

Problem 6. Palindrome Parsing

Mark is doing research on string parsing algorithms. Currently, he is interested in palindromes. A palindrome is a string which is spelled the same forwards and back. For example, "racecar" is a palindrome. Mark wants to figure out how many palindromes a string can be broken into. For instance, the string "HELLO" can be broken into the palindromes "H", "E", "LL", and "O". Your task is to find, given a string, the smallest number of palindromes it can be broken into.

Input

The input will start with a single number, n, with $1 \leq n \leq 1000$, followed by n lines. Each line will consist of a string containing only lower and upper case letters and numbers, with no whitespace. The total length of all the strings is at most 5000.

Output

Output n lines, each containing the smallest number of substrings that the corresponding input string can be broken into such that each substring is a palindrome.

Example

standard input	standard output
3 HELLO abcdefg racecar11	4 7 2

Problem 7. Highest Density Square

You're given n (not necessarily distinct) points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, along with a side length s of a square. All these values are integers. What's the maximum number of the given points that it is possible for an axis-aligned square of side-length s to contain? This is what you have to compute. (A square includes all the points on its boundary.)

Input

The first line contains two space-separated integers s and n. $0 \leq s \leq 10^6$, $1 \leq n \leq 5 \times 10^5$. The following n lines each contain a pair of integers, which are the x and y coordinates of one of the points. $0 \leq x, y \leq 10^6$.

Output

Output a single integer: The maximum number of points from the input that it is possible for an axis-aligned square of size s to contain.

Examples

standard input	standard output
2 7 0 0 2 0 4 0 1 1 0 2 2 2 4 2	5
3 5 5 5 5 5 5 6 8 4 10 10	4