Problem A: Bridges and Tunnels

Some days, the university campus gets very wet. Other days, it can get very cold. Although many times, it is pleasant to be outside, there are days when we would prefer to stay indoors.

Luckily, the campus designers have gradually connected the various buildings with tunnels and bridges. Sometimes these connections are built when the building is built, but it may also be possible to add them afterwards. Unfortunately, sometimes it is not possible to travel between two buildings without going outside because the system of bridges and tunnels is not complete. You would like to write a program to help a newcomer determine how many buildings can be reached from a newly constructed tunnel.

Input Specification

The first line of input contains one integer specifying the number of test cases to follow. Each test case begins with a line containing an integer *n*, the total number of bridges or tunnels built. This number will be no more than 100,000. We assume that all of the buildings have been built ahead of time, but at the beginning of a test case, no bridges or tunnels have been built yet. The rest of the test case consists of*n* lines giving the history of the bridges or tunnels, in the order in which they are built. Each of these lines contains the names of the two buildings being connected, separated by a space. Each building name is a sequence of up to 20 uppercase or lowercase letters, and building names are case-sensitive.

Sample Input

```
1
3
MC DC
DC Eng
MC MThree
```

Output Specification

Whenever a bridge or tunnel is built, print a line containing one integer, the number of buildings that can be reached from that bridge without going outside.

Output for Sample Input

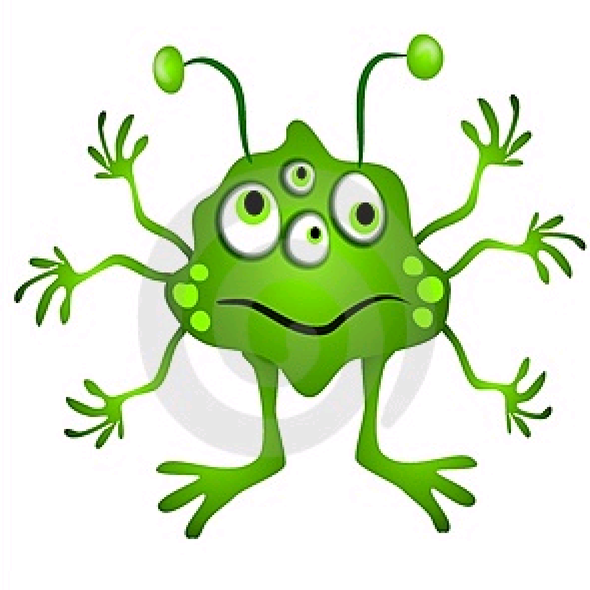
```
2
3
4
```

Ondřej Lhoták

Problem B: Alien Communicating Machines

Aliens on the planet Hex have sixteen fingers. As a result, they have developed a system of writing numbers with sixteen digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, in ascending order). Like the decimal system that we use, a number is written as a sequence of digits, and the value of a number is determined by the following rules:

- The value of a single-digit number is just the value of that digit.
- The value of number *a* consisting of a number *b* followed by a digit *d* is the value of *b* multiplied by a fixed base, plus the value of *d*. In our decimal system, the base is ten, whereas in the system used by the Hex aliens, the base is sixteen.



Still other planets have other aliens with different numbers of fingers. These aliens use the same rules for writing numbers, but each uses a different base.

You have the important job of promoting universal peace by improving communication among all of these aliens. Specifically, you are to write a program that translates numbers written by one alien into numbers that can be understood by a different alien.

Input Specification

The first line of input contains a single integer, the number of test cases to follow. Each test case is a single line containing three numbers *x*, *y*, and *z*. Both *x* and *y* are written in decimal, and are the bases used by the two aliens. Each is at least two and at most thirty-six. The number *z* is written in base *x*, using the letters A through Z as the digits with value 10 through 35. The number *z* will be no greater than four billion.

Sample Input

```
1
16 10 11
```

Output Specification

For each line of input, output a line containing a single number, equal in value to *z*, but written in base *y*.

Output for Sample Input

```
17
```

Ondřej Lhoták

Problem C: Geometric Sequence

A sequence of integers is called a geometric sequence if the ratio of consecutive numbers is constant. For example, (3,6,12,24) is a geometric sequence (each term is equal to twice the previous number).

Now, with such a sequence, we will shuffle it and remove some of the elements.
Given the result of such a transformation, try to recover the "geometric ratio" of the original sequence, i.e. the ratio of consecutive numbers.

If there are multiple possible ratios, output the one with the greatest absolute value. If there is still a tie, output the positive one.
If there is no such sequence, output 0.

Input Specification

The first line of input contains a single integer, $2 \leq N \leq 100{,}000$, the number of integers in the transformed sequence.
Following this will be N lines, each containing a single integer of the transformed sequence (each element will be less than or equal to 10^{18} in absolute value, and no element will be zero).

Sample Input

```
3
1
3
27
```

Output Specification

The ratio of the original sequence (if one exists). The relative error of the answer must be within 10^{-9}. That is, (abs(answer - expected) / expected < 10^{-9}).

Output for Sample Input

```
3
```

The original sequence could have been (1,3,9,27,...) or (...,27,9,3,1); the former has the greater ratio (3).

Hanson Wang

Problem D: Largest Square

There is an $N \times N$ mosaic of square solar cells ($1 \le N \le 2{,}000$). Each solar cell is either good or bad. There are W ($1 \le W \le 50{,}000$) bad cells. You need to find the largest square within the mosaic containing at most L ($0 \le L \le W$) bad cells.

Input Specification

The input will begin with a number $Z \le 20$, the number of test cases, on a line by itself. Z test cases then follow. The first line of the test case contains three space-separated integers: N, W, and L. W lines follow, each containing two space-separated integers representing the coordinates of a location of the bad solar cells.

Sample Input

```
1
4 3 1
1 1
2 2
2 3
```

Output Specification

For each input instance, the output will be a single integer representing the area of the largest square that contains no more than L bad solar cells.

Output for Sample Input

```
4
```

Explanation of Sample Output

The mosaic is 4×4, and contains the following arrangement of good and bad cells ('G' represents good, and 'B' represents bad):

```
BGGG
GBBG
GGGG
GGGG
```

Several 2×2 squares at the bottom contain no bad solar cells, but all 3×3 squares contain at least two bad solar cells.

Neal Wu

Problem E: Harmonious Matrices

Call an $m \times n$ matrix of bits "harmonious" if every cell in it has an even number of 1 bits as neighbors. A cell is a neighbor of itself, and also to the cells above, below, left, and right (if they exist). So the number of neighbors of a cell is at most five, but could be less, depending on where it is. The following is an harmonious 4×4 square of bits:

```
0 1 0 0
1 1 1 0
0 0 0 1
1 1 0 1
```

The task is to write a program which takes as input *m* and *n*, and produces an harmonious matrix of *m* rows and *n* columns of bits. The solution should avoid the all-zero matrix (if possible).

Input Specification

The input will begin with a number $Z \leq 40$ on a line by itself. This is followed by *Z* lines, each of which contains two space-separated positive integers *m* and *n*, each of which will be at most 40.

Sample Input

```
2
4 4
1 6
```

Output Specification

For each input instance, the output will be an $m \times n$ harmonious matrix of 0s and 1s. The matrix should be non-zero if possible.

Output for Sample Input

```
0 1 0 0
1 1 1 0
0 0 0 1
1 1 0 1
0 0 0 0 0 0
```

Danny Sleator