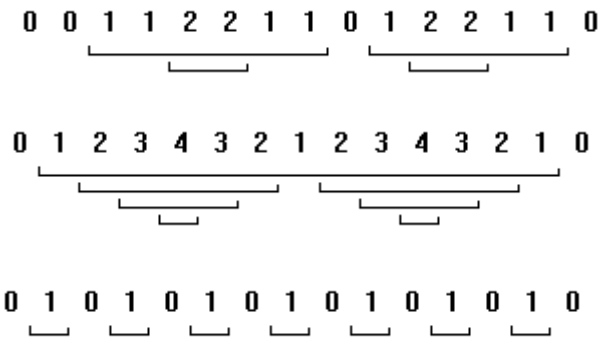
A • Islands in the Data Stream

Given a sequence of integers **a1, a2, a3, …, an**, an *island* in the sequence is a contiguous subsequence for which each element is greater than the elements immediately before and after the subsequence. In the examples below, each island in the sequence has a bracket below it. The bracket for an island contained within another island is below the bracket of the containing island.



Write a program that takes as input a sequence of **15** non-negative integers, in which each integer differs from the previous integer by at most **1**, and outputs the number of islands in the sequence.

Input

The first line of input contains a single integer ***P***, (**1** ≤ ***P*** ≤ **1000**), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input. It contains the data set number, ***K,*** followed by **15** non-negative integers separated by a single space. The first and last integers in the sequence will be 0. Each integer will differ from the previous integer by at most 1.

Output

For each data set there is one line of output. The single output line consists of the data set number, ***K***, followed by a single space followed by the number of islands in the sequence.

Sample Input	Sample Output
4 1 0 0 1 1 2 2 1 1 0 1 2 2 1 1 0 2 0 1 2 3 4 3 2 1 2 3 4 3 2 1 0 3 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 4 0 1 2 3 4 5 6 7 6 5 4 3 2 1 0	1 4 2 7 3 7 4 7

B • Von Neumann's Fly

The following problem was posed to John von Neumann:

> Two bicyclists, ***A*** and ***B***, start riding toward each other at the same time from places that are 250 miles apart, A traveling at 10 miles per hour, and ***B*** at 15 miles per hour. At the same time, a fly leaves the front wheel of ***A***'s bicycle, and flies toward ***B***'s bicycle at 20 miles per hour. As soon as he touches the front wheel of ***B***'s bicycle, he turns around and flies back. As the bicycles approach each other, he continues flying back and forth, touching each front wheel in turn, until, alas, he is crushed between them. Since the fly travels faster than either cyclist, he makes an infinite number of trips, yet travels a finite distance (the infinite series converges). How far did the fly travel?

Von Neumann immediately summed the infinite series (in his head!), and arrived at the correct answer: 200 miles.

You are to write a program that solves a more general version of that problem, with varying initial distances and speeds.

Input

The first line of input contains a single integer ***P***, (**1 ≤ *P* ≤ 1000**), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line containing five values: an integer ***N*** (the data set number), and four floating-point values: ***D*** (the initial distance between the bicycles, **10** ≤ ***D*** ≤ **1000**), ***A*** (cyclist ***A***'s speed in miles per hour, **1** ≤ ***A*** ≤ **30**), ***B*** (cyclist ***B***'s speed in miles per hour, **1** ≤ ***B*** ≤ **30**), and ***F*** (the fly's speed in miles per hour, ***A*** ≤ ***B***< ***F*** ≤ ***50***).

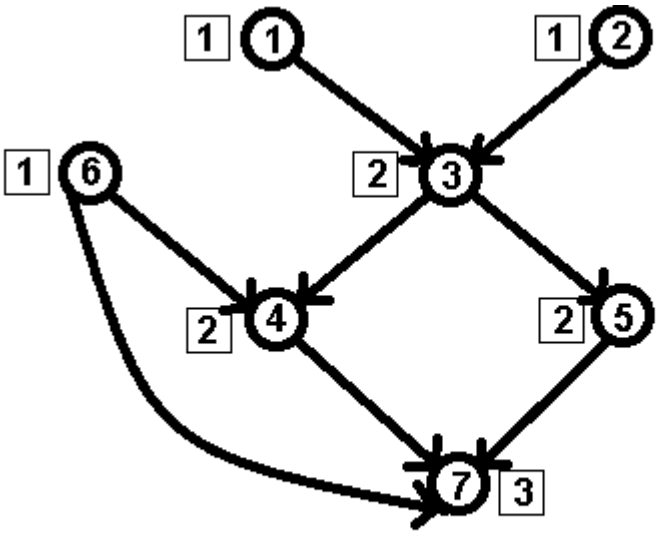
Output

For each data set there is one line of output. It contains the data set number followed by a single space, followed by the number of miles traveled by the fly, (the sum of the infinite series described by the input values), accurate to two decimal places.

Sample Input	Sample Output
5 1 250 10 15 20 2 10.7 3.5 4.7 5.5 3 523.7 15.3 20.7 33.3 4 1000 30 30 50 5 500 15 15 25	1 200.00 2 7.18 3 484.42 4 833.33 5 416.67

C • Strahler Order

In geology, a river system can be represented as a directed graph. Each river segment is an edge; with the edge pointing the same way the water flows. Nodes are either the source of a river segment (for example, a lake or spring), where river segments merge or diverge, or the mouth of the river.



Note: The number in a box is the order. The number in a circle is the node number.

The *Strahler order* of a river system is computed as follows.

- The order of each source node is `1`.
- For every other node, let `i` be the highest order of all its upstream nodes. If just one upstream node has order `i`, then this node also has order `i`. If two or more upstream nodes have order `i`, then this node has order `i+1`.

The order of the entire river system is the order of the mouth node. In this problem, the river system will have just one mouth. In the picture above, the *Strahler* order is three (3).

You must write a program to determine the order of a given river system.

The actual river with the highest order is the *Amazon*, with order **12**. The highest in the U.S. is the *Mississippi*, with order **10**.

Node **M** is the mouth of the river. It has no outgoing edges.

Input

The first line of input contains a single integer K, ($1 \le K \le 1000$), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of multiple lines of input. The first line of each data set contains three positive integers, K, M and P ($2 \le M \le 1000$). K is the data set number. M is the number of nodes in the graph and P is the number of edges. The first line is followed by P lines, each describing an edge of the graph. The line will contain two positive integers, A and B, indicating that water flows from node A to node B ($1 \le A, B \le M$). Node M is the mouth of the river. It has no outgoing edges.

Output

For each data set there is a single line of output. The line consists of the data set number, a single space and the order of the river system.

Sample Input	Sample Output
1 1 7 8 1 3 2 3 6 4 3 4 3 5 6 7 5 7 4 7	1 3

D • Pisano Periods

In 1960, Donald Wall of IBM, in White Plains, NY, proved that the series obtained by taking each element of the *Fibonacci* series modulo `m` was periodic.

For example, the first ten elements of the *Fibonacci* sequence, as well as their remainders modulo `11`, are:

```
n            | 1  2  3  4  5  6   7   8   9  10
F(n)         | 1  1  2  3  5  8  13  21  34  55
F(n) mod 11  | 1  1  2  3  5  8   2  10   1   0
```

The sequence made up of the remainders then repeats. Let `k(m)` be the length of the repeating subsequence; in this example, we see `k(11) = 10`.

Wall proved several other properties, some of which you may find interesting:

- If `m > 2`, `k(m)` is even.
- For any even integer `n > 2`, there exists m such that `k(m) = n`.
- $k(m) \leq m^2 - 1$
- $k(2^n) = 3 * 2^{(n-1)}$
- $k(5^n) = 4 * 5^n$
- $k(2 * 5^n) = 6n$
- If $n > 2$, $k(10^n) = 15 * 10^{(n-1)}$

For this problem, you must write a program that calculates the length of the repeating subsequence, `k(m)`, for different modulo values `m`.

Input

The first line of input contains a single integer ***P***, ($1 \leq P \leq 1000$), which is the number of data sets that follow. Each data set is a single line that consists of two space separated integer values ***N*** and ***M***. ***N*** is the data set number. ***M*** is the modulo value (`2 <= m <= 1,000,000`).

Output

For each data set there is one line of output. It contains the data set number (***N***) followed by a single space, followed by the length of the repeating subsequence for ***M, k(M)***.

Sample Input	Sample Output
5 1 4 2 5 3 11 4 123456 5 987654	1 6 2 20 3 10 4 15456 5 332808

E • Deranged Exams

The first question on the *Data Structures and Algorithms* final exam has a list of **N** terms and a second list of **N** definitions. Students are to match each term with the correct definition. Unfortunately, Joe, who wrote a Visual BASIC program in high school and assumed he knew all there was to know about Computer Science, did not bother to come to class or read the textbook. He has to guess randomly what the matches are. Let **`S(N,k)`** be the number of ways Joe can answer the question and get at least the first **`k`** matches wrong.

For this problem, you will write a program to compute **`S(N,k)`**.

Input

The first line of input contains a single integer ***P***, (**1** ≤ ***P*** ≤ **1000**), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input containing three space separated decimal integers. The first integer is the data set number. The second integer is the number, **`N`** `(1 <=` **`N`** `<= 17)`, of terms to be matched in the question. The third integer is the number, **`k`** `(0 <=` **`k`** `<=` **`N`**`)`, of initial matches to be incorrect.

Output

For each data set there is a single line of output. It contains the data set number followed by a single space which is then followed by the value of **`S(N,k)`**.

Sample Input	Sample Output
4 1 4 1 2 7 3 3 10 5 4 17 17	1 18 2 3216 3 2170680 4 130850092279664

Problem C: Coolest Ski Route

John loves winter. Every skiing season he goes heli-skiing with his friends. To do so, they rent a helicopter that flies them directly to any mountain in the Alps. From there they follow the picturesque slopes through the untouched snow.

Of course they want to ski on only the best snow, in the best weather they can get. For this they use a combined condition measure and for any given day, they rate all the available slopes.

Can you help them find the most awesome route?

Input

The input consists of:

- one line with two integers n ($2 \leq n \leq 1000$) and m ($1 \leq m \leq 5000$), where n is the number of (1-indexed) connecting points between slopes and m is the number of slopes.
- m lines, each with three integers s, t, c ($1 \leq s, t \leq n$, $1 \leq c \leq 100$) representing a slope from point s to point t with condition measure c.

Points without incoming slopes are mountain tops with beautiful scenery, points without outgoing slopes are valleys. The helicopter can land on every connecting point, so the friends can start and end their tour at any point they want. All slopes go downhill, so regardless of where they start, they cannot reach the same point again after taking any of the slopes.

Output

Output a single number n that is the maximum sum of condition measures along a path that the friends could take.

Sample visualization

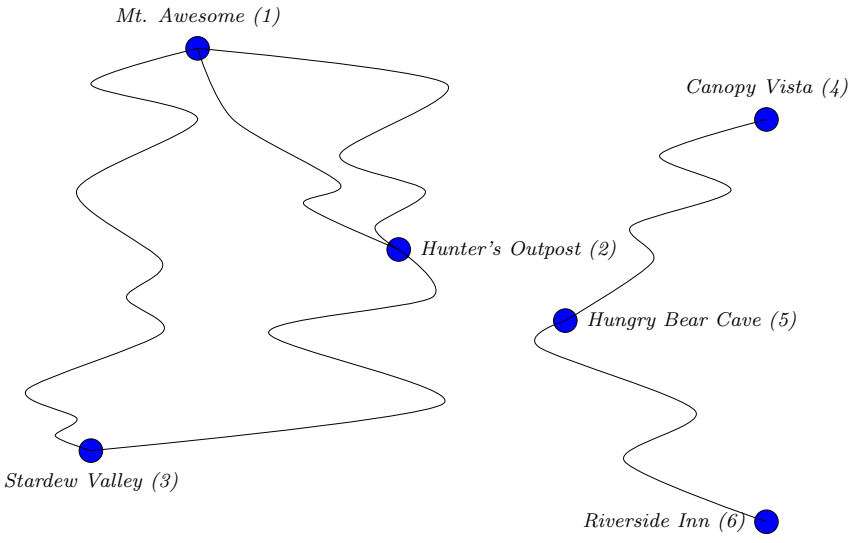


Figure C.1: Map of the second sample case

Sample Input 1

```
5 5
1 2 15
2 3 12
1 4 17
4 2 11
5 4 9
```

Sample Output 1

```
40
```

Sample Input 2

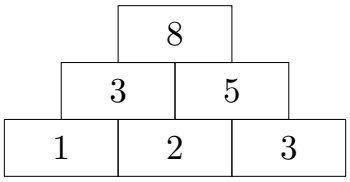
```
6 6
1 2 2
4 5 2
2 3 3
1 3 2
5 6 2
1 2 4
```

Sample Output 2

```
7
```

Problem D: Down the Pyramid

Do you like number pyramids? Given a number sequence that represents the base, you are usually supposed to build the rest of the "pyramid" bottom-up: For each pair of adjacent numbers, you would compute their sum and write it down above them. For example, given the base sequence $[1, 2, 3]$, the sequence directly above it would be $[3, 5]$, and the top of the pyramid would be $[8]$:



However, I am not interested in completing the pyramid – instead, I would much rather go underground. Thus, for a sequence of n non-negative integers, I will write down a sequence of $n + 1$ non-negative integers below it such that each number in the original sequence is the sum of the two numbers I put below it. However, there may be several possible sequences or perhaps even none at all satisfying this condition. So, could you please tell me how many sequences there are for me to choose from?

Input

The input consists of:

- one line with the integer n ($1 \leq n \leq 10^6$), the length of the base sequence.
- one line with n integers $a_1, \ldots, a_n$ ($0 \leq a_i \leq 10^8$ for each i), forming the base sequence.

Output

Output a single integer, the number of non-negative integer sequences that would have the input sequence as the next level in a number pyramid.

Sample Input 1

```
6
12 5 7 7 8 4
```

Sample Output 1

```
2
```

Sample Input 2

```
3
10 1000 100
```

Sample Output 2

```
0
```

Problem E: Expired License

Paul is an extremely gifted computer scientist who just completed his master's degree at a prestigious German university. Now he would like to culminate his academic career in a PhD. The problem is that there are so many great universities out there that it is hard for him to pick the best. Because some application deadlines are coming up soon, Paul's only way to procrastinate his decision is by simply applying to all of them.

Most applications require Paul to attach a portrait photo. However, it seems like there does not exist an international standard for the aspect ratio of these kinds of photos. While most European universities ask Paul to send a photograph with aspect ratio 4.5 by 6, some Asian countries discard the applications immediately if the photo does not have an aspect ratio of 7.14 by 11.22, precisely.

As Paul has never been interested in photo editing, he never had a reason to spend a lot of money on proper software. He downloaded a free trial version some months ago, but that version has already expired and now only works with some funny restrictions. The cropping tool, for example, no longer accepts arbitrary numbers for setting the aspect ratio, but only primes. This makes Paul wonder whether the desired aspect ratios can even be properly expressed by two prime numbers. Of course, in case this is possible, he would also like to know the primes he has to enter.

Input

The input consists of:

- one line with an integer n ($1 \leq n \leq 10^5$), the number of applications Paul has to file;
- n lines, each with two real numbers a and b ($0 < a, b < 100$), where $a \times b$ is the desired aspect ratio of one application.

All real numbers are given with at most 5 decimal places after the decimal point.

Output

For each application, if it is possible to represent the desired aspect ratio by two prime numbers p and q, output one line with p and q. Otherwise, output `impossible`. If multiple solutions exist, output the one minimizing $p + q$.

Sample Input 1

```
3
4.5 6
7.14 11.22
0.00002 0.00007
```

Sample Output 1

```
impossible
7 11
2 7
```

Problem I: It's Time for a Montage

The heroes of your favorite action TV show are preparing for the final confrontation with the villains. Fundamentally, there are two rivals who will fight each other: a very important main hero who wants to save the universe and an equally important main villain who wants to destroy it. However, through countless recursive spin-offs, they may have slightly less important sidekicks (a hero and a villain who are rivals themselves), who in turn may also have their own (even less important) sidekicks, and so on. Note that there is an equal number of heroes and villains, and each rival pair has at most one sidekick pair.

Initially, every character will fight their rival, with the winner being determined by who has the higher *Power Level*. If a hero and their corresponding villain have the same Power Level, their battle will be determined by their sidekicks' battle, as the winning sidekick can help as a sort of tiebreaker. (If rivals of equal Power Level do not have sidekicks, the hero character will win with the help of random passersby.) However, whenever a battle is won by either side, there is nothing the sidekicks can do about it – this is because the people behind the show believe some fans might get upset if a character were to get defeated by a bunch of less important characters, so they would lose regardless of the Power Levels.

After the battles between rivals (and possible tiebreakers) are done, the most important character remaining will defeat the rest of the opposing side and determine the fate of the universe. Fortunately, the heroes can ensure victory through hard, rigorous training. For each day they spend training, the Power Level of each hero increases by 1, while the villains' Power Levels remain constant.

But you already knew all this. The question plaguing your mind is how long the training is going to take.

Input

The input consists of:

- one line with an integer n ($1 \leq n \leq 1\,000$), giving the number of rival pairs.
- one line with n integers $h_1, \ldots, h_n$ ($1 \leq h_i \leq 1\,000$ for each i), the i-th value giving the Power Level of the i-th most important hero.
- one line with n integers $v_1, \ldots, v_n$ ($1 \leq v_i \leq 1\,000$ for each i), the i-th value giving the Power Level of the i-th most important villain.

Output

Output a single integer, the minimum number of days the heroes need to spend training in order for their side to win.

Sample Input 1

```
4
5 3 1 1
8 6 9 1
```

Sample Output 1

```
4
```

Sample Input 2

```
1
2
1
```

Sample Output 2

```
0
```

Sample Input 3

```
2
4 2
7 5
```

Sample Output 3

```
3
```

Problem L: Logic Puzzle

While browsing a kiosk at a recent trip, you bought a magazine filled with various kinds of logic puzzles. After a while of solving, however, you start to get a bit bored of the puzzles. Still wanting to complete all the puzzles in the magazine, you start wondering about ways to solve some of them algorithmically.

The puzzle you are currently trying to solve is called *Mosaic*, and it is quite similar to the classic *Minesweeper* video game:

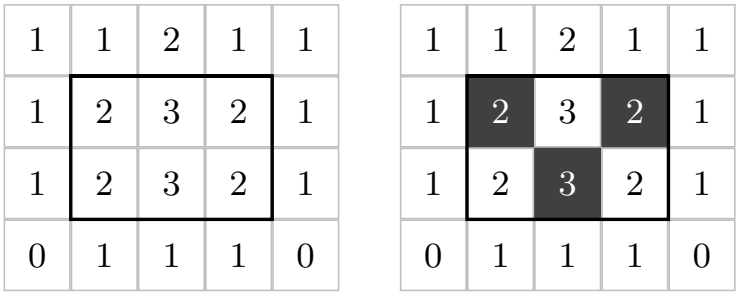


Figure L.1: Illustration of the first sample

You are given a two-dimensional grid of cells, initially all white, and you have to color some of the cells in black. You are also given a grid of clue numbers, which extends beyond the borders of the puzzle grid by one cell in each direction. The number in a cell indicates (exactly) how many cells in the 3×3 block centered at this cell need to be colored in black. You may not color any cells outside of the original grid.

Input

The input consists of:

- one line with two integers h, w ($1 \leq h, w \leq 100$), the height and width of the puzzle;
- $h + 2$ lines, each with $w + 2$ integers $c_1, \ldots, c_{w+2}$ ($0 \leq c_i \leq 9$), the clue numbers.

Output

If the given clue numbers are inconsistent, output `impossible`. Otherwise, output h lines with w characters each, the solution to the puzzle. Use `X` for black cells and `.` for white cells. If there are multiple solutions, any of them will be accepted.

Sample Input 1

```
2 3
1 1 2 1 1
1 2 3 2 1
1 2 3 2 1
0 1 1 1 0
```

Sample Output 1

```
X.X
.X.
```

Sample Input 2

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

Sample Output 2

```
impossible
```