A. Lunar New Year and a Wander

3 seconds, 256 megabytes

Lunar New Year is approaching, and Bob decides to take a wander in a nearby park.

The park can be represented as a connected graph with n nodes and m bidirectional edges. Initially Bob is at the node 1 and he records 1 on his notebook. He can wander from one node to another through those bidirectional edges. Whenever he visits a node not recorded on his notebook, he records it. After he visits all nodes at least once, he stops wandering, thus finally a permutation of nodes $a_1, a_2, \ldots, a_n$ is recorded.

Wandering is a boring thing, but solving problems is fascinating. Bob wants to know the lexicographically smallest sequence of nodes he can record while wandering. Bob thinks this problem is trivial, and he wants you to solve it.

A sequence x is lexicographically smaller than a sequence y if and only if one of the following holds:

- x is a prefix of y, but $x \neq y$ (this is impossible in this problem as all considered sequences have the same length);
- in the first position where x and y differ, the sequence x has a smaller element than the corresponding element in y.

Input

The first line contains two positive integers n and m ($1 \leq n, m \leq 10^5$), denoting the number of nodes and edges, respectively.

The following m lines describe the bidirectional edges in the graph. The i-th of these lines contains two integers u_i and v_i ($1 \leq u_i, v_i \leq n$), representing the nodes the i-th edge connects.

Note that the graph can have multiple edges connecting the same two nodes and self-loops. It is guaranteed that the graph is connected.

Output

Output a line containing the lexicographically smallest sequence $a_1, a_2, \ldots, a_n$ Bob can record.

input

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

output

```
1 2 3
```

input

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

output

```
1 4 3 2 5
```

input

```
10 10
1 4
6 8
2 5
3 7
9 4
5 6
3 4
8 10
8 9
1 10
```

output

```
1 4 3 7 9 8 6 5 2 10
```

In the first sample, Bob's optimal wandering path could be $1 \to 2 \to 1 \to 3$. Therefore, Bob will obtain the sequence $\{1, 2, 3\}$, which is the lexicographically smallest one.

In the second sample, Bob's optimal wandering path could be $1 \to 4 \to 3 \to 2 \to 3 \to 4 \to 1 \to 5$. Therefore, Bob will obtain the sequence $\{1, 4, 3, 2, 5\}$, which is the lexicographically smallest one.

B. Firetrucks Are Red

5 seconds, 256 megabytes

Lily is fascinated by numbers. She believes the whole world revolves around them, and that everything is connected by numbers. Her friends, Alice, Bob, Charlie and Diane, are not convinced. But she gives them an example:
"Alice lives in house number 25 on her street, but that is exactly Bob's age. Bob is born on June 4th, and Charlie was his parents' fourth child. Finally, Diane has five fingers on her left hand, which happens to be the same as the number of toes that Bob has on his right foot!"

This shows that her friends are all connected—either directly or indirectly—by numbers. But she still has to convince her family as well as her coworkers.

Given a group of n individuals, and a set of numbers that describe each individual, help Lily come up with a proof that shows that everyone in this group is either directly or indirectly connected by numbers, or determine that this is not possible.

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 2 \cdot 10^5$), the number of individuals in the group. The individuals are numbered from 1 to n.
- n lines, describing the individuals in the group.
  The ith such line starts with an integer m_i ($1 \leq m_i \leq 2 \cdot 10^5$), the number of numbers that describe individual i.

  The remainder of the line has m_i distinct integers $d_{i,1}, \ldots, d_{i,m_i}$ ($1 \leq d_{i,j} \leq 10^9$ for each j), the set of numbers that describe individual i.

It is guaranteed that the sum over all m_i is at most $2 \cdot 10^5$.

Output

Output a proof in the form of $n - 1$ lines, each of which contains three integers p, q and r, where p and q are distinct individuals that are both described by the number r. Using only these relations, it must be possible to show that any pair of individuals in the group are connected either directly or indirectly.

If no such proof exists, output "`impossible`". If there are multiple proofs, you may output any one of them.

input

```
6
2 17 10
1 5
2 10 22
3 17 22 9
2 17 8
3 9 22 16
```

output

```
impossible
```

input

```
6
2 17 10
2 5 10
2 10 22
3 17 22 9
2 17 8
3 9 22 16
```

output

```
1 4 17
4 3 22
3 2 10
4 6 22
1 5 17
```

C. Famil Door and Brackets

2 seconds, 256 megabytes

As Famil Door's birthday is coming, some of his friends (like Gabi) decided to buy a present for him. His friends are going to buy a string consisted of round brackets since Famil Door loves string of brackets of length n more than any other strings!

The sequence of round brackets is called *valid* if and only if:

1. the total number of opening brackets is equal to the total number of closing brackets;
2. for any prefix of the sequence, the number of opening brackets is greater or equal than the number of closing brackets.

Gabi bought a string s of length m ($m \le n$) and want to complete it to obtain a valid sequence of brackets of length n. He is going to pick some strings p and q consisting of round brackets and merge them in a string $p + s + q$, that is add the string p at the beginning of the string s and string q at the end of the string s.

Now he wonders, how many **pairs** of strings p and q exists, such that the string $p + s + q$ is a valid sequence of round brackets. As this number may be pretty large, he wants to calculate it modulo $10^9 + 7$.

Input

First line contains n and m ($1 \le m \le n \le 100\,000$, $n - m \le 2000$) — the desired length of the string and the length of the string bought by Gabi, respectively.

The second line contains string s of length m consisting of characters '`(`' and '`)`' only.

Output

Print the number of pairs of string p and q such that $p + s + q$ is a valid sequence of round brackets modulo $10^9 + 7$.

input
```
4 1
(
```
output
```
4
```

input
```
4 4
(())
```
output
```
1
```

input
```
4 3
(((
```
output
```
0
```

In the first sample there are four different valid pairs:

1. p = "`(`", q = "`))`"
2. p = "`()`", q = "`)`"
3. p = "", q = "`())`"
4. p = "", q = "`)()`"

In the second sample the only way to obtain a desired string is choose empty p and q.

In the third sample there is no way to get a valid sequence of brackets.

D. Even or Odd

1 second, 256 megabytes

We are given n (possibly unfair) coins. For the i-th coin, the probability of flipping head is p_i, and the probability of flipping tail is $1 - p_i$. We would like to choose a subset S of the n coins, so that after tossing coins in S, the probability that the total number of heads is even equals the probability that the total number of heads is odd. Output the total number of such subset S.

Input

The first line contains a single integer T which is the number of test cases. It is guaranteed that $1 \leq T \leq 40$.

For each test case, the first line contains a single integer n with $1 \leq n \leq 50$. Each of the n following lines contains a single floating number p_i. It is guaranteed that $0 \leq p_i \leq 1$, and each p_i contains exactly six digits after the decimal point.

Output

For each test case, print the number of subsets of coins that have the required property.

input

```
3
1
0.500000
2
0.500000 0.499999
4
0.378458 0.666667 0.333333 0.200000
```

output

```
1
2
0
```

E. Odd One Out

1 second, 256 megabytes

We are given n sets of integers, each of which forms an arithmetic progression. Formally, the i-th set is described by three integers s_i, e_i and d_i and defined to be

$$F_i = \{s_i + kd_i \mid k \in \mathbb{Z}, 0 \leq k \leq (e_i - s_i)/d_i\}.$$

It is guaranteed that there is at most one x such that x appears in an odd number of the n sets $F_1, F_2, \ldots, F_n$. I.e., $|\{i \mid x \in F_i\}|$ is odd. Output x.

Input

The first line contains a single integer T which is the number of test cases. It is guaranteed that $1 \leq T \leq 5$. For each test case, the first line contains a single integer n with $1 \leq n \leq 200000$. Each of the following n lines contains three integers s_i, e_i and d_i. It is guaranteed that $s_i \leq e_i$, $d_i \geq 1$, and $0 \leq s_i, e_i, d_i \leq 2^{31} - 1$.

Output

For each test case, output x. If such x does not exist, output -1.

input

```
2
2
1 10 1
2 10 1
2
1 10 1
1 10 1
```

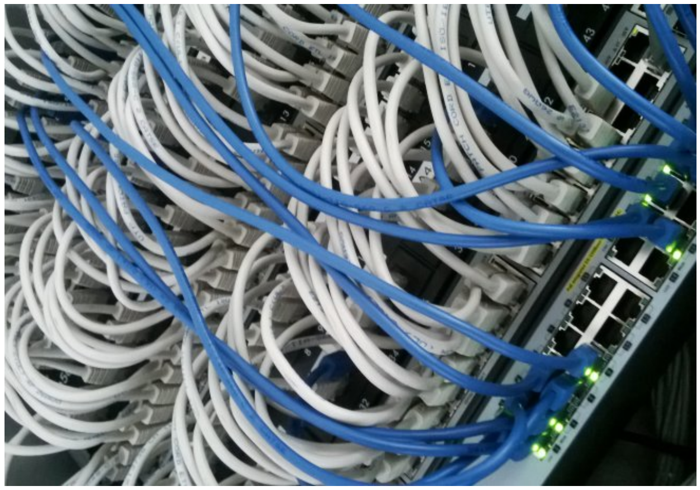
output

```
1
-1
```

Hint: Given some bound B, it's possible to count in $O(1)$ time the number of elements of F_i that are at most B. Use this, plus one of the techniques that we have covered so far.

F. Disposable Switches

4 seconds, 256 megabytes



Picture by Ted Sakshaug on Flickr, cc by

Having recently been hired by Netwerc Industries as a network engineer, your first task is to assess their large and dated office network. After mapping out the entire network, which consists of network switches and cables between them, you get a hunch that, not only are some of the switches redundant, some of them are not used at all! Before presenting this to your boss, you decide to make a program to test your claims.

The data you have gathered consists of the map of the network, as well as the length of each cable. While you do not know the exact time it takes to send a network packet across each cable, you know that it can be calculated as $\ell/v + c$, where

- ℓ is the length of the cable,
- v is the propagation speed of the cable, and
- c is the overhead of transmitting the packet on and off the cable.

You have not been able to measure v and c. The only thing you know about them is that they are real numbers satisfying $v > 0$ and $c \geq 0$, and that they are the same for all cables. You also know that when a network packet is being transmitted from one switch to another, the network routing algorithms will ensure that the packet takes an optimal path—a path that minimises the total transit time.

Given the map of the network and the length of each cable, determine which switches could never possibly be part of an optimal path when transmitting a network packet from switch 1 to switch n, no matter what the values of v and c are.

Input

The input consists of:

- One line with two integers n, m ($2 \leq n \leq 2\,000$, $1 \leq m \leq 10^4$), the number of switches and the number of cables in the network. The switches are numbered from 1 to n.
- m lines, each with three integers a, b and ℓ ($1 \leq a, b \leq n$, $1 \leq \ell \leq 10^9$), representing a network cable of length ℓ connecting switches a and b.

There is at most one cable connecting a given pair of switches, and no cable connects a switch to itself. It is guaranteed that there exists a path between every pair of switches.

Output

First output a line with an integer k, the number of switches that could never possibly be part of an optimal path when a packet is transmitted from switch 1 to switch n. Then output a line with k integers, the indices of the k unused switches, in increasing order.

input

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

output

```
2
4 6
```

input

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

output

```
0
```

input

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

output

```
1
4
```