# A. Subtractions

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You've got two numbers. As long as they are both larger than zero, they go through the same operation: subtract the lesser number from the larger one. If they equal substract one number from the another. For example, one operation transforms pair (4,17) to pair (4,13), it transforms (5,5) to (0,5).

You've got some number of pairs $(a_i, b_i)$. How many operations will be performed for each of them?

## Input

The first line contains the number of pairs $n$ $(1 \le n \le 1000)$. Then follow $n$ lines, each line contains a pair of positive integers $a_i, b_i$ $(1 \le a_i, b_i \le 10^9)$.

## Output

Print the sought number of operations for each pair on a single line.

**Sample test(s)**

input

```
2
4 17
7 987654321
```

output

```
8
141093479
```

# B. Dominoes

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a set of dominoes. Each domino is a rectangular tile with a line dividing its face into two square ends. Can you put all dominoes in a line one by one from left to right so that any two dominoes touched with the sides that had the same number of points? You can rotate the dominoes, changing the left and the right side (domino "1-4" turns into "4-1").

### Input

The first line contains number $n$ $(1 \le n \le 100)$. Next $n$ lines contains the dominoes. Each of these lines contains two numbers — the number of points (spots) on the left and the right half, correspondingly. The numbers of points (spots) are non-negative integers from 0 to 6.

### Output

Print "`No solution`", if it is impossible to arrange the dominoes in the required manner. If the solution exists, then describe any way to arrange the dominoes. You put the dominoes from left to right. In each of $n$ lines print the index of the domino to put in the corresponding position and then, after a space, character "+" (if you don't need to turn the domino) or "–" (if you need to turn it).

**Sample test(s)**

input

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

output

```
2 -
1 -
5 -
3 +
4 -
```

# C. Berland Traffic

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Berland traffic is very different from traffic in other countries. The capital of Berland consists of $n$ junctions and $m$ roads. Each road connects a pair of junctions. There can be multiple roads between a pair of junctions. For each road we know its capacity: value $c_i$ is the maximum number of cars that can drive along a road in any direction per a unit of time. For each road, the cars can drive along it in one of two direction. That it, the cars can't simultaneously move in both directions. A road's traffic is the number of cars that goes along it per a unit of time. For road ($a_i$, $b_i$) this value is negative, if the traffic moves from $b_i$ to $a_i$. A road's traffic can be a non-integer number.

The capital has two special junctions — the entrance to the city (junction 1) and the exit from the city (junction $n$). For all other junctions it is true that the traffic is not lost there. That is, for all junctions except for 1 and $n$ the incoming traffic sum equals the outgoing traffic sum.

Traffic has an unusual peculiarity in the capital of Berland — for any pair of junctions ($x$, $y$) the sum of traffics along any path from $x$ to $y$ doesn't change depending on the choice of the path. Such sum includes traffic along all roads on the path (possible with the "minus" sign, if the traffic along the road is directed against the direction of the road on the path from $x$ to $y$).

Your task is to find the largest traffic that can pass trough the city per one unit of time as well as the corresponding traffic for each road.

## Input

The first line contains a positive integer $n$ — the number of junctions ($2 \le n \le 100$). The second line contains integer $m$ ($1 \le m \le 5000$) — the number of roads. Next $m$ lines contain the roads' descriptions. Each road contains a group of three numbers $a_i$, $b_i$, $c_i$, where $a_i$, $b_i$ are the numbers of junctions, connected by the given road, and $c_i$ ($1 \le a_i, b_i \le n$; $a_i \ne b_i$; $0 \le c_i \le 10000$) is the largest permissible traffic along this road.

## Output

In the first line print the required largest traffic across the city. Then print $m$ lines, on each line print the speed, at which the traffic moves along the corresponding road. If the direction doesn't match the order of the junctions, given in the input, then print the traffic with the minus sign. Print the numbers with accuracy of at least five digits after the decimal point.

If there are many optimal solutions, print any of them.

**Sample test(s)**

input

```
2
3
1 2 2
1 2 4
2 1 1000
```

output

```
6.00000
2.00000
2.00000
-2.00000
```

input

```
7
11
1 2 7
1 2 7
1 3 7
1 4 7
2 3 7
2 5 7
3 6 7
4 7 7
5 4 7
5 6 7
6 7 7
```

output

```
13.00000
2.00000
2.00000
3.00000
6.00000
1.00000
3.00000
4.00000
7.00000
1.00000
2.00000
6.00000
```