Problem C: Card Trading

Recently, I got into playing the trading card game *Wizardry – The Meeting*. And since I really wanted to build an awesome deck, I decided to search online for only the best cards. It turns out most of those cards are quite expensive and can only be acquired by insane luck, when purchasing a random set of cards, or by bidding in online auctions. As auctions are a huge time sink and I really rather wanted to play instead of bidding the whole day, I came up with a different idea: A trading card marketplace.

Each card type is produced in bulk, so a buyer does not really care from which seller they buy a specific card. Therefore, the idea is to create one web page for each card type and users can set buy and sell offers. Take the card “Green Mana” for instance. If you wanted to buy one, you could create a *buy offer*, e.g. for 10.00€. This offer means that you are willing to buy the card for 10.00€ or less (if there is a seller for less). On the other hand, if you wanted to sell one “Green Mana” card, you could create a *sell offer*, e.g. for 12.01€. This offer means you are willing to sell your card for 12.01€ or more (if there is a buyer for more).

Now, every couple of seconds, the website automatically calculates a card price based on both types of offers. It then considers only those offers that are compatible with this price (as described above) and satisfies as many of those as possible.

As an aspiring entrepreneur, I decided that I deserve a cut of every sale happening on the website. But I have a little trouble to come up with an algorithm that determines the price such that the *turnover*, i.e. the price times the number of successful sales, is as high as possible (which would mean my cut being as high as possible).

Input

The input consists of:

- One line with one integer n ($1 \leq n \leq 10^5$), the number of different prices at which offers exist.
- n lines, each containing one real number p and two integers b and s ($0 < p \leq 10^4, 0 \leq b, s \leq 10^6$), the price of the offers with exactly two decimal places, the number of buy offers at this price and the number of sell offers at this price.

It is guaranteed that each price in the input has at least one buy or sell offer and that no price appears more than once.

Output

If no price exists, such that at least one sale occurs, output “`impossible`”. Otherwise, output the price resulting in the highest turnover and that turnover itself. If multiple such prices exist, output any. Output both numbers to *exactly* two decimal places.

Sample Input 1

```
5
12.00 0 3
11.99 2 0
11.98 5 0
10.00 1 0
12.01 0 6
```

Sample Output 1

```
impossible
```

Sample Input 2

```
6
2.85 14 0
4.50 0 1
5.26 3 3
6.17 1 0
14.78 0 2
21.04 1 0
```

Sample Output 2

```
5.26 21.04
```

Sample Input 3

```
6
2.85 14 0
4.50 0 1
5.26 2 3
14.78 0 2
1.83 0 1
21.04 1 0
```

Sample Output 3

```
21.04 21.04
```

Notes

In the second sample case, the optimal card price is 5.26€, as it results in the highest possible turnover of 21.04€, with four sales happening. In total, there are five buyers willing to pay at least 5.26€: Three are willing to pay exactly 5.26€, one is willing to pay 6.17€ and one is even willing to pay 21.04€. On the other hand, there are just four sellers willing to part with their card at 5.26€: Three at exactly this price and one would already be happy with 4.50€.

Note that there is an alternative solution: at a card price of 21.04€, there will be exactly one sale, resulting in the same optimal turnover.