2023 ICPC Texas State – Mexico Invitational Programming Contest

Apr. 1, 2023

Problem 1: Greetings

Description

Assumming many teenagers of the world have switched to the new hot social app called BAPC (Bidirectional and Private Communication) in 2023. This app has some stricter social rules than previous social apps such as Snapchat and Instrgram. For example, if someone greets you with "he...ey", you have to respond with "hee...eey" as well, but using twice as many e's!

Given a string of the form he...ey of length at most 1000, print the greeting you will respond with, containing twice as many e's.

Input Specification

- The input consists of one line containing a single string s as specified, of length at least 3 and at most 1000.

Output Specification

Sample Input 1	Sample Output 1
hey	heey

Sample Input 2	Sample Output 2
heeeeey	heeeeeeeeeey

Problem 2: Bracket Sequence

Description

Two great friends, Eddie John and Kris Cross, are attending the Brackets Are Perfection Conference. They wholeheartedly agree with the main message of the conference and they are delighted with all the new things they learn about brackets.

One of these things is a bracket sequence. If you want to do a computation with + and ×, you usually write it like so:

$(2 \times (2 + 1 + 0 + 1) \times 1) + 3 + 2$.

The brackets are only used to group multiplications and addi tions together. This means that you can remove all the operators. A bracket sequence can then be shortened to:

(2 (2 1 0 1) 1) 3 2.

That is much better, because it saves on writing all those operators. Reading bracket sequences is easy, too. Suppose you have the following bracket sequence:

5 2 (3 1 (2 2) (3 3) 1).

You start with addition, so this is the same as the following:

5 + 2 + (3 1 (2 2) (3 3) 1).

You know the parentheses group a multiplication, so this is equal to

5 + 2 + (3 × 1 × (2 2) × (3 3) × 1).

Then there is another level of parentheses: that groups an operation within a multiplication, so the operation must be addition.

$5 + 2 + (3 \times 1 \times (2 + 2) \times (3 + 3) \times 1) = 5 + 2 + (3 \times 1 \times 4 \times 6 \times 1) = 5 + 2 + 72 = 79$.

Since bracket sequences are so much easier than normal expressions with operators, it should be easy to evaluate some big ones by writing a program to do it for you.

Input Specification

- One line containing a single integer $1 \leq n \leq 3 \cdot 10^5$
- One line consisting of n tokens, each being either (,), or an integer $0 \leq x < 10^9 + 7$.

It is guaranteed that the tokens form a bracket sequence. Note that an empty ( ) is not a valid bracket sequence, nor a subsequence of any valid bracket sequence.

Output Specification

Output the value of the given bracket sequence. Since this may be very large, you should print it modulo $10^9 + 7$.

Sample Input 1	Sample Output 1
2 2 3	5

Sample Input 2	Sample Output 2
8 (2 (2 1)) 3	9

Sample Input 3	Sample Output 3
4 (12 3)	36

Sample Input 4	Sample Output 4
6 (2) (3)	5

Sample Input 5	Sample Output 5
6 ((2 3))	5

Sample Input 6	Sample Output 6
11 1 (0 (583920 (2839 82)))	1

Problem 3: Floor Plan

You are an architect and you have just been appointed to build a new swimming hall. The organisation behind these plans has acquired funding for a swimming pool and surrounding building as large as they want, but unfortunately they could not find anyone willing to pay for the floor surrounding the pool. They decided to pay for the floor tiles out of their own pocket. Because this has already cost them an arm and a leg, they want you to use all the floor tiles in your proposed plan.

Being an architect, you care for aesthetics. You see it as absolutely vital that both the swimming pool and the sur rounding building are perfect squares. This raises an in teresting problem: how can you make sure that the square shapes are guaranteed, while still using all the floor tiles the organisation bought?

Given the number of tiles n, find the length of the side of the building m and and the length of the side of the pool k such that $n = m2 - k2$, or print impossible if no suitable m and k exist.

Input Specification

- One line containing a single integer $1 \leq n \leq 10^9$

Output Specification

Print two non-negative integers m, k such that n = m2 −k2, or print impossible if no such integers exist. If there are multiple valid solutions, you may output any one of them.

Sample Input 1	Sample Output 1
7	4 3

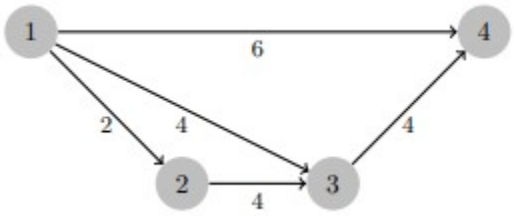
Sample Input 2	Sample Output 2
10	impossible

Sample Input 3	Sample Output 3
15	4 1

Problem 4: Parks Journey

Description

Together with some friends you are planning a holiday to the beautiful National Parks Centre. While there, you want to visit the parks as much as you can. As part of the preparation you, together with your best friend, decided to make an extensive map of the parks, the roads between them, and the lengths of these roads. To ensure that the visitors to the Parks Centre do not circle endlessly through the gorgeous parks, the routes between the parks are one-way only and are made such that it is impossible to visit a park more than once (that is, the corresponding graph is acyclic). Together you spend the entire day to create an incredibly detailed and admittedly rather large map of the Parks Centre.?



The map of the parks corresponding to sample input 1.

The next day disaster strikes: your friend happily announces that he got rid of the cumber some map! Instead, he decided to replace it with a simple table containing only the average distances

between the parks, weighing each route equally. Thus, he would give you an average distance of 8 between park 1 and park 4 as in figure J.1, because there are 3 paths and their average length is (6 + 8 + 10)/3 = 8. You feel defeated, and you dread the thought of making the map all over again. Perhaps it might be easier to try and use the table of average distances your friend made in order to reconstruct the original map. One thing you remember, is that the roads on the map were never inefficient. If there was a direct road between two parks,

then every path via at least one other park was strictly longer. This condition holds for the first sample input because, for example, the road from 1 to 4 has length 6, which is shorter than the length of any other path from 1 to 4. Given an input of average distances between parks, output the original map: a weighted directed acyclic graph for which these average distances hold.

Input Specification

- The first line contains an integer $1 \leq n \leq 100$, the number of vertices (parks).
- The next n lines contain the average distance from vertex i to vertex j as the jth number on the ith line, or -1 in case there is no path from vertex i to vertex j. All distances are either -1 or non-negative integers, at most 10^{15}.

You know the following facts about the map (the original graph):

- There is no way to follow the roads and return to a park once you have left it. That is, the graph is acyclic.
- The lengths of the roads in the original graph are all integers.
- Between any two parks, there are at most 1000 distinct paths you can take from one to the other.
- Direct roads are always efficient: if there is a direct road from park i to park j, every other path from i to j is strictly longer than that direct road.

Output Specification

Print n lines each containing n integers. The jth number on the ith line should be the positive length of the edge from vertex i to vertex j, or -1 if there is no such edge. It is guaranteed that a solution exists.

Sample Input 1

```
4
0 2 5 8
-1 0 4 8
-1 -1 0 4
-1 -1 -1 0
```

Sample Output 1

```
-1 2 4 6
-1 -1 4 -1
-1 -1 -1 4
-1 -1 -1 -1
```

Sample Input 2

```
6
0 -1 48 -1 132 -1
24 0 84 36 153 108
-1 -1 0 -1 84 -1
-1 -1 60 0 116 72
-1 -1 -1 -1 0 -1
-1 -1 -1 -1 96 0
```

Sample Output 2

```
-1 -1 48 -1 -1 -1
24 -1 -1 36 -1 -1
-1 -1 -1 -1 84 -1
-1 -1 60 -1 36 72
-1 -1 -1 -1 -1 -1
-1 -1 -1 -1 96 -1
```

Problem 5: COWVID

Description

In 2023, the pandemic of COVID among humans was finally under control. Unfortunately, an outbreak of the highly contagious bovine disease COVID was found in a farm managed by Farmer Tom. Despite his best attempt at making his N cows ($1 \leq N \leq 1000$) practice "social distancing", many of them still unfortunately contracted the disease. The cows, numbered 1...N, are each standing at distinct points along a long path (essentially a one-dimensional number line), with cow i standing at position x_i. Farmer Tom knows that there is a radius R such that any cow standing up to and including R units away from an infected cow will also become infected (and will then pass the infection along to additional cows within R units away, and so on).

Unfortunately, Farmer Tom doesn't know R exactly. He does however know which of his cows are infected. Given this data, please write a program to help Farmer Tom determine the minimum possible number of cows that were initially infected with the disease.

Input Specification

The first line of input contains N. The next N lines each describe one cow in terms of two integers, x and s, where x is the position ($0 \leq x \leq 10^6$), and s is 0 for a healthy cow or 1 for a sick cow. At least one cow is sick, and all cows that could possibly have become sick from spread of the disease have now become sick.

Output Specification

Please output the minimum number of cows that could have initially been sick, prior to any

spread of the disease.

Sample Input	Sample Output
6 7 1 1 1 15 1 3 1 10 0 6 1	3

In this example, we know that R<3 since otherwise the cow at position 7 would have infected the cow at position 10. Therefore, at least 3 cows must have started out infected - one of the two cows at positions 1 and 3, one of the two cows at positions 6 and 7, and the cow at position 15.

Problem 6: Knapsack Packing

Description

One of the most difficult things about going on a holiday is making sure your luggage does not exceed the maximum weight. You, chairman of the Backpacker's Association for Packing Carry-ons, are faced with exactly this problem. You are going on a lovely holiday with one of your friends, but now most of your time is spent in frustration while trying to pack your backpack. In order to optimize this process, you and your friend have independently set upon trying to find better ways to pack.

After some time you have a serious breakthrough! Somehow you managed to solve the Knapsack problem in polynomial time, defying expectations everywhere. You are not interested in any theoretical applications, so you immediately return to your friend's apartment in order to now quickly pack your backpack optimally.

When you arrive there, you find that your friend has set upon her own solution, namely to enumerate all possible packings. This means that all items you possibly wanted to bring are scattered across the entire apartment, and it would take a really long time to get all the items back together.

Luckily you can use the work your friend has done. For every possible subset of items that you can possibly bring, she has written down the total weight of these items. Alas, she did not write down what items were part of this total, so you do not know what items contributed to each total weight. If the original weights of the items formed a collection (a1, . . . , an) of non-negative integers, then your friend has written down the following multiset:

$$S((a_1,\ldots,a_n)) := \left\{ \sum_{i\in I} a_i \;\middle|\; I \subseteq \{1,\ldots,n\} \right\}.$$

For example, if your friend had two items, and the weights of those two items are 2, 3, then your

friend has written down:

- 0, corresponding to the empty set {};
- 2, corresponding to the subset {2};
- 3, corresponding to the subset {3};
- 5, corresponding to the subset {2, 3}.

You want to reconstruct the weights of all the individual items so you can start using your Knapsack algorithm. It might have happened that your friend made a mistake in adding all these weights, so it might happen that her list is not consisten.

Input Specification

- One line containing a single integer $1 \leq n \leq 18$ the number of items.
- 2^n lines each containing a single integer $0 \leq w \leq 2^{28}$, the combined weight of a subset of the items. Every subset occurs exactly once.

Output Specification

Output non-negative integers $a_1, \ldots, a_n$ on n lines in non-decreasing order such that $S((a_1, \ldots, a_n)) = \{b_1, \ldots, b_{2^n}\}$, provided that such integers exist. Otherwise, output a single line containing `impossible`.

Sample Input 1

```
1
0
5
```

Sample Output 1

```
5
```

Sample Input 2

```
3
7
5
2
4
1
6
3
0
```

Sample Output 2

```
1
2
4
```

Sample Input 3

```
2
0
1
2
4
```

Sample Output 3

```
impossible
```

Sample Input 4

```
2
0
1
1
2
```

Sample Output 4

```
1
1
```