Homework Assignment #5: Programming Exercise #1

Due Time/Date

2:20PM Tuesday, October 17, 2023. Late submission will be penalized by 20% for each working day overdue.

Problem Description

Solve Problem H "Prehistoric Programs" of the 2021 World Finals of the International Collegiate Programming Contest (see `https://icpc.global/worldfinals/problems/icpc2021.pdf` or the attachment).

Important Notes

This assignment constitutes 4% of your grade. You may discuss the problem with others, but copying code is strictly forbidden. **Some of you may be requested to demonstrate your program.**

Submission Guidelines

- Pack everything, excluding compiler-generated files, in a .zip file, named with the pattern "`b117050xx-hw5.zip`".
- Upload the .zip file to the NTU COOL site for Algorithms 2023.
- If you use a Makefile, make sure that it outputs "`hw5`". Otherwise, make sure that the whole application can be compiled by a single command like "`gcc hw5.c`", "`g++ hw5.cpp`", "`javac hw5.java`", etc.

Grading

Your work will be graded according to its correctness, efficiency, and presentation. Before submission, you should have tested your program on several input cases. You should organize and document your program in such a way that other programmers, for example your classmates, can understand it. In the documentation of your program (preferably in the code as comments), you are encouraged to describe how you have applied the algorithmic techniques, in particular design by induction, learned in class.

Below is a more specific grading policy:

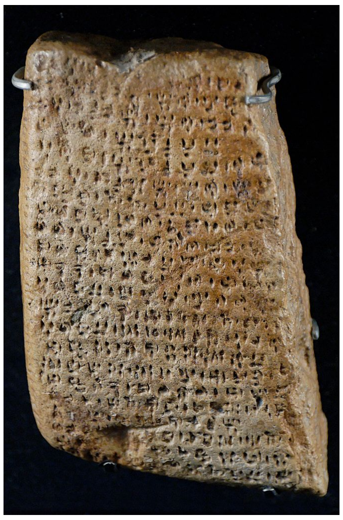
Criteria	Score
incomplete or doesn't compile	≤ 20
complete, compiles, but with major errors	≤ 40
complete, compiles, but with minor errors	≤ 70
correct (passing all test cases)	≤ 90
correct and reasonably efficient (at least around the class-average)	≤ 100
well-organized and with helpful code comments	$+ \leq 10$

Problem H
Prehistoric Programs
Time limit: 6 seconds

Archaeologists have discovered exciting clay tablets in deep layers of Alutila Cave. Nobody was able to decipher the script on the tablets, except for two symbols that seem to describe nested structures not unlike opening and closing parentheses in LISP. Could it be that humans wrote programs thousands of years ago?

Taken together, the tablets appear to describe a great piece of work – perhaps a program, or an epic, or even tax records! Unsurprisingly, after such a long time, the tablets are in a state of disorder. Your job is to arrange them into a sequence so that the resulting work has a properly nested parenthesis structure. Considering only opening and closing parentheses, a properly nested structure is either

- (), or
- (A), where A is a properly nested structure, or
- AB, where A and B are properly nested structures.


Clay tablet with undeciphered script
Source: Wikimedia Commons

Input

The first line of input contains one integer n ($1 \leq n \leq 10^6$), the number of tablets. Each of the remaining n lines describes a tablet, and contains a non-empty string of opening and closing parentheses; symbols unrelated to the nesting structure are omitted. The strings are numbered from 1 to n in the order that they appear in the input. The input contains at most 10^7 parentheses.

Output

Output a permutation of the numbers from 1 to n such that concatenating the strings in this order results in a properly nested structure. If this happens for multiple permutations, any one of them will be accepted. If there is no such permutation, output `impossible` instead.

Sample Input 1	Sample Output 1
2 ())())() ((()	2 1

Sample Input 2

```
5
(
))
((
))
(
```

Sample Output 2

```
1
5
3
2
4
```

Sample Input 3

```
2
((
)
```

Sample Output 3

```
impossible
```