

# Яблоневый сад

|                         |                   |
|-------------------------|-------------------|
| Имя входного файла:     | стандартный ввод  |
| Имя выходного файла:    | стандартный вывод |
| Ограничение по времени: | 2 секунды         |
| Ограничение по памяти:  | 512 мегабайт      |

Мальчик Женя захотел посадить в саду яблоню. Однако его жизнь была достаточно необычной, поэтому в своём возрасте из всех деревьев он знал только двоичные деревья поиска.

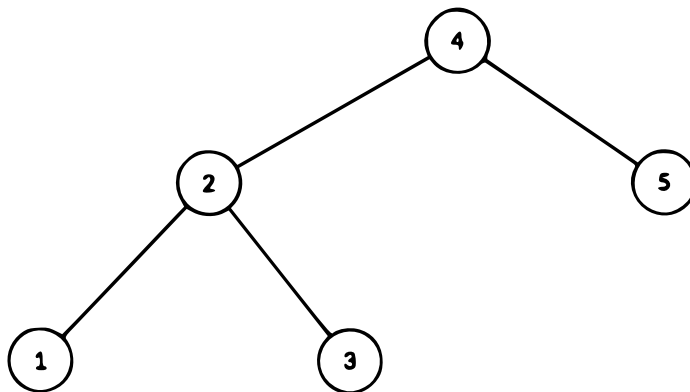
**Двоичное дерево** — это корневое дерево, в котором у каждой вершины есть не более двух сыновей (их называют *левым* и *правым* сыном).

**Двоичное дерево поиска** — это двоичное дерево, в каждой вершине которого записано число  $k$ , называемое *ключом вершины*, при этом верно следующее свойство:

1. Для любой вершины  $u$  из поддерева левого сына  $v$  выполняется  $k_u < k_v$ ;
2. Для любой вершины  $u$  из поддерева правого сына  $v$  выполняется  $k_v < k_u$ .

Женя придумал двоичное дерево поиска, ключами в котором является перестановка размера  $n$ . Однако сегодняшний вечер у Жени уже был занят: он планировал встретиться с друзьями в неизвестном ресторане «Брёвна». Поэтому, чтобы не забыть дерево, Женя решил записать информацию о нём на листочек. Рисовать он не умел, поэтому придумал специальный метод шифрования, чтобы гарантированно вспомнить дерево после встречи с друзьями:

```
def encode_tree(v):  
    if v is None:  
        return []  
    return [v.x] + encode_tree(v.left) + encode_tree(v.right)
```



Для приведённого дерева результат функции `encode_tree` будет `[4, 2, 1, 3, 5]`.

Однако Женя перехитрил сам себя: вернувшись домой вечером, он понял, что никак не может расшифровать написанные на листочке числа. Создание яблоневого сада под угрозой, поэтому вам требуется помочь Жене по заданной зашифрованной записи восстановить исходное дерево, либо же сообщить, что Женя что-то напутал, и по зашифрованной записи восстановить дерево невозможно.

## Формат входных данных

В первой строке задано целое число  $n$  ( $1 \leq n \leq 300\,000$ ) — количество вершин в дереве.

Во второй строке задано  $n$  целых чисел  $k_i$  ( $1 \leq k_i \leq n$ ) — результат выполнения функции `encode_tree`. Гарантируется, что массив  $k_i$  — перестановка чисел от 1 до  $n$ .

## Формат выходных данных

Если исходные данные противоречивы, и восстановить дерево невозможно, то выведите `Impossible`.

В противном случае выведите  $n$  строк. В  $i$ -й строке выведите два числа: ключи левого и правого сына вершины с ключом  $i$ . Если левый или правый сын отсутствует, то выведите  $-1$ .

## Примеры

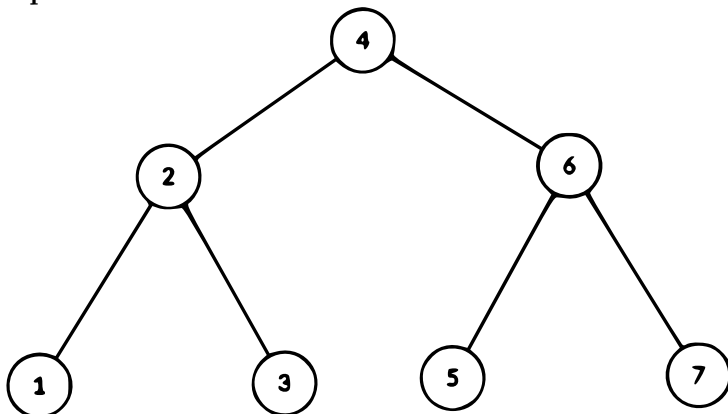
| стандартный ввод   | стандартный вывод                                     |
|--------------------|-------------------------------------------------------|
| 5<br>4 2 1 3 5     | -1 -1<br>1 3<br>-1 -1<br>2 5<br>-1 -1                 |
| 3<br>2 3 1         | Impossible                                            |
| 7<br>4 2 1 3 6 5 7 | -1 -1<br>1 3<br>-1 -1<br>2 6<br>-1 -1<br>5 7<br>-1 -1 |
| 3<br>1 2 3         | -1 2<br>-1 3<br>-1 -1                                 |

## Замечание

Первый тест — это пример из условия.

Во втором тесте подходящего дерева не существует.

Третий тест:



Четвёртый тест:

