# D Delicious Trees

Time limit: 3s

There is a Group of Adventurous, Playful Coatis living nearby a magnificent fruit tree. It provides the band with delicious fruits throughout the year. The coatis have maximized the tree's potential by keeping it in the shape of an AVL tree.[1]

The coatis climbing in their AVL tree, deciding how to cut it. CC BY-NC-ND 2.0 by Cloudtail the Snow Leopard on Flickr

The group has grown a lot over the years, and this single tree is no longer enough to satisfy the hunger of the many coati kits. One particularly smart coati, named Stefan, comes up with the idea to apply the concept of parallellization to this tree: rather than having one big tree, they should cut it into smaller trees, each of which can then grow new fruit to feed the hungry band. Obviously, after the big tree has been cut up, each of the smaller trees should be an AVL tree. Find any way to cut the AVL tree into some predetermined number of smaller AVL trees, or say this is impossible.

As an example, consider the first sample input, a fully balanced binary tree with height 3. Cutting the edges to the parents of vertices 2 and 3 produces three AVL trees: one tree containing only vertex 1, and two fully balanced trees with height 2. Note that it is allowed that a tree is temporarily unbalanced in between the first and the last cut.

## Input

The input consists of:

- One line with two integers $n$ and $k$ ($2 \leq n, k \leq 10^5$), the number of vertices in the big AVL tree and the number of smaller trees they want to end up with.
- $n$ lines, the $i$th of which contains two integers $l$ and $r$ ($0 \leq l, r \leq n$), the indices of the left and right child of the $i$th vertex, or 0 if that child is absent.

It is guaranteed that the input is a valid AVL tree rooted at vertex 1.

## Output

If it is impossible to end up with $k$ smaller AVL trees, output "`impossible`".
Else, output $k-1$ numbers between 2 and $n$ (inclusive), indicating that the edges to the parents of these $k-1$ vertices should be cut, to get $k$ AVL trees.

If there are multiple valid solutions, you may output any one of them.

[1] An AVL tree is a binary tree with the constraint that at each vertex, the difference between the two depths of its children's (possibly absent) subtrees is at most 1. Note that the depth of an absent subtree is equal to 0.

**Sample Input 1**

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

**Sample Output 1**

```
2 3
```

**Sample Input 2**

```
3 42
2 3
0 0
0 0
```

**Sample Output 2**

```
impossible
```