Stacks of Flapjacks

Background

Stacks and Queues are often considered the bread and butter of data structures and find use in architecture, parsing, operating systems, and discrete event simulation. Stacks are also important in the theory of formal languages.

This problem involves both butter and sustenance in the form of pancakes rather than bread in addition to a finicky server who flips pancakes according to a unique, but complete set of rules.

The Problem

Given a stack of pancakes, you are to write a program that indicates how the stack can be sorted so that the largest pancake is on the bottom and the smallest pancake is on the top. The size of a pancake is given by the pancake's diameter. All pancakes in a stack have different diameters.

Sorting a stack is done by a sequence of pancake ``flips''. A flip consists of inserting a spatula between two pancakes in a stack and flipping (reversing) the pancakes on the spatula (reversing the sub-stack). A flip is specified by giving the position of the pancake on the bottom of the sub-stack to be flipped (relative to the whole stack). The pancake on the bottom of the whole stack has position 1 and the pancake on the top of a stack of *n* pancakes has position *n*.

A stack is specified by giving the diameter of each pancake in the stack in the order in which the pancakes appear.

For example, consider the three stacks of pancakes below (in which pancake 8 is the top-most pancake of the left stack):

```
         8                7                2
         4                6                5
         6                4                8
         7                8                4
         5                5                6
         2                2                7
```

The stack on the left can be transformed to the stack in the middle via *flip(3)*. The middle stack can be transformed into the right stack via the command *flip(1)*.

The Input

The input consists of a sequence of stacks of pancakes. Each stack will consist of between 1 and 30 pancakes and each pancake will have an integer diameter between 1 and 100. The input is terminated by end-of-file. Each stack is given as a single line of input with the top pancake on a stack appearing first on a line, the bottom pancake appearing last, and all pancakes separated by a space.

The Output

For each stack of pancakes, the output should echo the original stack on one line, followed by some sequence of flips that results in the stack of pancakes being sorted so that the largest diameter pancake is on the bottom and the smallest on top. For each stack the sequence of flips should be terminated by a 0 (indicating no more flips necessary). Once a stack is sorted, no more flips should be made.

Sample Input

```
1 2 3 4 5
5 4 3 2 1
5 1 2 3 4
```

Sample Output

```
1 2 3 4 5
0
5 4 3 2 1
1 0
5 1 2 3 4
1 2 0
```