Problem Wk.3.3.2: Indexing Nested Lists

It would be handy to have a procedure that allows accessing lists that are nested to arbitrary depth. It would take a nested list and some sort of an index, and return the part of the list at that index (which could be a list or a primitive type such as a number or a string). Consider the nested list:

```
nested = \
[[[1, 2],
  3],
 [4,
  [5, 6]],
 7,
 [8, 9, 10]]
```

To select the element 9 out of it, we need to do something like

```
nested[3][1]
```

However, note that the level of nesting of the element we want shows up in the expression. If we wanted to get the element 5, we would need:

```
nested[1][1][0]
```

In this way, we cannot write a general procedure that takes the location of an element in the list and gets us the element. We want you to write a recursive procedure `recursiveRef` that would work as follows:

```
>>> recursiveRef(nested, [3, 1])
9
>>> recursiveRef(nested, [1, 1, 0])
5
>>> recursiveRef(nested, [1, 1])
[5, 6]
```

Note that the indices are lists of integers.

Think about what the base case for the recursion should be? Hint: What should be the value of `recursiveRef(`*`anything`*`, [])`?

Assume that the indices are always valid, so you don't have to do any error checking.

The tests are random, so Check more than once before Submit.