Assignment
Recursion

Constraints of this assignment

Because we want you to practice using recursion and understanding it, we will add constraints to this assignment. These are the constraints:

1. You cannot use loops
2. You cannot use variables

Most of these problems can be easily solved using loops and variables that you can assign different values throughout the function call. However, we are training your recursive thinking, so we are not allowing that, which increases the difficulty of the question ten-fold and forces you to use a recursive solution. For these examples, there is no reason to use recursion, but once you master recursion, you'll be able to solve Backtracking and Tree problems way easier. For the final Sudoku project, you will be using recursion a lot.

This is actually a good way to understand functional programming and pure functions. If you're interested you should read about Functional Programming vs Imperative Programming. The reason we don't use variables since they can store state and pure functional programming consists of ensuring that functions, inside the functional paradigm, will only depend on their arguments, regardless of any global or local state.

Problem 1: Is List Increasing order?

Given a list, determine if the list is in increasing order.

```
def is_increasing(lst):
    # your code here
    return
```

Here is what the output should look like:

```
> is_increasing([1,2,3,4])
True
> is_increasing([1,3,2,4])
False # because 3 is bigger than 2, so it decreased.
```

Problem 2: Numbers greater than n

Given a list, output a new list that only retains the numbers from the list that is greater than n

```
def filter_gt_n(lst, n):
    # your code here
    return
```

Here is what the output should look like:

```
> filter_gt_n([1,2,3,4], 2)
[3, 4] # 2 is not included because it is not GREATER than 2
> filter_gt_n([2,2,3,3], 1)
[2, 2, 3, 3]
> filter_gt_n([2,2,3,3], 4)
[]
```

Problem 3 (Challenge): Find the intersection of two lists

This is the same question as to the assignment for Loops Problem 2. However, this time you cannot use loops or variables. Hint: Write helper functions.

Lists may contain duplicates, but do not have duplicates in your produced lists.

```
def intersection(lst1, lst2):
    # your code here
    return
```

Here is what the output should look like:

```
> intersection([1,2,3], [2,4,6])
[2]
> intersection([1,2,3], [4,5,6])
[]
> intersection([2,3,2,4], [2,2,4])
[2,4]
```