02-201 Programming Practice Problems # 2

Carl Kingsford

11. Suppose you have a type:

```
type Node struct {
   a int
   next *Node
}
```

That can be used to represent a linked list. Write a function

```
kthFromEnd(start *Node, k int) int
```

that returns the integer in the node of the linked list that is `k` nodes from the end (i.e. if `k=0` you should return the last integer.) You can assume that the linked list ends when `next==nil`, and that `k` is less than the number of nodes in the list.

12. Suppose you have the same `Node` type as in the previous problem. Write a function:

```
integerOfBase(start *Node, base int)
```

that treats each integer in the linked list as a digit in a larger integer of base `base`. You can assume all of the integers in the list are between 0 and `base-1`. The most-significant digit is at the node pointed to by `start` and the least-significant digit (the ones-digit) is at the end of the list. For example, a list $L = \boxed{1} \rightarrow \boxed{3} \rightarrow \boxed{5}$ would return 135 if called with `integerOfBase(L, 10)`.

13. Suppose you have a type `Stack` that has two methods:

```
func (s *Stack) push(i int)
func (s *Stack) pop() int
```

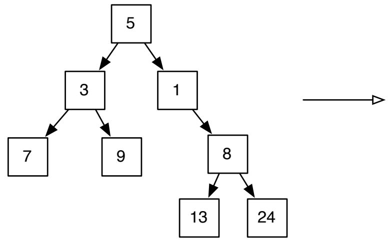
that push and pop integers as usual with Stacks. Implement a type `Queue` with to methods: `enqueue(i int)` and `dequeue() int` *using only calls to push and pop on stacks* — that is, you can't create any arrays or maps in your `Queue` type: you can only create `Stack`s. Hint: use 2 stacks.

14. **(Harder)** Suppose you have a type:

```
type TreeNode struct {
    a int
    left, right *TreeNode
}
```

that can be used to represent a binary tree, where `left` and `right` are the left and right children of the node. Write a function `isBinarySearchTree(t *TreeNode) bool` that returns `true` if `t`'s nodes are in binary search tree order.

15. Assume you have the same `TreeNode` type as in the previous problem. Write a function to print out the integers in a tree *one level at a time*, with each level on its own line. For example:




```
5
3 1
7 9 8
13 24
```