1 Question 1

Create a function `(odd-lengths t)` that consumes a BST and returns a new version of t where all values that are strings of odd length are replaced with the string "odd".

```
(define-struct node (key val left right))
;; A binary search tree (BST) is either
;; * () or
;; * (make-node Nat Any BST BST)...
```

2 Question 2

Create a function `leftmost` that consumes a non-empty SSTree and returns the leftmost label in the tree.

```
;; a simple search tree (SSTree) is either
;; * () or
;; * a SNode, where keys in left are less than key, and in right greater.
```

3 Question 3

Create a function `(llt-count value T)` that consumes a Num and a LLT and counts the number of times that value appears in T.

```
(llt-count 7 (list 7 (list 4 (list 7 6 5) 7))) => 3
```

Do not use flatten.

```
;; a leaf-labelled tree (LLT) is either
;; * a Num or
;; * a non-empty (listof LLT).
```

4 Question 4

Create a function `count-ops` that consumes a BinExp and returns the number of operations in that expression.

```
;; an Operator is (anyof '+ '-' '* '/)
define-struct binode (op arg1 arg2))
;; a binary arithmetic expression internal node (BINode)
;; is a (make-binode Operator BinExp BinExp)
;; A binary arithmetic expression (BinExp) is either:
;; * a Num or
```
5 Question 5

Exercise

Write a function swap-ops that consumes an AExp and returns a new AExp in which the '+ and '* operations have been swapped (i.e. all '*' become '+' and all '+' become '*').

;; an Operator is (anyof '+ '-' '*' '/')

(define-struct ainode (op args))

;; an arithmetic expression internal node (AINode)
;; is a (make-ainode Operator (listof AExp))
;; An arithmetic expression (AExp) is either:
;; a Num or
;; a AINode