Exercises on Reasoning about programs 10/1/2017

1)Using the definition

[] ++ ys = ys

(x:xs) ++ ys = x : (xs ++ ys)

verify the following two properties, by induction on xs:

xs ++ [] = xs

xs ++ (ys ++ zs) = (xs ++ ys) ++ zs

Hint: the proofs are similar to those for the add function.

2)Show that

exec (c ++ d) s = Exec d (exec c s)

where exec is the function that executes the Code consisting of sequences of PUSH n and ADD operations.

3)Given the type and instance declarations below, verify the functor laws for the Tree type, b induction on trees.

```
data Tree a = Leaf a | Node (Tree a) (Tree a)

instance Functor Tree where

--fmap :: (a-> b) -> Tree a -> Tree b

fmap  g   (Leaf x) = Leaf (g x)

fmap  g (Node l  r) = Node (fmap g l)  (fmap g  r)
```

4) Verify the applicative law for the Maybe type.

5) Given the equation comp' e c = comp e ++ c, show how to construct the recursive definition for comp' by induction on e.