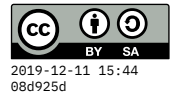


Assignment 6



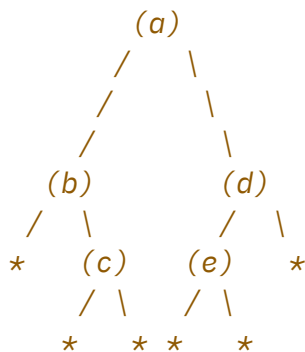
due *Tue 15 Oct*

module A06 **where**

```
{- For this assignment, we'll write some functions using the following
   tree data type, which stores elements at every node, and then uses an
   'Empty' constructor to represent the end of a branch.
-}
```

```
data Tree a
  = Empty
  | Branch a (Tree a) (Tree a)
deriving Show
```

```
{- Here's an example balanced tree: corresponding to this
   crude ASCII diagram:
```



```
-}
```

```
t1 :: Tree Char
t1 = Branch 'a' (Branch 'b' Empty (Branch 'c' Empty Empty))
              (Branch 'd' (Branch 'e' Empty Empty) Empty)
```

```
{- TODO: this should determine the total number of data elements
   stored in a tree, so for t1 that would be 5.
-}
```

```
treeSize :: Tree a -> Int
treeSize _ = 0
```

```
{- TODO: this should map a function over each element of a tree,
    keeping the same structure.
-}
```

```
mapTree :: (a -> b) -> Tree a -> Tree b
mapTree _ _ = error "TODO"
```

```
{- TODO: this converts a tree to a list of values, where "preorder"
    means the node should come BEFORE its left, and then its right. So
    for t1, that produces "abcde".
-}
```

```
preOrderList :: Tree a -> [a]
preOrderList _ = []
```

```
{- TODO: same idea, but now "inorder" which means the left side comes
    first, then the current node, then the right side. So for t1, that
    produces "bcaed".
-}
```

```
inOrderList :: Tree a -> [a]
inOrderList _ = []
```

```
{- TODO: height of a tree is the length of the longest path from root to
    an Empty node. (Empty itself has height zero.)
-}
```

```
treeHeight :: Tree a -> Int
treeHeight _ = 0
```

```
data Direction = GoLeft | GoRight
    deriving Show
```

```
type Path = [Direction]
```

```
{- TODO: Finally, try to recreate the find function for this type of
    tree. It returns Just with a list of directions for getting to the
    requested value, or Nothing if it's not in the tree.
-}
```

```
findInTree :: Eq a => a -> Tree a -> Maybe Path
findInTree _ _ = Nothing
```