Type Constructor vs Data Constructor:

- A **type constructor** is a function that takes 0 or more types and gives you back a new type. If it has zero arguments it is called a **nullary type constructor** or simply a **type**.
  Note: All type constructors must start with a capital letter.
- A **data constructor/tag** is a function that takes 0 or more values and gives you back a new value. If it has zero arguments is called a **nullary data constructor** or simply a **constant**.
  Note: All data constructors must start with a capital letter.
- In a data declaration, a type constructor is the thing on the left hand side of the equals sign. The data constructor(s) are the things on the right hand side of the equals sign. You use type constructors where a type is expected, and you use data constructors where a value is expected.
- Examples:
  1. This is an example of a nullary type constructor with 2 nullary data constructors:
     data Bool = True | False
  2. This is an example of a unary type constructor:
     data Tree a = Tip | Node a (Tree a) (Tree a)
     Here, “a” is a type variable.

Examples:

1. Consider the code and output below:

```
data BinaryTree = Empty | Node BinaryTree Integer BinaryTree
    deriving Show

insert :: Integer -> BinaryTree -> BinaryTree
insert k Empty = Node Empty k Empty
insert k bt@(Node left key right)
    |k < key = Node (insert k left) key right
    |k > key = Node left key (insert k right)
    |otherwise = bt

find:: Integer -> BinaryTree -> Maybe Integer
find k Empty = Nothing
find k (Node left key right)
    |k == key = Just(k)
    |k < key = find k left
    |k > key = find k right
```

```
*Main> x = insert 10 Empty
*Main> insert 9 x
Node (Node Empty 9 Empty) 10 Empty
*Main> insert 11 x
Node Empty 10 (Node Empty 11 Empty)
*Main> y = insert 9 x
*Main> insert 11 y
Node (Node Empty 9 Empty) 10 (Node Empty 11 Empty)
```

```
*Main> x = insert 10 Empty
*Main> y = insert 20 x
*Main> z = insert 14 y
*Main> z
Node Empty 10 (Node (Node Empty 14 Empty) 20 Empty)
*Main> find 14 z
Just 14
*Main> find 10 z
Just 10
*Main> find 20 z
Just 20
*Main> find 100 z
Nothing
*Main> find 0 z
Nothing
```

The line "**data BinaryTree = Empty | Node BinaryTree Integer BinaryTree**" means that BinaryTree is either Empty or Node BinaryTree Integer BinaryTree. Hence, when we output a BinaryTree for insert, we either have to output Empty or Node BinaryTree Integer BinaryTree.

2. Consider the code and output below:

```
data Boolean = T | F
    deriving(Show, Eq)

sumChecker :: (Num a, Ord a) => a -> a -> a -> Boolean
sumChecker x y z = if (x + y) == z
    then
        T
    else
        F

isBigger :: (Num a, Ord a) => a -> a -> Boolean
isBigger x y = if (x > y)
    then
        T
    else
        F

isSmaller :: (Num a, Ord a) => a -> a -> Boolean
isSmaller x y = if (x < y)
    then
        T
    else
        F

isEqual :: (Num a, Ord a) => a -> a -> Boolean
isEqual x y = if (x == y)
    then
        T
    else
        F
```

```
*Main> sumChecker 1 2 3
T
*Main> sumChecker 1 2 4
F
*Main> isBigger 1 2
F
*Main> isBigger 2 1
T
*Main> isSmaller 1 2
T
*Main> isSmaller 2 1
F
*Main> isEqual 1 2
F
*Main> isEqual 1 1
T
```

Here, Boolean is a type constructor while T and F are data constructors. As stated above, Boolean, T and F must be capitalized.