Part 7

—

A Stack

CSCI 3110 Code

Summer 2015

A stack is of course easy to implement as a list. Pushing an element prepends the element to the front of the list. Popping an element removes the frontmost element from the stack. To hide the implementation details, we wrap the list in a newtype:

```
newtype Stack t = Stack [t]
```

We want to be able to convert back and forth between a stack and a list:

```
stackFromList :: [t] -> Stack t
stackFromList = Stack

listFromStack           :: Stack t -> [t]
listFromStack (Stack xs) = xs
```

An empty stack can now be created from an empty list:

```
emptyStack :: Stack t
emptyStack = stackFromList [ ]
```

Finally, the basic stack operations: pushing, popping, and inspecting the top of the stack:

```
push              :: Stack t -> t -> Stack t
push (Stack xs) x = Stack (x : xs)

pop                 :: Stack t -> Stack t
pop (Stack (_ : xs)) = Stack xs

top               :: Stack t -> Maybe t
top (Stack [ ])    = Nothing
top (Stack (x: _)) = Just x
```