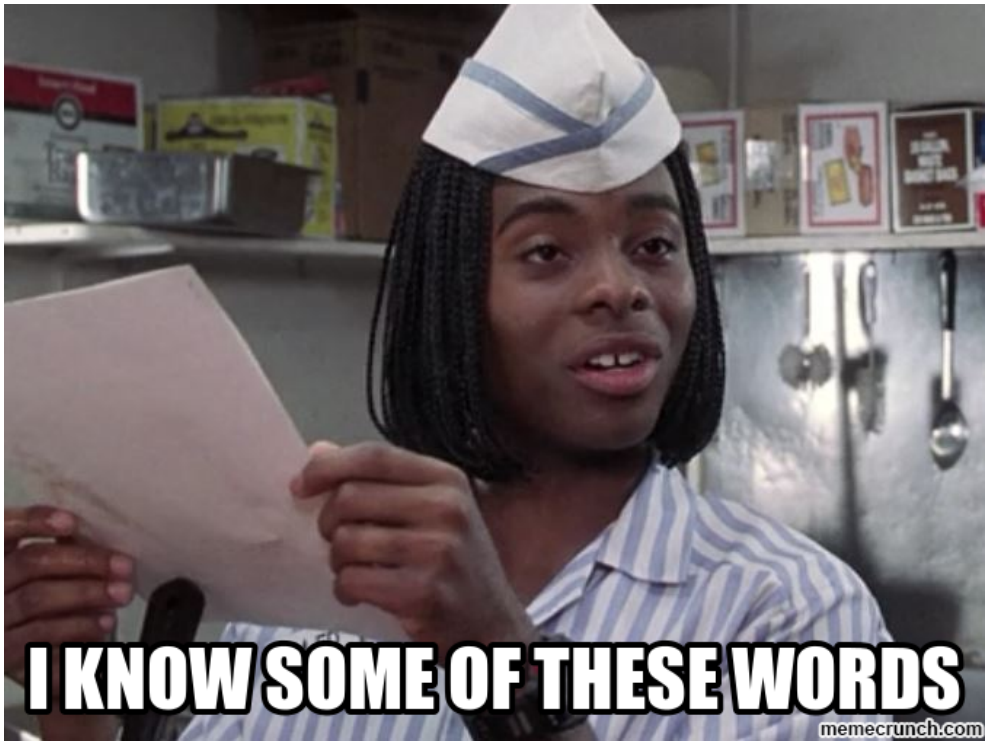
What the Haskell?!

A first look at functional programming

What is Haskell?

Haskell is a *polymorphically statically typed*, *lazily evaluated*, *purely functional* programming language.



So…what is functional programming?

- **Forget what you know about programming!**
- Focus on what things *are* rather than how they are computed.
- Try to minimize side effects
- First class functions
- Generally rely on recursion rather than loops

"Functional programming combines the flexibility and power of abstract mathematics with the intuitive clarity of abstract mathematics."

What else is Haskell?

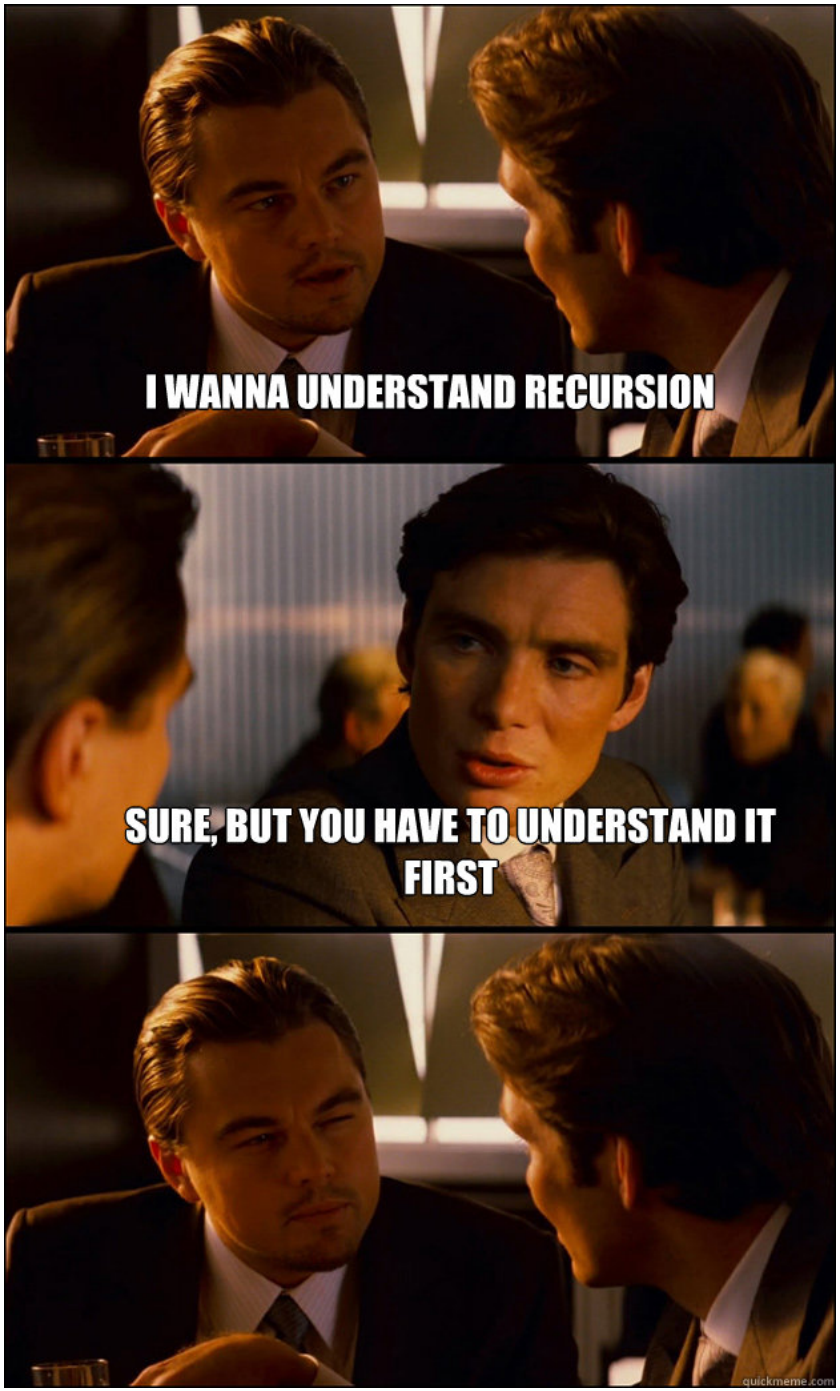
- Referentially transparent
  - Every expression will always yield the same results
- Data is completely immutable
- Lazily evaluated
  - "Only give me the value I want when I absolutely need it"
- Strongly Typed
- Parametrically Polymorphic
  - Functions can be evaluated on many different types of data in the exact same way
- Built around ideas from Category Theory

Let's do some stuff.

"The problem with Haskell is that it's a language built on lazy evaluation and nobody's actually called for it"

What the…? Basics

- Type Signatures
- Function application (no parenthesis!)
- Getting our feet wet with recursion

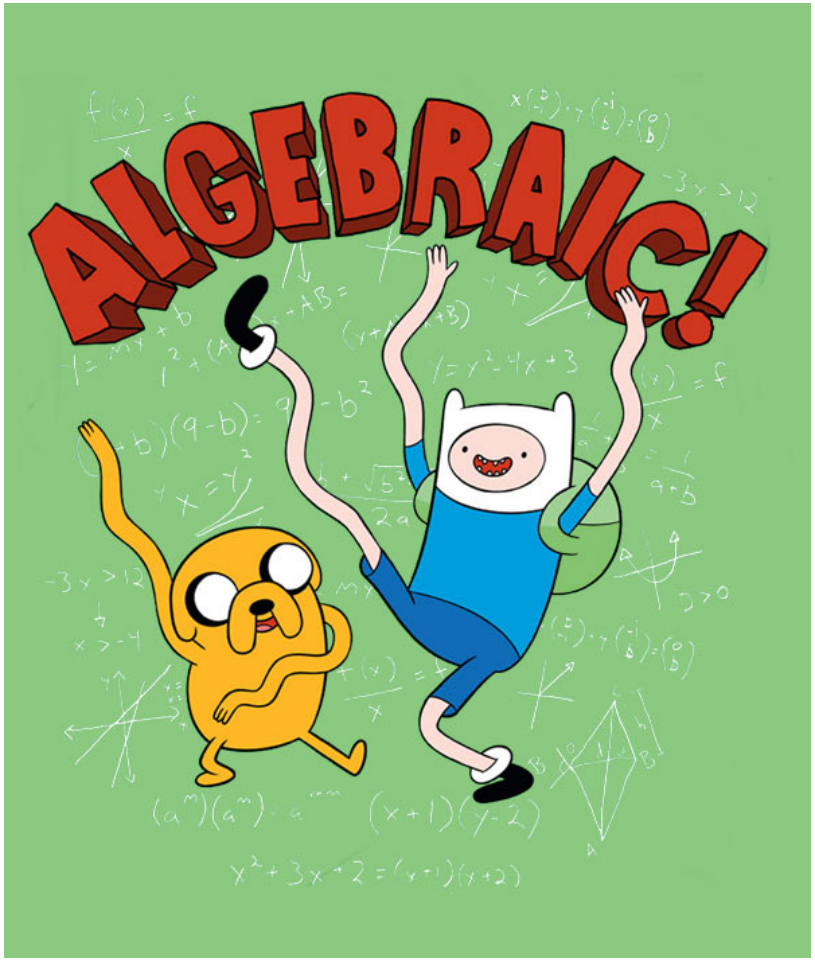


What the…?
Lists (`[a]`)

- Higher order functions: functions that take functions as arguments
- Lambdas: Anonymous functions
- Map: apply a function to every value in a list
- Folds: reduce a list with a function
- Maybe: Saves our butts when doing unsafe things
- Algebraic Data Types!

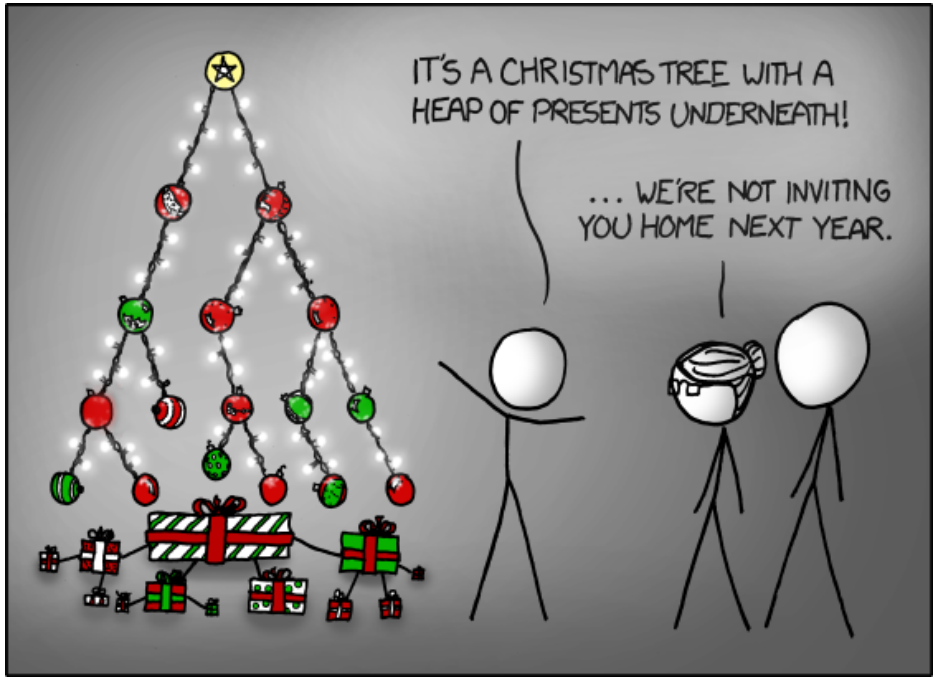
What the…?
Solving Project Euler # 8

- Sometimes, direct recursion is the easiest way to do things (`takes`)
- Function composition lets us write expressive, short, dense programs.



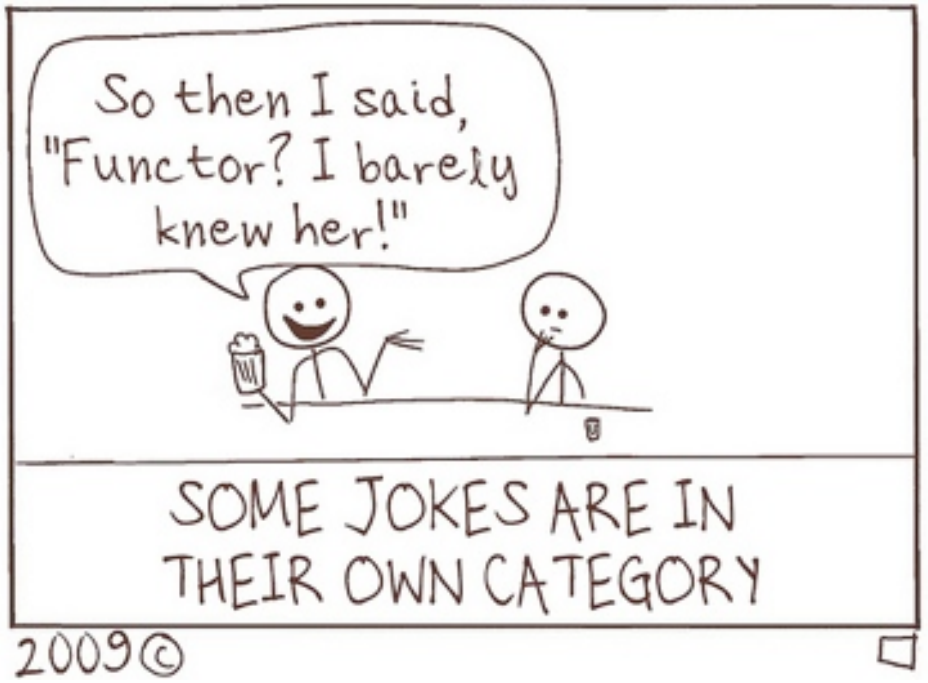
What the…?
Binary Search Tree

- Recursive data types are easily expressed.
- Maps and folds aren't specific to `[a]`
- Typeclasses at work (`Ord`)
- Again, programs tend to be short but dense (`treeSort`)!

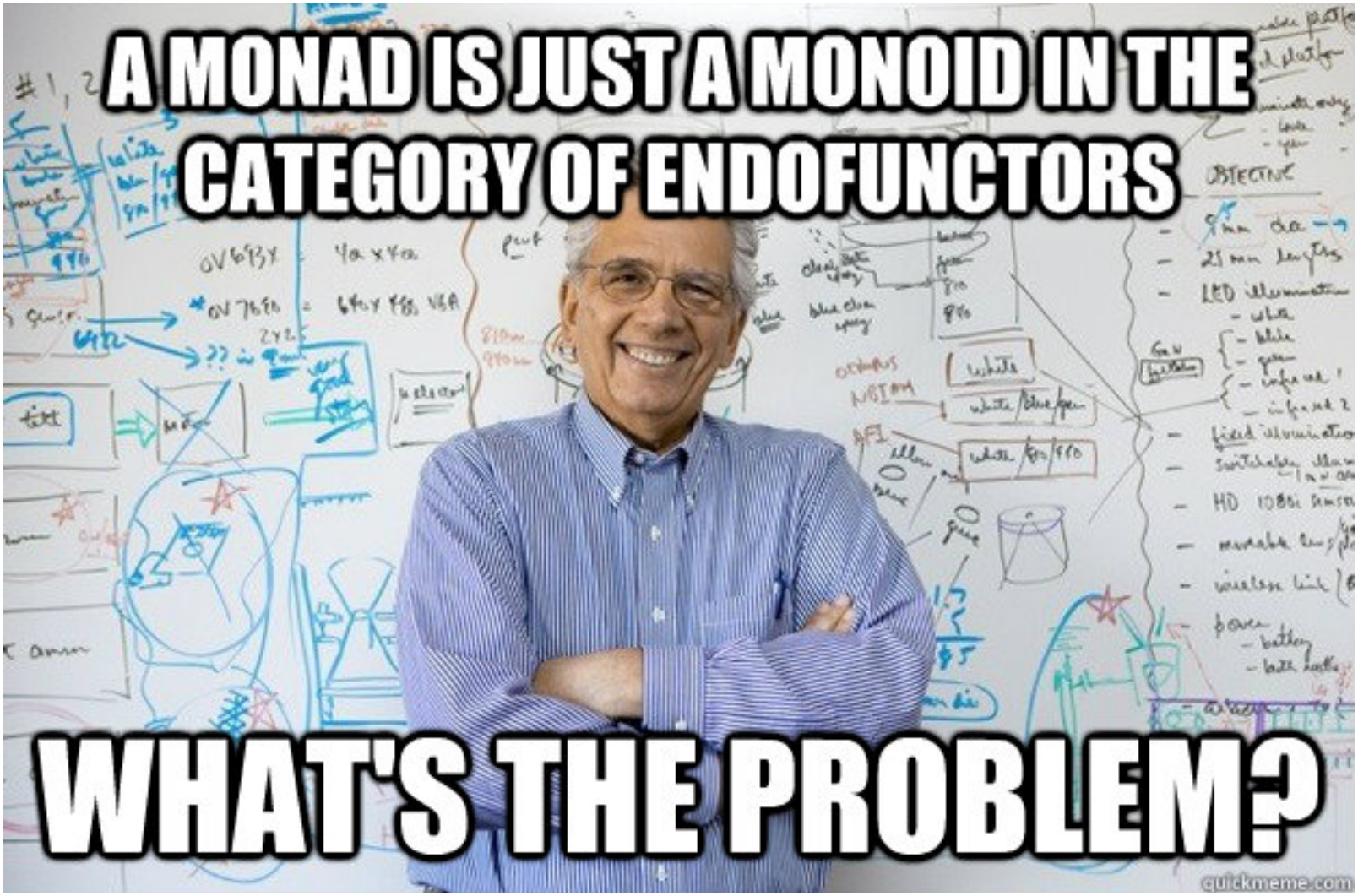


What the…? Functors

- Functors generalize the notion of mapping functions over data structures
- Functors are defined on Type *Constructors* (like `[]` or `Tree`)
- We can write functions that work on any `Functor` (a generic programming technique)!



What the…?
Monads



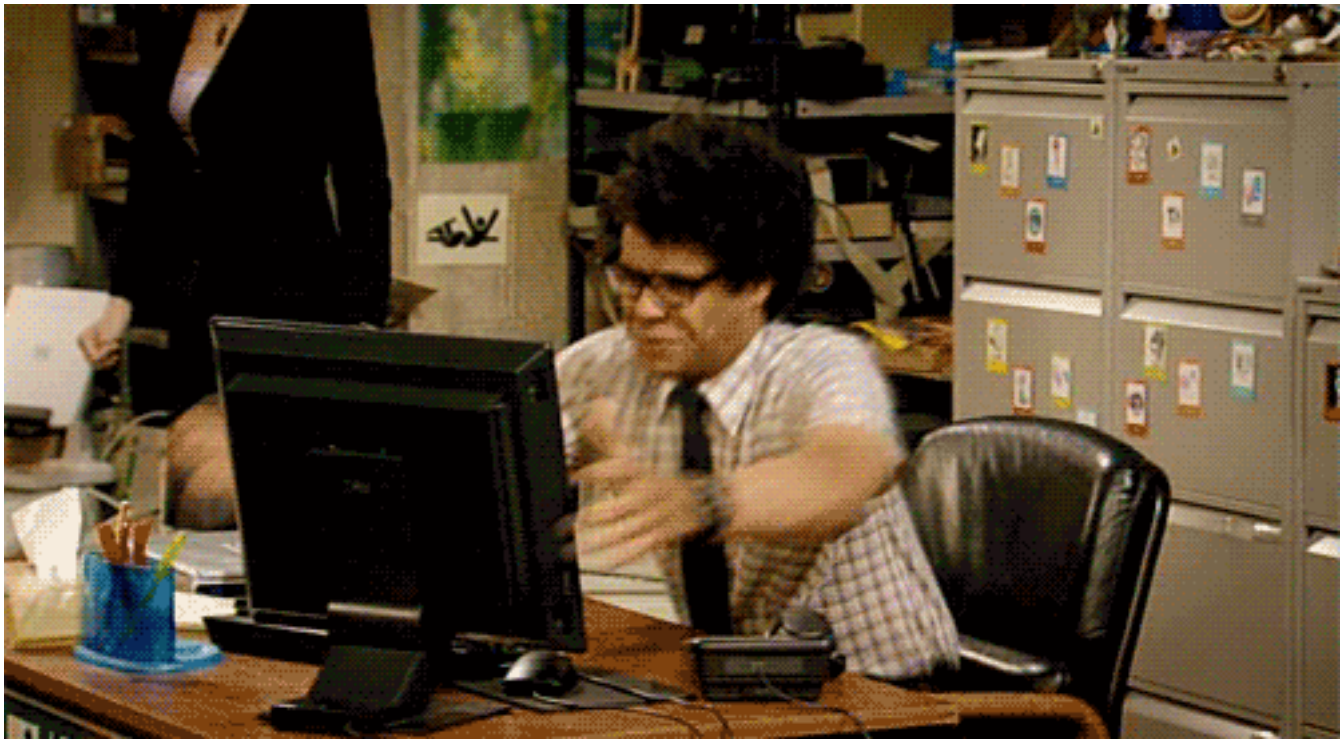
What the…?
Monads (for real)

- Monads let us make contextual values out of values wrapped in contextual values.
- Many interesting monads out there that we haven't talked about (`State`, `Cont`, `Random`, `Parser`)
- Do Notation is syntactic sugar for (>>=) that lets us "unwrap" monadic values.

Further Reading/ Learning Resources

- Learn You a Haskell For Great Good! – Miran Lipovača
- Real World Haskell – Bryan O'Sullivan
- Dan Piponi's blog: A Neighborhood of Infinity
- /r/haskell, #haskell on IRC

Tired of feeling like this?



Try Haskell!