**START OF THE BEGINNER SECTION**

**(https://github.com/input-output-hk/haskell-course)**

**1. Intro and tools**

* Intro to the course and lectures
  + What we’ll cover
  + Repository structure
* Intro to Haskell
  + How to open and use JupyterLab
  + Purely functional programming language
  + Basic syntax
  + Haskell Type system
  + Laziness
  + GHC (and GHCi)

**2. Data types, Signatures, and Polymorphism**

* Pragmatic intro to types
* Type signature
  + Function’s signatures
  + Variables in Haskell
    - Parameters in functions
    - Names/Definitions
* Infix and prefix functions
* Data Types in depth
  + Int, Integer
  + Float, Double
  + Rational
  + Bool
  + Char
  + Lists
  + Strings
  + Tuples + Tuples VS Lists
* Polymorphic values and type variables

**3. Conditions and helper constructions**

* If-then-else
* Guards
* let expressions
* where
* Should I use let or where?
* Things to keep in mind

**4. Pattern matching and Case**

* What is pattern matching
* Pattern matching on
  + Function implementations
  + Lists
  + Tuples
* Case

**5. Improving and combining functions**

* Higher-order functions
  + The filter function
  + The any function
* Lambda functions
* Precedence and associativity
* Curried functions
  + Partial application
* Composing and applying functions
  + The $ operator
  + The . operator
* Point-free style

**6. Recursion**

* Why Recursion?
* Thinking Recursively
  + sum and product
* Steps to create your own recursive function
* Examples of recursion
  + and, length, reverse, drop, take, map, filter
* Extracting the foldr pattern
* The foldl function
* The foldl' function
* When to use foldr, foldl, and foldl'

**7. Intro to Type Classes**

* The awesomeness of type classes
* What are type classes
* Common type classes
  + Eq, Ord
  + Num, Integral, Floating
  + Read, Show
* The most general valid type
* Multiple constraints

**8. Creating Types**

* Type synonyms
  + How to define type synonyms
  + Why use type synonyms
* Defining new types
  + data
  + Value parameters
  + Pattern matching types
  + Record syntax
* Parameterizing types
  + Parameterizing type synonyms
  + Parameterizing new types
* Honorable mention of newType

**9. Creating Type Classes and Instances**

* Revisiting Type Classes
* The Eq type class
  + Defining the Eq type class
  + Defining an instance for the Eq type class
  + Improving our Eq type class (minimal complete definition)
  + Defining an instance for a parameterize type.
* The Ord type class
  + Exploring Ord type class (Subclassing)
* Deriving
* Complete example

**10. Basic IO**

* We need side effects
* What is IO
* main + putStrLn + composing other functions
* >>
* >>=
* do notation
  + do
  + <-
  + let
* Some examples
* Read/Write to console
* Read/Write to file

**11. Pragmas, Modules, and Cabal**

* Prelude
* pragmas/extensions
* Overview of base modules
* Importing base modules
* A few modules
  + Data.Char
  + Data.Tuple
  + Data.Array
* Creating our own modules
* Cabal
  + What is it and why we use it
  + Cabal file
  + Using external libraries with Cabal

**12. Bits and Bytes**

* Grouping bits and bytes
* Haskell and bytes
* Lazy and strict byte strings
* Example

**13. Maybe and Either (only practical use)**

* Maybe
  + Why and when to use Maybe
  + Syntax
  + Examples
* Either
  + Why and when to use Either
  + Syntax
  + Examples
* Project using Maybe and IO

**14. Learning on your own and Map**

* Using GHCi to find out more
* Hoogle
* HaskellWiki
* Walking through while teaching Map module

**END OF THE BEGINNER SECTION**

**Congratulations!  You can call yourself a (beginner) Haskell programmer!**

**YOU'RE READY FOR THE BEGINNER PLUTUS AND MARLOWE PIONEER PROGRAM!  (Keep going for Plutus.)**

**START OF THE INTERMEDIATE SECTION**

**15. Monoid**

* Basic idea (definition without details)
* Intuitive examples
* Extracting the pattern
* Complete definition (with all the details/laws)

**16. Functor**

* Basic idea (definition without details)
* Intuitive examples
* Extracting the pattern
* Complete definition (with all the details/laws)

**17. Applicative**

* Basic idea (definition without details)
* Intuitive examples
* Extracting the pattern
* Complete definition (with all the details/laws)

**18. Aeson**

* Aeson

**19. Monad**

* Basic idea (definition without details)
* Intuitive examples
* Extracting the pattern
* Complete definition (with all the details/laws)
* do notation in general

**20. Reader Monad**

* Incentive/Motivation
* Binding strategy
* Definition
* Examples

**21. Writer Monad**

* Incentive/Motivation
* Binding strategy
* Definition
* Examples

**22. State Monad**

* Incentive/Motivation
* Binding strategy
* Definition
* Examples

**23. Monadic functions / Operating with Monads**

* liftM
* sequence and sequence\_
* mapM and mapM\_
* filterM
* foldM

**START OF THE INTERMEDIATE SECTION**

**Congratulations!  You can call yourself a (COMPLETE KNOWLEDGE SELF GUIDED) Haskell programmer!**

**YOU'RE READY FOR THE INTERMEDIATE PLUTUS AND MARLOWE PIONEER PROGRAM!**