Reading
You should read (if you haven’t already) the first three chapters of *Introduction to Functional Programming using Haskell* (IFPH).

Logistics
This homework is officially due in class on **Wednesday, February 12**. However, it comes with an automatic extension: anything submitted by **5pm on Friday, February 14** will be accepted as being on time.

You should work **alone** on this assignment. You should turn in a hard copy of your source code and a transcript demonstrating convincingly that your code is correct. For this assignment, you do not need to submit your code electronically.

Exercises (due Wednesday, February 12)

1. Consider the following definitions:

   ```haskell
   comp2 :: (a -> b) -> (b -> b -> c) -> (a -> a -> c)
   comp2 f g x y -> g (f x) (f y)
   succ :: Int -> Int
   succ x = x + 1
   f eq :: Float -> Float -> Bool
   f eq x y = (x == y)
   ```

   (a) Using these definitions, give a calculation of the Haskell expression
   \((\text{comp2} \ \text{succ} \ (*) \ 3 \ 4)\).

   (b) What is the most general type of the function \(\lambda h \rightarrow \text{comp2} \ h \ f e q\) ?

2. Write a Haskell function

   ```haskell
   summation :: (Integer -> Integer) -> (Integer -> Integer -> Integer)
   ```

   such that \(\text{summation} \ f\) returns a function that, when given \(j\) and \(k\), returns the summation \(\sum_{i=j}^{k} f(i)\).

   For example, \(\text{summation} \ (\lambda y \rightarrow y*y)\) should return the function that, when given \(j\) and \(k\), returns \(\sum_{i=j}^{k} i^2\).

3. Create a Haskell type class called *MyClass* with the following membership requirements:

   ```haskell
   size :: a -> Int
   combine :: a -> a -> a
   next :: a -> a
   ```

   You should also include a fourth function (of your own choosing) to the *MyClass* requirements, and give it a default definition based on some subset of the other requirements.

   Next, introduce each of the following types as instances of *MyClass* (all of them should use the default definition of your fourth function, except for *Char*):

   ```haskell
   -- Examples of instances of MyClass
   ```
(a) **Bool**
   Here, `size` should always return 1; `combine` should be the boolean-and function; and `next` should return `True` for `False` and vice versa.

(b) **Integer**
   Here, `size` should return the number of digits necessary for representing the number (e.g., `size 10546` returns 5, and `size -340170` returns 6); `combine` should be addition; and `next` should be the “increment by 1” function.

(c) **Char**
   Here, `size c` should return the ASCII number for `c`; `combine c1 c2` should return the character associated by the (mod 255) sum of the numbers associated with `c1` and `c2`; and `next c` should return the successor (mod 255) of `c`.

(d) **(a,b)**, where `a` and `b` are both instances of MyClass.
   Here, `size` should compute the sum of the individual sizes of the components, while `combine` and `next` should work componentwise on their arguments.