1 Haskell

(a) Install the Haskell Platform, via [https://www.haskell.org/platform/](https://www.haskell.org/platform/).

(b) Get familiar with Haskell. Take a look at [http://www.seas.harvard.edu/courses/cs152/2015sp/resources.html](http://www.seas.harvard.edu/courses/cs152/2015sp/resources.html) for some links to tutorials. In particular, get comfortable doing functional programming in Haskell. Write the factorial function. Write the append function for lists.

(c) Get comfortable using monads, and the bind syntax. Try doing the exercises at [https://wiki.haskell.org/All_About_Monads#Exercises](https://wiki.haskell.org/All_About_Monads#Exercises) (which will require you to read the previous sections to understand do notation, and their previous examples).

(d) Also, look at the file [http://www.seas.harvard.edu/courses/cs152/2015sp/sections/haskell-examples.hs](http://www.seas.harvard.edu/courses/cs152/2015sp/sections/haskell-examples.hs), which includes some example Haskell code (that will likely be covered in Section).

2 Algebraic structures

(a) Show that the option type, with map defined as in the lecture notes (Lecture 16, Section 2.2) satisfy the functor laws.

(b) Consider the list type, \( \tau \text{list} \). Define functions return and bind for the list monad that satisfy the monad laws. Prove that they satisfy the monad laws.

3 Concurrency

(a) Consider the following program.

\[ (3 + 7) || ((\lambda x. x + 1) 2) \]

Show an execution sequence for this program (i.e., give a sequence of expressions such that \( e_0 \rightarrow e_1 \rightarrow \ldots \rightarrow e_n \) where \( e_0 = (3 + 7) || ((\lambda x. x + 1) 2 + 5) \) and \( e_n \) is a value.

Now give a different execution sequence for this program.

How many different execution sequences of this program are there?

(b) Consider the following program.

\[
\text{let foo = ref 2 in (let y = (foo ::=!foo+!foo || foo ::=!1) in !foo)}
\]

What are the possible final values of the program?
4 Type and effect system

Recall the type and effect system to ensure determinacy, covered in Lecture 17.

(a) Consider the program (from class) of a bank balance, where the bank balance is in the region $A$.

   \[
   \text{let } \text{bal} = \text{ref}_{\alpha} A 0 \text{ in } (\text{let } y = (\text{bal} :=! \text{bal} + 25 || \text{bal} :=! \text{bal} + 50) \text{ in } ! \text{bal})
   \]

   Try to produce a typing derivation for this program (using the type-and-effect typing rules from lecture). Where do the typing rules fail? Why?

(b) Write a program that allocates two locations (in different regions) and reads and writes from both of them. Moreover, make sure that your program is well-typed according to the type-and-effect system. Is your program deterministic?