1. Write down a sequence of reduction steps reducing the term $(\lambda x.x(xy))(\lambda u.u)$ to normal form. In this problem reduction is allowed anywhere in a term, including under a λ.

2. Define functions in *untyped* lambda calculus for:

   (a) Identity: define the identity function id that simply returns its argument

   (b) Composition: define a function *compose* that takes two functions f and g as arguments and returns the function $f \circ g$.

   (c) Fibonacci: define a function fib that computes the nth Fibonacci number, F_n, where:

$$\begin{aligned} F_0 &= 1 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2}, n > 1 \end{aligned}$$

   You may assume that integers, integer addition, subtraction, and test for zero are defined, but you must use the combinator fix given on page 68 of Pierce to accomplish recursion (i.e. recursion is not natively provided by the language).

3. Repeat 2a and 2b in the simply typed lambda calculus F_1. Assume that there are only 2 types: Booleans and Integers. What goes wrong?

   Hint: try this expression from the untyped lambda calculus where id is the identity function from 2a and inc is the integer increment function: $\lambda i.i + 1$

$$(idinc)(id3)$$

   What about the following generalization of the above?

$$(\lambda f.\lambda y.\lambda z.((f\, y)\, (f\, z)))\, id\, inc\, 3$$

4. Give a type derivation in F_1 for: $y : T \to T \vdash (\lambda x.x(xy))(\lambda u.u) : T \to T$.

5. Repeat 2a and 2b in F_2. Give types for your answers.

6. Type safety is often formulated as the conjunction of two properties: progress and preservation. *Progress* means that a well-typed term is either a primitive value or can be reduced by a step. *Preservation* means that well-typed terms stay well-typed after being reduced by a step. Sketch a proof of progress and preservation for F_1, enriched with a unit type and Booleans.