1. (12 points) Reduce the following λ-calculus expressions as much as possible using only β-reductions. If the reduction does not converge, write → divergent.

   (a) \((\lambda x.\lambda y.y\ 22\ 5)\)

   (b) \((\lambda x.x)(\lambda y.y\ 0)\)

   (c) \((\lambda w\lambda z.z + w\) 97\)

   (d) \((\lambda a.\lambda b.a)(\lambda a.\lambda b.b)(\lambda x.x)\)

   (e) \((\lambda x.x\ x\ x)(\lambda x.x\ x\ x)\)

   (f) \((\lambda a.(\lambda b.b\ b)(\lambda c.c\ c))\)

2. (4 points) Prove that the following λ expressions are semantically equivalent:

   (a) \((\lambda x.\lambda y.x\ +\ y\) 44 and \((\lambda b.44 + b)\)

   (b) \(\lambda x.\lambda y.y\) and \(\lambda a.\lambda b.b\)

3. (8 points) In lecture we saw that it is possible to encode recursion in λ-calculus using fixed-point operators. We also studied one such operator, the Y-combinator.

   After learning about the Y-combinator in lecture, a student in CS312 proposes the following “simpler” fixed-point operator:

   \(\lambda y.(\lambda x.y\ (x\ x))\)

   Recall that any fixed-point operator must have the property that \(v = h(v)\) for any function \(h\). Is the proposed construct a fixed-point operator or not? Prove your answer.

4. In this question you will write three different programs that compute the factorial of 5 in L:

   (a) (2 points) Write a program in L that computes the factorial of 5

   (b) (3 points) Write the same program without using the function definition

   (c) (6 points) Write the same program without function definitions or let bindings (hint: the Y-combinator may be useful)

5. (5 points) Write a function in L that, applied to a list, returns the length of this list. For example your function should return "2" for the list 3305, 0 for the list Nil and 3 for the list "cs312"@"is@"fun"

6. (10 points) Write a function in L that reverses a list. Be careful! The reversal of
[1, [2, 3]]

is [3, [2, 1]], not [[3, 2], 1].

*Hint:* You will probably first need to write a list concatenation helper function.