**Assignment 2**

1. Language variations:
   1. No, every program in L1 can be transformed into an equivalent program in L11. The define form is simply syntactic sugar for naming values or functions. Therefore, we can replace each use of a defined name with its actual value or expression, and the behavior of the program will remain the same.
   2. Yes, there are programs in L2 that cannot be transformed into equivalent programs in L21 without changing their behavior. This is true for cases involving recursion. In L2, define is used not only for defining variables but also for enabling giving names to functions, which cannot be expressed using only lambda expressions.
   3. לחינח
   4. Yes, there are programs in L2 that cannot be transformed into equivalent programs in L23. As we studied in the lectures, L2 is Turing complete, so it allows the implementation of higher-order functions like map, which can take a lambda and a list, and apply the lambda to each element (as we did in L3). However, in L23, where functions cannot be passed as arguments, it is not possible to implement map in a general way. Therefore, programs that rely on passing functions as arguments such as map cannot be expressed in L23.
2. Let extend L3 syntax to support dictionaries in multiple ways:
   1. As a primitive operators:
      1. We will add to <prim-op> the following:
         1. dict | dict? | get
   2. As a special form:
      1. We will add new form <dict-entry>:
         1. <dict-entry> ::= (symbol <cexp>) / DictEntry(key:symbol, val: CExp)
      2. We will add to <cexp> the following:
         1. (dict (<dict-entry>)\*) / DictExp(entries:DictEntry[])
         2. (get <cexp> <cexp>) / DictRefExp(dict: CExp, key: CExp)
   3. As a L3 user procedures.
   4. Theoretical answers:
      1. Q: Should your implementations for the three dictionary versions(2.1, 2.2, 2.3) be modified for the case of normal order?

A: