1. [10 points] Describe the three phases of programming language analysis. Use an example of what the input and output of each phase is.

2. [10 points] Give two reasons why programming with pure functions can be beneficial. Explain your answer using a function that is not pure.

   a. What does it mean for a language to be **dynamically typed**? Explain your answer using a concrete example from Scheme that causes a type error.
   b. What does it mean for a language to be **statically typed**? Explain your answer using a concrete example from Java that causes a type error.

4. [10 points] Draw the linked structure for the Scheme list
   \[(\text{list 1 (list 3 4) 5}).\]

5. [10 points] Explain the difference between linear and iterative recursion. Address the time and space requirements of each.

6. [10 points] Write a Scheme predicate `forall` that takes a predicate `p` and a list `L` and returns true if every item in `L` satisfies `p`. `forall` should be iteratively recursive. Do not use `map`, `fold`, or `filter` to implement `forall`. Here is an example of how `forall` should be called `(forall even? (list 1 2 3 4 5))`

7. [10] Write `forall` using `map` and `fold`. The interface should be the same as in question 6.

8. [10 points] Write a Scheme function `leaves` that returns the number of leaves in a binary tree, where binary trees are represented like we did in class. You can assume `left`, `right`, `root`, and the predicate `leaf?` are defined for you.

9. [10 points] Consider a list of numbers such as `(list 1 2 3 4 5)`. Write a Scheme expression that computes \[\sum_{i=1}^{n} (x_i - 1)^2 + 1\] where \(x_i\) refers to the \(i^{th}\) item in the list. Full credit for being concise (e.g., `map`, `fold`, etc.) and is space and time efficient at runtime.

10. [10 points] Write a Scheme function `fibs` that takes an integer parameter `n` and returns a list of the first `n` fibonacci numbers in increasing order. For example,
   \[\text{(fibs 3) is (list 1 1 2) and (fibs 10) is (list 1 1 2 3 5 8 13 21 34 55)}\]
   The function `fibs` must compute the list efficiently.