1. Problem 1 (20 Points)
Languages that employ a reference model of variables also tend to employ automatic garbage collection. Is this more than a coincidence? Explain.

2. Problem 2 (20 Points)
Most binary arithmetic operators are left-associative in most programming languages. However, most compilers are free to evaluate the operands of a binary operator in either order. Are these statements contradictory? Why or why not?

3. Problem 3 (30 Points)
Consider the following pseudocode:

```plaintext
x : integer – global

procedure set x(n : integer)
    x := n

procedure print x
    write integer(x)

procedure first
    set x(1)
    print x
```
procedure second
    x : integer
    set x(2)
    print x

main()
    set x(0)
    first()
    print x
    second()
    print x
end

What does this program print if the language uses static scoping? What does it print with dynamic scoping? Why?

4. Problem 4 (20 Points)
Why don't we use

\[ PP[\text{<program C>}] = CC[C] \circ (\lambda(m,i,o).o) \]

instead of the definition of \( PP[\text{<program C>}] \) on PPT 26, slide 23?
Explain.

5. Problem 5 (40 Points)
Modify the attribute grammar for Tiny, so that "read" is no longer an intrinsic function, but instead a statement.

Instead of

\[ \text{assign } i := \text{read}; \]

we would have

\[ \text{read}(i); \]

The read statement would handle exactly ONE identifier.
6. Problem 6 (40 Points)
Add a 'swap' statement to Tiny. It would look like this:

   swap i :=: j;

Assume a parser ensures i and j are names, not expressions.
Also, assume they are type compatible.

7. Problem 7 (30 Points)
Build the functional graph for the binary number 101.11, according to the attribute grammar for
binary numbers developed in class.