Language Constructs

1. **Primitives**: simplest entities in the language
   . evaluate to themselves
   examples:

   . evaluate to a procedure
   examples:

2. **Combinations**: compound elements built by combining smaller ones (primitive procedures and subexpressions)
   (foo a b c)  First expression after left parenthesis must be a procedure to be applied; a, b, c are subexpressions representing the procedure's arguments
   . evaluate subexpressions, then apply value of the operator

   (+ 3 4)

   (+ (+ 3 4) (+ 10 11) (+ 1 1))

3. **Abstractions**: compound elements can be named and used as single entities
   . needs a special form called `define`  (why?)

   (define bar 4)
   (define foo +)
   (foo bar 3)

   (define foo*2 (* foo 2))
   (define foo*2 (* (foo 3 4) 2))
Examples

(* 5 99)

(+5 99)

(* (5 9))

(* -5 99)

(* (- 5 99))

What special characters have we seen so far in Scheme?

Problems

What is the result printed by the Scheme interpreter for each expression? Assume that the first 7 expressions are evaluated in order.

1. 42

2. (/ 5 2)

3. (+ (* 2 3) (* -4 8))

4. +

5. (define + (* 2 5))

6. (* 2 +)

7. (+ 2 5)

8. Write the Scheme expression representing the following (assume that + has not been redefined):

\[
\frac{5 + 4 + (2 - (3 - (6 + \frac{1}{4})))}{3(6 - 2)(2 - 7)}
\]