Haskell Expression Grouping (a.k.a. “under-chunking!”) Exercises
Intro to Functional Programming • WU CS 154 • Fritz Ruehr

The following exercises are intended to give you some practice in finding the “boundaries” between parts of a Haskell expression—in other words, which functions group with which arguments, etc. Recall that the rules include (roughly speaking):
- a function is placed before its (possibly several) arguments;
- arguments which are themselves “structured” or “non-atomic” (i.e., built from pieces) must be enclosed in parentheses;
- prefix applications (“function-in-front”) group before infix operators (like + and *);
- between different infix operators, the usual “order of operations” applies (* and / before + and -);
- and the parts of an if-the-else expression group last.

All these rules can be “over-ridden” by the use of parentheses—in fact, you can always use “un-required” parentheses for emphasis, or if you’re not sure you’ll get what you mean. Finally, note that grouping or “chunking” tells how expressions are put together—but not “when” they are evaluated.

For each of the expressions below, draw lines underneath the parts of the expression to group it the way Haskell would interpret it, according to the rules above. For example, you might draw lines to group like this:

\[
f ( g x + 2 * h ( f x ) )
\]

1. \( f \ y ( g x + 3 * 7 ) + x * 7 \)

2. \( 25 * f o o ( x + f y * 3 ) + 1 \)

3. \( 2 * f ( i f x < 5 t h e n y e l s e x - 3 ) * z + g y \)

4. \( i f x < 0 t h e n i f y > 3 t h e n 3 e l s e f x e l s e 2 \)

5. \( i f i f b & & c t h e n x < 5 e l s e T r u e t h e n y + 1 e l s e x - 2 \)

Warning! Answers on next page!
Answers:

1. \( f \ y \ ( g \ x + 3 \times 7 ) + x \times 7 \)

   Nothing too special here—should be an easy exercise.

2. \( 25 \times \text{foo} \ ( x + f \ y \times 3 ) + 1 \)

   A little tricky in the middle there near the “f y” part—note that “foo” is a function of the parenthesized arguments.

3. \( 2 \times f ( \text{if} \ x < 5 \ \text{then} \ y \ \text{else} \ x - 3 ) \times z + g \ y \)

   Whew! This is a long one—the “if-then-else” part really does need to be parenthesized or Haskell will complain!

4. \( \text{if} \ x < 0 \ \text{then} \ \text{if} \ y > 3 \ \text{then} \ 3 \ \text{else} \ f \ x \ \text{else} \ 2 \)

   Well, that one was just mean! But you are allowed to put one “if-then-else” inside another like that—try it!

5. \( \text{if} \ \text{if} \ b \ \&\& \ c \ \text{then} \ x < 5 \ \text{else} \ \text{True} \ \text{then} \ y + 1 \ \text{else} \ x - 2 \)

   OK, this one’s even more cruel: this is an “if-then-else” nested in the “if” part of another one!