Preconditions and Postconditions

- Exercises
  - Exercise 0: Preparation
  - Exercise 1: Are they all real?
  - Exercise 2: Differentiating Between Errors
  - Exercise 3: When Can You Count Between?
  - Exercise 4: An Odd Factorial
  - Exercise 5: Finding Vvalues
  - Exercise 6: Substitution

Exercises

Exercise 0: Preparation
Start DrScheme.

Exercise 1: Are they all real?

a. Write the \texttt{all-real?} procedure described in the accompanying reading.

b. What preconditions should \texttt{all-real?} have?

c. Is it necessary to test those preconditions? Why or why not?

Exercise 2: Differentiating Between Errors
Revise the definition of \texttt{greatest-of-list} given in the corresponding reading so that it prints a different (and appropriate) error message for each error condition.

I’d recommend that you use \texttt{cond} rather than \texttt{if} in writing this revised version.

Exercise 3: When Can You Count Between?
Revise the definition of the \texttt{count-from} procedure presented in the reading on recursion with natural numbers so that it enforces the precondition that its first argument be less than or equal to its second argument.
Exercise 4: An Odd Factorial

Here is a procedure that computes the product of all of the odd natural numbers up to and including number:

\[
\text{(define odd-factorial}
\]
\[
\text{(lambda (number)}
\]
\[
\text{(if (= number 1)}
\]
\[
\text{1)}
\]
\[
\text{(* number (odd-factorial (- number 2))))))}
\]

a. What precondition or preconditions does odd-factorial impose on its argument?

b. What will happen if these preconditions are not met?

c. Revise the definition of odd-factorial as a husk-and-kernel program in which the husk enforces the precondition.

d. How can we be certain, in this case, that none of the recursive calls we make to the kernel procedure violates the precondition?

Exercise 5: Finding Vvalues

a. Define and test a procedure named index that takes a symbol sym and a list ls of symbols as its arguments and returns the index of sym in ls. You should use 0-based indices (so that the initial value in a list is at index 0).

\[
> \text{(index 'gamma (list 'alpha 'beta 'gamma 'delta))}
\]
\[
2
\]
\[
> \text{(index 'easy (list 'easy 'medium 'difficult 'impossible))}
\]
\[
0
\]
\[
> \text{(index 'the (list 'and 'the 'cat 'sat 'on 'the 'mat))}
\]
\[
1
\]

b. Arrange for index to signal an error (by invoking the error procedure) if sym does not occur at all as an element of ls.

Exercise 6: Substitution

Define and test a procedure named substitute that takes three arguments -- a symbol new, another symbol old, and a list ls of symbols -- and returns a list just like ls except that every occurrence of old has been replaced with an occurrence of new. Use the husk-and-kernel structure to make sure that new and old are symbols and that ls is a list of symbols before starting into the recursion.

\[
> \text{(substitute 'alpha 'omega (list 'phi 'chi 'psi 'omega 'omega))}
\]
\[
\text{(phi chi psi alpha alpha)}
\]
\[
> \text{(substitute 'starboard 'port (list 'port 'starboard 'port 'port))}
\]
\[
\text{(starboard starboard port port)}
\]
\[
> \text{(substitution 'in 'out null)}
\]
\[
\text{()}
\]