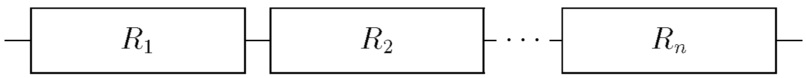
3. An electrical engineer often needs to find the *total resistance* of a part of an electrical circuit. In general, circuits can contain resistors, capacitors, inductors, *etc.* This example will only consider segments of a circuit that contain combinations of resistors.

   Here is the informal description of the inputs that your programs will process. A *circuit segment* can be either a resistor, a serial segment, or a parallel segment.

   A *resistor* has only one interesting property: a resistance, which is a real number, greater or equal to `0.0`, or the symbol `'infinity`.

   A *serial segment* consists of an arbitrary number of segments connected in a row:



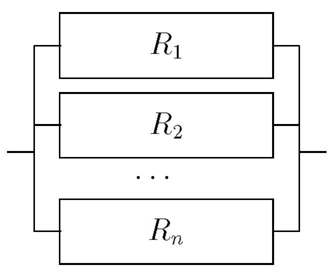
   The total resistance of a serial segment is the sum of the individual resistances. In other words, the entire segment behaves like a single resistor with that total resistance:

$$R_{seg} = \sum_{i=1}^{n} R_i$$

   If one of the segments in the series has infinite resistance, then

$$R_{seg} = \ \mathsf{'infinity}.$$

   A parallel segment consists of an arbitrary number of segments that are connected to a common starting and ending point:



   The total resistance of a parallel segment is the inverted sum of the inverted resistances of the segments:

$$R_{seg} = \frac{1}{\sum_{i=1}^{n} \frac{1}{R_i}}$$

   If one of the segments has no resistance, then the total resistance is just `0`; if one of the resistors has infinite resistance, that resistor contributes nothing to the sum of inverses.[1] If all of the resistors have infinite resistance, i.e., the sum above is 0, then the whole parallel segment has infinite resistance.

[1]Inverse refers to *multiplicative* inverse of a number, i.e., the inverse of n is $\frac{1}{n}$.

(a) Write up a rigorous data description for circuit segments and resistances using the syntax described in class.

Create appropriate predicates, selectors and constructors for your new data types using **define-structure**.

Define templates for segment-processing and resistance-processing functions.

(b) Derive the functions *plus*, *times*, and *inverse*, which accept two resistances and compute the following results:

- *plus* computes the sum of two resistances. If one is 'infinity, the sum is 'infinity, too.
- *times* computes the product of two resistances. If one is 'infinity, the product is 'infinity, too.
- *inverse* computes the inverse of a resistance. If the resistance is 0, the result is 'infinity; if it is 'infinity, the result is 0; and otherwise, it is the inverse of the number.

Evaluate the expression (*inverse* (*plus* (*inverse* 'infinity) 10)) by hand.

(c) Derive the program *resistance*, which accepts a circuit segment and computes the total resistance of the segment. Hints: (1) The program consists of many functions. (2) Some functions need to compute arithmetic expressions involving resistances. Use the functions *plus*, *times*, and *inverse* instead of $+$, $*$, and $/$. (3) Test every function individually before you test the complete program.