

Procedures That Return Multiple Values

- Exercises
 - Exercise 0: Preparation
 - Exercise 1: Testing `mixed-number-parts`
 - Exercise 2: Computing with Multiple Results
 - Exercise 3: Separating A Number's Parts
 - Exercise 4: Divide
 - Exercise 5: Tallying By Parity
 - Exercise 6: Separating Association Lists

Exercises

Exercise 0: Preparation

- a. Scan through the reading on procedures that return multiple values. Make sure you understand the standard Scheme procedures `values` and `call-with-values`.
- b. Start DrScheme.

Exercise 1: Testing `mixed-number-parts`

What are the results of applying `mixed-number-parts` to each of the following values? Explain each result.

- $17/5$
- $4/7$
- $40/2$
- $-2/5$
- $-10/3$
- 1.3
- -0.5

Exercise 2: Computing with Multiple Results

- a. What do you expect to happen when you add 2 to the result of `(mixed-number-parts 7/5)`? Try it and see.
- b. What do you expect to happen when you take the car of the result of `(mixed-number-parts 7/5)`? Try it and see.

Exercise 3: Separating A Number's Parts

Using `call-with-values` and `mixed-number-parts`, write `(whole-part rat)`, which gives the whole part of a rational number, and `(frac-part rat)`, which gives the fractional part.

Exercise 4: Divide

Write the `(divide dividend divisor)` procedure mentioned in the reading. Your procedure should return two values: (1) a number, q , such that $q*divisor$ is less than or equal to dividend and $(q+1)*divisor$ is greater than dividend; (2) the remainder.

Hint: Think about the strategy used in `mixed-number-parts`.

Exercise 5: Tallying By Parity

Define and test a procedure `tallies-by-parity` that takes any list of integers as its argument and returns two values, the number of even integers in the list and the number of odd integers in the list.

Hint: Use tail recursion.

You may not use `partition` in your solution.

- a. Solve the problem without using `call-with-values`.
- b. Solve the problem with `call-with-values`.

Exercise 6: Separating Association Lists

Define and test a recursive procedure that takes an association list `als` as argument and returns two results: a list of the keys of `als`, and a list of the values of `als`.