Laboratory: Pairs and pair structures

- **Exercises**
  - Exercise 0: Preparation
  - Exercise 1: Some pictures
  - Exercise 2: Some pairs
  - Exercise 3: More pictures
  - Exercise 4: Are they pairs?
  - Exercise 5: Counting pairs
- **Notes**
  - Notes on Exercise 5

### Exercises

#### Exercise 0: Preparation

Start DrScheme.

#### Exercise 1: Some pictures

Draw box-and-pointer diagrams for each of the following lists:

- \((x)\ y\ z\)
- \((x\ (y\ z))\)
- \(((a)\ b\ (c\ ())))\)

#### Exercise 2: Some pairs

Enter each of the following expressions into Scheme. In each case, explain why Scheme does or does not use the dot notation when displaying the value.

- \((\text{cons}\ 'a\ "Walker")\)
- \((\text{cons}\ 'a\ \text{null})\)
- \((\text{cons}\ \text{null}\ 'a)\)
- \((\text{cons}\ \text{null}\ (\text{cons}\ \text{null}\ \text{null}))\)

#### Exercise 3: More pictures

Draw a box-and-pointer representation of the value of each expression in the previous exercise.
Exercise 4: Are they pairs?

What do you think that `pair?` will return for each of the following? How about `list?`. Confirm you answer experimentally and explain any that you found particularly tricky.

- `(cons 'a 'b)`
- `(cons 'a (cons 'b 'c))`
- `(cons 'a null)`
- `null`
- `(list 'a 'b 'c)`
- `(list 'a)`
- `(list)`

Exercise 5: Counting pairs

Define and test a procedure named `cons-cell-count` that takes any Scheme value and determines how many boxes would appear in its box-and-pointer diagram. (The data structure that is represented by such a box, or the region of a computer’s memory in which such a structure is stored is called a `cons cell`. Every time the `cons` procedure is used, explicitly or implicitly, in the construction of a Scheme value, a new cons cell is allocated, to store information about the car and the cdr. Thus `cons-cell-count` also tallies the number of times `cons` was invoked during the construction of its argument.)

For example, the structure in the following box-and-pointer diagram contains seven cons-cells, so when you apply `cons-cell-count` to that structure, it should return 7. On the other hand, the string "sample" contains no cons-cells, so the value of `(cons-cell-count "sample")` is 0.
Use cons-cell-count to find out how many cons cells are needed to construct the list (0 (1 (2 (3 (4))))). Draw a box-and-pointer diagram of this list to check the answer.

**Notes**

**Notes on Exercise 5**

If, for some reason, you are having trouble creating the list

(0 (1 (2 (3 (4)))))

try

(list 0 (list 1 (list 2 (list 3 (list 4)))))