Scheme Programming

CSCI 312 Homework 2

December 22, 2022

File names: Names of files and variables, when specified, must be EXACTLY as specified. This includes simple mistakes such as capitalization.

Individual work: All work must be your own. Do not share code with anyone other than the instructor and teaching assistants. This includes looking over shoulders at screens with the code open. You may discuss ideas, algorithms, approaches, *etc.* with other students but NEVER actual code.

Requirements:

1. You must use natural recursion to solve each of these problems.
2. Do not use builtin procedures that do the bulk of the work for you. For instance, if required to write a function to append two lists, do not use `append`!
3. Do not use Racket's looping constructs such as `do` or `for`.
4. Do not use named `let`, either, even though this is a disguised recursive program.
5. Use `#lang plai`
6. Place all solutions in one file, named `csci312hw02`<*yourname*>`.rkt`
7. Include a comment block at the top with your name, class name, homework number.
8. Include simple tests for each procedure. With the plai language tests are as simple as:

```
(test (+ 2 2) 4)
```

9. You may write as many helper procedures as you like, in order to implement the requested procedure. They do not have to be nested.
10. Mark the beginning of each problem solution with a comment bar like these:

```
;;;; Problem 1 ;;;;;;;;;;;;;;;;;;;;;;;;;;;
(define insertR
    ...
;;;; Problem 2 ;;;;;;;;;;;;;;;;;;;;;;;;;;;
(define zip
   ...
```

All helper functions for the main procedure should be within the appropriate comment bars.

11. You may assume good input; you do not have to check for erroneous input.

Grading: Problems are worth a maximum of 5 points each.

- Solutions that are exceptionally well done get 5 points.
- Solutions that are simply correct get 4 points.
- Solutions that have small errors or missing tests get 3 points.
- Solutions with major errors get 1 point.
- Missing solutions or solutions with syntax errors get 0 points.

The problems:

1. Write and test a procedure that takes two symbols and a list of symbols and returns the list of symbols but with with the second symbol inserted to the right of each occurrence of the first symbol. Example:

```
(insertR 'a 'b '(a x a a y z a))  => '(a b x a b a b y z a b)
```

2. Write and test a procedure that takes two lists of symbols and returns a list of lists of symbols, each consisting of the two corresponding symbols from the two original lists. If one list is longer than the other, drop the tail of the longer one. Example:

```
(zip '(x y z) '(a b c d e))  => '((x a) (y b) (z c))
```

3. Write and test a procedure to take a symbol and a list of symbols and return the 0-based indices of every occurrence of the symbol in the list. If there are no occurrences, return the empty list. Example:

```
(indices 'x '(a b x c d e x f x))  =>  '(2 6 8)
```

4. Write and test a procedure that takes a list of alternating integers and symbols, and returns a list with each symbol repeated the integer number of times. Example:

```
(repeater '(3 x 2 y 5 z))  =>  '(x x x y y z z z z z)
```

5. Write and test a procedure that takes two lists of symbols and returns the *Cartesian product* of the two lists. That is, a list of lists, each of which is a list of one item from the first list and one item from the second. All possible pairs are included, exactly once, but order does not matter. Example:

```
(product '(a b) '(x y z))  =>  '((a x) (a y) (a z) (b x) (b y) (b z))
```