CSE 1729 - Introduction to Principles of Programming — October 13, 2017

Problem Set 6

1. In lecture, we discussed an example function, `(remove v elements)`, that removed the first occurrence in `elements` that was equal to `v`.
   (a) For this problem, write the Scheme function `(remove-if f elements)` that returns the list with all elements x for which `(f x)` is true removed.
   (b) The lecture `(remove v elements)` function did not remove any elements nested in lists inside lists. Now define a function `(nested-remove v elements)` that will remove items from nested lists as well. You should remove all of the items in the list that are equal to the thing to be removed, using `equal?` as your equality test.

   To add to the challenge, you should also be able to remove lists from lists, for example

```
>(nested-remove 'b '(b 2 (a b)))
(2 (a))
>(nested-remove '(a b) '(1 2 (a b)))
(1 2)
```

2. Masahiko Fujiwara showed that 1458 is one of four positive integers (with the trivial case 1) which, when its digits are added together, produces a sum which, when multiplied by its reversal, yields the original number:

$$\begin{aligned} 1+4+5+8 &= 18 \\ 18 \times 81 &= 1458 \end{aligned}$$

   (a) Define a function named `(explode x)` which converts any positive integer x to a list of single-digit integers (use division by 10 and the floor function). For instance, the expression `(explode 12345)` would return `'(1 2 3 4 5)`. You may only use `cons`, `car`, `cdr`, functions you wrote in lab or are contained in the lecture slides. You may not use string or char functions.
   (b) Define a Scheme function named `(implode l)` which takes a list of digits in base 10 and converts them to an integer (the inverse operation of `explode`). You may only use `cons`, `car`, `cdr`, functions you wrote in lab or are contained in the lecture slides. You may not use string or char functions.
   (c) Write a function named `(has-property x)` which accepts an integer x and returns `#t` if x has this property and `#f` otherwise. Test your function with the integer 1458. Feel free to use the `sum-list` and `reverse` functions from lecture.
   (d) Use the `find` function from Lab 4 to identify two other integers with this property. *Hint: 1 is the first integer with this property and 1458 is the third. You are looking for the second and fourth integers with this property.*

3. *More fun with sets* Using an unordered list to represent sets (as in lab, and in Section 2.3.3 in the textbook), implement the following set functions. Feel free to define appropriate helper functions.

   (a) `(superset? a b)` Returns `#t` if set `a` is a superset of set `b`, that is, every element of set `b` is a member of set `a`; `#f` otherwise.

   (b) `(set-difference a b)` Returns the set-difference between set `a` and set `b`. This difference, denoted $a - b$, is defined as

   $$a - b = \{x : (x \in a \wedge x \notin b)\}$$

   (c) `(cross-product a b)` Returns the set of tuples (x, y) such that x is in a and y is in b. Use two-element lists for tuples. This is denoted $a \times b$, and is defined as

   $$a \times b = \{(x, y) : x \in a \wedge y \in b\}$$

4. Define a Scheme function, `(nestless lst)`, that takes a list and returns a list with the same elements but no nesting. For example:

```
>(nestless '((a b) c (d e (f g))))
(a b c d e f g)
>(nestless '(a b))
(a b)
>(nestless '())
()
```

   You can use the built-in Scheme function `pair?` to determine if an element is a pair (possibly representing a list) or an atomic value.

5. Define a Scheme function, `(merge l1 l2)`, which takes two lists ℓ_1 and ℓ_2 as arguments. Assuming that each of ℓ_1 and ℓ_2 are *sorted* lists of integers (in increasing order, say), `merge` must return the sorted list containing all elements of ℓ_1 and ℓ_2.

   To carry out the merge, observe that since ℓ_1 and ℓ_2 are already sorted, it is easy to find the smallest element among all those in ℓ_1 and ℓ_2: it is simply the smaller of the first elements of ℓ_1 and ℓ_2. Removing this smallest element from whichever of the two lists it came from, we can recurse on the resulting two lists (which are still sorted), and place this smallest element at the beginning of the result.

Submitting Solutions

Once you have finished:

1. Save your work to a file using the appropriate filename, problemSet6.rkt.
2. Submit your solution file for grading via the MIMIR site.