CMPSC 461: Programming Language Concepts

Spring 2020

Programming assignment 2: Scheme Programming
Total: 24 points. Due on Mar 28th at 6pm in Canvas.

1. (3 points) Write a higher-order function `dncall` that takes three parameters: n, f, x; it returns x when $n = 0$, returns $f(f(x))$ when $n = 1$, returns $f(f(f(f(x))))$ when $n = 2$, etc. That is, it returns the result of calling f on x, for $2n$ times. For instance, invoking `dncall` with $n = 2$, the add-one function, and $x = 2$ should return 6.

2. (3 points) Write a Scheme `keep-if` function. It takes two arguments. The first is a boolean function f, and the second is a list l. The `keep-if` function returns a new list with all elements x in l such that $f(x) = \#t$ being kept, while those elements x such that $f(x) = \#f$ being removed.

   For example,

   ```
   (keep-if (lambda (x) (> x 3)) '(10 1 7 2))
   ```

   should return the list (10 7), since both 10 and 7 are greater than 3.

   Define the `keep-if` function by using case analysis and recursion.

3. (3 points) The least function takes a list of numbers, and returns its least element. For example, (least '(7 3 6 2)) should return 2. Write the least function in Scheme in two steps:

   (a) Write a least_helper function; it takes a number k and a list of numbers, x, as parameters, and returns the number which is smallest among k and numbers in x. For example, (least_helper 5 '(4 5 6)) should return 4.

   (b) Write the least function based on least_helper.

4. (5 points) Write a Scheme function `to-words` that takes an integer between -99 and 99, inclusive, as a parameter and returns a list of words corresponding to the integers reading in English. Make your function as short as possible; i.e., don't write a COND statement with 200 tests. Example calls to your function are given below:

```
(to-words 13) ; should return (thirteen)
(to-words 42) ; should return (forty two)
(to-words -55) ; should return (negative fifty five)
```

Note, in order to simplify things, we are not expecting dashes in the output (i.e., 42 is output as forty two not forty-two). Also, if the input number is outside of the range of [-99,99], your program should output 'error. Hint: You may want to use the built-in integer division functions `quotient` and `remainder`.

5. (10 points) In this question, we are going to develop a Scheme program that calculates *relevant word counts*. The input is two lists, the first is a list of words to be counted and the second is a list of irrelevant words; the output should be counts for those relevant words that appear in the first input list. For example, if the first input list is

   ```
   '(time is long but life is short)
   ```

   and the list of irrelevant words is

   ```
   '(but)
   ```

   Then the output should be

   ```
   '((short 1) (is 2) (life 1) (long 1) (time 1))
   ```

   Note that you are not asked to sort the words in the output. Therefore, the output is correct as long as the counts are correct.

   Do the following steps to implement the program. In our discussion, we call a word with a count a *word-count pair*, for example, `(short 1)` and `(is 2)` are word-count pairs. We call a list of word-count pairs a *word-count list*.

   (a) (2 points) Write a function `filterWords` that takes a list of words and a list of irrelevant words and returns an output list with irrelevant words filtered out. E.g., if the first input list is

   ```
   '(time is long but life is short)
   ```

and the list of irrelevant words is

```
'(but)
```

Then the output should be

```
'(time is long life is short)
```

You can use the "member?" function we discussed in class, if you find it helpful.

(b) (2 points) Write a function `iniWordCountList` that takes a list of words and creates a word-count list. The resulting word-count list should have the word count 1 for every word. Use the `map` function we discussed in class to implement `iniWordCountList`. For instance,

```
(iniWordCountList '(time is long life is short))
```

should generate

```
((time 1) (is 1) (long 1) (life 1) (is 1) (short 1))
```

(c) (2 points) Write a function `mergeWordCounts`. It takes two inputs. The first is a word-count pair and the second is a word-count list. This function generates a new word-count list. If the input word-count pair has a corresponding pair in the word-count list with the same word, then the counts should be merged; otherwise, the output word-count list should have the input word-count list with the word-count pair at the end of the list. For instance,

```
(mergeWordCounts '(is 1) '((time 1) (is 1)))
```

should generate

```
((time 1) (is 2))
```

As another example

```
(mergeWordCounts '(life 1) '((time 1) (is 2)))
```

should generate

```
((time 1) (is 2) (life 1))
```

(d) (2 points) Write a function `mergeByWord`, which takes a word-count list and produces a new word-count list; the output word-count list should have one word-count pair for each word that appears in the input list and the count should be the sum of all counts for that word in the input list. For instance, if the input list is

```
((time 1) (is 1) (long 1) (but 1) (life 1) (is 1) (short 1))
```

then the output should be

```
((short 1) (is 2) (life 1) (but 1) (long 1) (time 1))
```

Write `mergeByWord` based on the `reduce` function we discussed in class and `mergeWordCounts`. The `reduce` function is not built-in in Scheme; you can type it in yourself:

```
(define (reduce f l v)
  (if (null? l) v
      (f (car l) (reduce f (cdr l) v))))
```

(e) (2 points) Finally, write a `relevantWordCount` function that takes in a list of words and a list of relevant words, and outputs the right word-count list. Write this function based on `filterWords`, `iniWordCountList`, and `mergeByWord`.

Notes For programming assignments, examples are given only for the purpose of clarification. By no means that our tests will solely be based on those examples. It's your responsibility to thoroughly test your code by designing your own test cases.

Chapter 6 of the Scheme standard `http://www.schemers.org/Documents/Standards/R5RS/HTML/` contains a list of standard procedures provided by Scheme (for example, the remainder and the square root functions). You may find them helpful during your programming.

Submission format: Put all your code into one DrRacket file and submit your file through Canvas. Please clearly mark your answers using comments so that we can tell the correspondence between your code and questions. Please ensure that your file starts with a comment that includes your name and your Penn State network user id.