CS5001 Assignment 9

Programming Language: ISL with Lambda

Due Dates: Thursday 3/29 @ 10:00pm

Problem 1.

Here is a data definition for s-expressions:

```
;; An Atom is one of:
;; - Number
;; - Symbol
;; - String
;;
;; An SExp is one of:
;; - Atom
;; - [List-of SExp]
```

Here are the rules for the written, textual representation of an s-expression (that is, as a string):

- A number is represented as a base-ten numeral: `"107"`, `"-92"`.
- A symbol is represented with the characters of the symbol: `"foo"`, `"bar"`. (You may assume that symbols don't have odd characters, such as space, parentheses, quote characters, etc.)
- A string is represented with the characters of the string, delimited by a pair of double-quote characters: `"\"foo\""`, `"\"bar\""`. (Note that backslash-doublequote is how you put a doublequote character in a string. You may assume that strings don't have odd characters, such as double-quote characters themselves.)
- A list of s-expressions is represented by (1) a left parenthesis; (2) the items of the list, separated by one or more spaces; (3) a close parenthesis. For example, `"(a (37 \"foo\") c)"` is the string encoding of a three-element list, whose first element is the symbol `"a"`, whose second element is a two-element list (a number and a string), and whose third element is the symbol `"c"`.

a. Design a function (and any necessary helper functions) that will consume an s-expression and produce a string with the textual representation of that s-expression. For example:

```
(check-expect (sexp->string '(a (37 "foo") c)) "(a (37 \"foo\") c)")
```

Note:

- Your function can add a bit of extra whitespace before/after list elements if it makes it simpler; that's fine. For example, this is a fine three-element list of symbols: `"(a b c )"`.
- You may want to look up the useful functions `number->string` and `symbol->string`.

b. Develop the templates for a function that consumes two `SExp`s.

c. Design a program that will determine if two `SExp`s contain the same atoms regardless of the ordering. For example: `(contains-same-atoms? '(1 2 3 () ("r" b)) '("r" 1 (2) 3 b))` would return true. Do not solve this by flattening the `SExp`s into `[List-of Atom]` first.

Problem 2.

Consider the following data definitions:

```
; A Road is one of:
; - 'dead-end
; - (make-straightaway String PositiveNumber Road)
; - Intersection

(define-struct straightaway [name distance more])
; INTERPRETATION: A road with some name and some amount of distance
; until the next portion of road

; An Intersection is a [List-of Road]
```

a. Design the function `total-road-length` which takes a Road and produces the total length of it and all its connected roads. Use list abstractions where appropriate.

b. Design the function `road-names` which takes a Road and produces a list of all the names of roads connected to the given Road (including the name of the given Road itself if it has one). Use list abstractions where appropriate. You may assume that all road names are unique.