Homework: FuncLang (Part II)

Learning Objectives:

1. Functional programming
2. Understand and expand FuncLang interpreter

Instructions:

- Total points: 59 pt
- Early deadline: Oct 3 (Wed) 2018 at 6:00 PM; Regular deadline: Oct 5 (Fri) 2018 at 6:00 PM
- We will grade functional programming based on our tests
- Download hw5code.zip from Canvas
- Set up the programming project following the instructions in the tutorial from hw2 (similar steps)
- How to submit:
  - For questions 1–2, please submit one pdf or a zip file that contains all the source code
  - For questions 3–4, please submit your solutions in one zip file with all the source code files (just zip the complete project's folder).
  - Submit the zip file and one pdf file to Canvas under Assignments, Homework 5

Questions:

1. (8 pt) Write FuncLang programs to process a list of strings:

   (a) (4 pt) Given you a list of strings, report the total length of all the strings in the list. You can use the built-in function length which returns the length of a string (e.g. (length "hello") returns 5). Hint: you can use functions define in the previous homework to define Total).
   $ (Total (list "hello" "hi" "you"))
   $ 10

   (b) (4 pt) Given you a list of strings, report the length of the longest string in the list
   $ (Max (list "hello" "hi" "you" "supercalifragilisticexpialidocious"))
   $ 34

2. (6 pt) Write a FuncLang program called Shuffle, which takes an input list, "shuffles" it and returns a list whose members are ordered randomly. You can use Random(lst) (it randomly selects an element from the list) to write your program.

3. (20 pt) Extend the FuncLang interpreter by supporting ">" and "<" and "=" on strings and lists, supporting "=" on boolean values. For "=", we return true if the two strings have the exact length and content. Two list values are considered equal if they have the same size and each element of the list is equal to corresponding element in the other list. For "<" and ">", the string and list comparison is done using the length of the strings and lists. That is, "> first second" returns true if the first string/list is longer than the second string/list; and "< first second" returns true if the first string/list is shorter than the second string/list.

   For example,

```
$ (= "abc" "abc")
#t
$ (= "abc" "abcdef")
#f

$ (> "abc" "abcd")
#f
$ (< "abc" "abcdef")
#t


$ (= #t #t)
#t
$ (= #t #f)
#f


$ (= (list) (list))
#t
$ (= (list 1 2 3 4) (list 1 2 3 4))
#t
$ (= (list 1 2 3 4) (list 1 2 3 4 5))
#f
$ (= (list 1 2 3 4 (list)) (list 1 2 3 4 (list)))
#t
$ (= (car (list 1 2 3)) 1)
#t
$ (= (car (list 1 2 3)) 2)
#f
$ (= (cdr (list 1 2 3)) 2)
#f
$ (= (cdr (list 1 2 3)) (list 2 3))
#t
$ (= (cdr (list 1 2 3)) (cdr (list 4 2 3)))
#t
$ (= (cons 0 (list 1 2)) (list 0 (list 1 2)))
#f
$ (= (cons 0 (list 1 2)) (list 0 1 2))
#t
```

```
$ (> (list 1 2) (list))
#t
$ (> (list) (list 1))
#f $ (< (list 1 2) (cdr (list 2 3 4 5)))
#t
```

4. (25 points) Extend the functionality of the Funclang interpreter to add support for following eight predicates: number?, boolean?, string?, procedure?, pair?, list?, null?, unit? A predicate is a function from Funclang value to boolean, i.e. #t or #f.

   The predicate number? evaluates to #t if its input is a number. Similarly, boolean?, string? procedure? pair?, list?, null?, and unit? evaluate to #t if their inputs are boolean, string, procedure, pair, list, null, and unit respectively. Following transcript illustrates some of these predicates.

```
> (number? 342)
#t
> (boolean? (> 300 42))
#t
> (string? "342")
#t
> (procedure? (lambda (x) x))
#t
> (list? (list 3 4 2))
#t
> (list? (cons 1 2))
#f
> (pair? (cons 1 2))
#t
> (list? (cons 1 (list)))
#t
> (list? (list ))
#t
> (null? (list ))
#t
```

   Funclang doesn't currently support expressions that produce unit values, so you wouldn't be able to test unit?, but you should implement it and we will verify your code directly to see if it behaves as intended (e.g (unit? value)).