MP 0 - Basic Scheme

CS421 Agha - Spring 2015

1 Introduction

Assigned: January 22, 2015

Due: 11:59pm, January 30, 2015

Outline: This first assignment is to make sure everyone has a working Scheme environment as well as to give some practice with Scheme. Furthermore, we want to work out any problems with submitting to svn before we get too deep into the course. As such this first assignment will **NOT** be graded on correctness. We do however require you to do and submit this assignment for a completion grade. It will account for 2% of your grade.

2 Submission

First check that you have access to the following svn repository.

`https://subversion.ews.illinois.edu/svn/sp15-cs421/NETID`

Replace "NETID" with your actual netid. If you are registered for the course you should have access.
Checkout your svn directory with `svn co` and add your file with `svn add`.
Make sure your file is named `mp0.scm` and commit with the following command:

```
svn commit -m "your commit message" mp0.scm
```

Please check the following link if you need a refresher on how to use svn.

`https://wiki.cites.illinois.edu/wiki/display/cs242fa14/Subversion+Tutorial`

3 Problems

Please have your solutions in the same order as listed here.

1. Declare a variable `pi` with the value `3.14159`.
2. Declare a procedure `piFunc` which takes no argument and returns the value of the variable `pi`.
3. Declare a procedure `simpleFun` which adds 3 to the input given and then multiples the sum by 4.
   ```
   > (simpleFun 2)
   20
   ```
4. Declare a procedure `mediumFun` which takes an input and writes `"CS"` then `"421"` and returns the input + 3.
5. Declare a procedure `hardFun` which takes an input and writes `"a"` if the input is <0, writes `"b"` if the input = 0, and writes `"c"` if the input >0.
6. Declare a procedure `recFun` which takes in a list and a number `"n"`. Using 0-indexing, add n+i to the i^{th} element of the list, where i is the index of the element
   ```
   > (recFun '(2 3 4) 1)
   '(3 5 7)
   ```
7. Declare a curried procedure `simpleCurry` which takes one argument at a time and computes x*(y+z) given three arguments in succession x, y, z are the arguments for each procedure.
   ```
   > (((simpleCurry 2)3)4)
   14
   ```
8. Using `simpleCurry`, define oneCurry which when given two arguments x, y in succession, multiplies them by 3
   ```
   > (((oneCurry 3) 4)
   21
   ```
9. Scheme has a predefined procedure which adds 1 to an argument. Thus `(add1 4)` returns `5`. Declare a procedure `myAdd1` which takes a list `l1` as an argument and returns a list `l2` such that the i^{th} element of `l2` is the `1` + i^{th} element of `l1`.
   ```
   > (myAdd1 '(3 4 71))
   '(4 5 72)
   ```
10. Use `myAdd1` to define a procedure `myFlexAdd1` which takes a variable number of arguments and returns a list whose i^{th} element is 1 + the i^{th} argument.
    ```
    > (myFlexAdd1 3 4 71)
    '(4 5 72)
    ```