CIS521 - Homework 1

Due Thursday, Feb 6, 2014, 10:30 a.m.

1 [5 points]. Explain why Python is both *strongly* and *dynamically* typed.

2 [5 points]. Suppose Bob has a 2-dimensional 9×9 array (a list of lists) representing a Sudoku board. It contains a number if a position has been filled in already and `None` otherwise.

Suppose that for a given square `(x,y)`, Bob has a list of possible values `possiblilities` and he wants to know which is the right one. Suppose further that he has a function `isSolvable(board)` that takes a board representation as input and returns whether or not it is possible to solve, but which may alter the input board as it decides this. Bob codes the following function to do this:

```
def tryOutPossibilities(board, x, y, possiblilities):
    for z in possiblilities:
        board[x][y] = z
        if isSolvable(board):
            print z + " is the correct choice!"
```

There is a serious problem with Bob's solution as coded. What is it?

3 [5 points]. Given a list `a`, write expressions in slice notation for the following portions of the list:

(a) the third through sixth elements, inclusive

(b) all but the last element

(c) the entire list

4 [10 points]. Suppose that Fred has a program which frequently needs to get the results of same function `f` that takes as input a tuple of immutable objects and returns an output computed entirely from the input tuple. Unfortunately, this function is extremely expensive to compute and so he would like to cache his results so that if he wants the value of `f` for a tuple that has been computed before, he can get the value very quickly. Write a class called `CacheListFunction` to help him. Your class should have the following two methods:

- a constructor that takes one argument: the function `f` to cache. `f` itself can be assumed to take in a single argument as input: a tuple of immutable objects.
- `invoke()`, which should take an immutable tuple as an argument.This function should see if we've calculated `f` on that tuple before; if so, it should return our memorized value. Otherwise it should compute the value, memoize it, and return it. `f` should never be called twice with the same input.

5 [5 points]. Sally wants a dictionary which maps lists of strings to another string she thinks of. She tries this:

```
thoughtOfString={}
sounds=['bark!','meow!','neigh!']
thoughtOfString[sounds]='quack!'
```

but it results in an error. What is the problem? How should she fix it?

6 [10 points]. Sam Sloppycoder wrote the following function which, given a two-dimensional array `m`, a function `g` and the array's dimensions, returns a copy of `m` where each element has been transformed by the function `g`:

```
def TDapply(g, m, d1, d2):
    a=[]
    for i in range(0, d1):
        r=[]
        for j in range(0, d2):
            r.append(g(m[i][j]))
        a.append(r)
    return a
```

Sam's function can be rewritten as a two-argument function `TDapply(g,m)` consisting of a single statement using list comprehensions (which also works more generally on lists of lists of heterogeneous lengths). The first argument of this function is the function `g` to be applied to each element, and the second argument is the list. Write this function.

7 [10 points]. Alice has a function `getNextList()` which will give her the next list in an extremely long sequence of small lists of numbers (fewer than 20 numbers in each list). She wants to concatenate the first ten million of these lists. Alice can think of two ways to do this:

```
def approachA():
    l=[]
    for x in range(1000000):
        l.extend(getNextList())
def approachB():
    l=[]
    for x in
        range(1000000):
        l=l+getNextList()
```

Which will be faster? Why? (Hint: if it's not clear which is faster, try doing this with a dummy `getNextString()` function that always returns `[1,2,3,4]`, reducing the number of lists concatenated to 10,000.)

8 [20 points]. A secret organization has sent you a message using a very simple code: within a text, every letter that follows an uppercase **V**, **F** or **D** should be extracted and concatenated and everything else should be discarded. To address one special case, if you have something like "VFq", it decodes to "F" and not to "Fq" or "q". To keep life simple, you can assume that there are no spaces in the input and that the cyphertext is all one line.

Write a function `decode()` that takes one argument, the string of ciphertext to decode, and returns the decoded message as a string. You can find a practice ciphertext in vfd.txt on the course homepage.

9 [30 points]. In mathematics, a *polynomial* is an expression consisting of variables and constants using addition, subtraction and multiplication. A *univariate polynomial* is a polynomial with just one variable. Some examples of univariate polynomials include $x^2 + 5$ and $(x - 3) \times (x + 9)$. Note that integer exponents can be written using multiplication (i.e. $x2 = x \times x$). Every polynomial can be rewritten as the sum of terms of the form $a_i x^{n_i}$ where a_i is a coefficient and n_i is an integer power, called its coefficient form. For example, the polynomial $(x - 3) \times (x + 9)$ can be written as $x^2 + 6x - 27$. In this problem, you will write a program in Python which will take a univariate polynomial and rewrite it in coefficient form.

(a) Create a data structure for representing univariate polynomials. You are free to use any structure you want. Briefly describe your structure and give the representation for $(x^5 + 5)(x^3 - 2)$.

(b) Write a function called `coefficientForm()` which takes as input a univariate polynomial in the form you specified above. The output of `coefficientForm()` should be the polynomial using the same data structure as above but in coefficient form. This requires distributing multiplication over addition and subtraction (i.e. $a \times (b + c) = a \times b + a \times c$). You should also combine constants, for example, $5 + 3$ should be 8, and $5x^2 + 3x^2$ should be $8x^2$. You do not need to combine exponents(a polynomial having $x^5 x^3$ is fine).

(c) For each of the following polynomials, give the representation of each one and output of your program.

i) $(9 - x^2 + 6x^3)(x^2 - 5x)$

ii) $(x^2 + x + 1)(x + 1)$

iii) $9x^2 + (3x - 2)(5 - x)$