Prelim 2 Review Questions
(some edits made on Sunday April 24, 2016)

1 Functions on Lists

(a) The *even-odd* sort of a list that has even length permutes entries so that all the even-index entries come first followed by all the odd-indexed entries. To illustrate, suppose we have the following length-8 list:

'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'

Here are the length-4 lists of the even-indexed entries and the odd-indexed entries:

'a'	'c'	'e'	'g'

'b'	'd'	'f'	'h'

And here is the even-odd sort of the above length-8 list:

'a'	'c'	'e'	'g'	'b'	'd'	'f'	'h'

This operation *could* — but for this question you are *not* allowed to do so — can be carried out very simply using list slicing and list concatenation: indeed, if `x` has length `n` and `n` is even, then the list `x[0:n:2] + x[1:n:2]` is the even-odd sort of `x`. Implement the following procedure so that it performs as specified, *using just for-loops and subscripting. No list slicing or list concatenation allowed.*

```
def EvenOddSort(x):
    """ Performs an even-odd sort of x

    Precondition: x is a list with even length"""
```

Note that `EvenOddSort` does not return any values. Again, no list slicing or list concatenation allowed.

For a hint on important and potentially common errors, see this footnote.[1]

(b) Assuming that the procedure `EvenOddSort` is available, implement the following function so that it performs as specified:

```
def MultipleSort(x,N):
    """ Returns a list obtained by performing N even-odd
    sorts of the list x. The list x is not altered.

    Precondition: x is a list with  even length and N is a positive int.
    """
```

Use a loop that calls `EvenOddSort` `N` times. (Don't try to do some fancy "if `N` is even, I'll get the same list back" type of reasoning.)

Some notes on potential errors in this footnote.[2]

[1] Don't change the list *while* constructing an even-odd-sorted version! And if `y` is your even-odd sorted version, don't just do `x=y`! (Why?)

[2] `EvenOddSort` doesn't return anything. Don't operate on `x` or you'll change it.

2 Farthest Point

Assume the existence of the following class, and that the command `import math` has been included beforehand.

```
class Point(object):
   """ Attributes:
          x   the x-coordinate   [float]
          y   the y-coordinate   [float]
   """
   def __init__(self,x,y):
       self.x = x
       self.y = y

   def Dist(self,other):
       """ Returns a float that is the distance from self to other.

       Precondition: other is a Point
       """
       return math.sqrt((self.x-other.x)**2+(self.y-other.y)**2)
```

Complete the following function so that it performs as specified

```
def FarthestPt(L,idx,P)
   """ Returns an integer j with the property that the distance from
   L[j] to P is maximum among all the ***unvisited*** points.

   If idx[i] = 1, then we say that L[i] has been visited. If idx[i] = 0, then
   we say that L[i] is unvisited.

   Preconditions: L is a list of references to Point objects, P is a reference
   to a point object, and idx is a list of ints that are either zero or 1. The
   lists idx and L have the same length and idx has at least one zero entry.
   """
```

3 Nested Loops

1. What is the output if the following is executed?

```
s = "abcd"
for i in range(4):
    for j in range(i+1,4):
         print i, j, s[i]+s[j]
```

2. For each key in dictionary `D`, write down the key and corresponding value in `D`.

```
D1 = {'a':'one', 'b':'two', 'c': 'three', 'd':'four'}
D2 = {'c':'five', 'd':'six', 'e': 'seven' ,'f':'eight'}
D = {}
for d in D1:
   D[d] = D1[d]
```

```
for d in D2:
   D[d] = D2[d]
```

4 More work with lists, which are objects

(a) If the following is executed, then what are the first five lines of output?

```
x = [10,20,30]
for k in range(1000):
     print "k:", k, "x in the loop", x
     x.append(x[0])
     x = x[1:4]
```

(b) If the following is executed, then what is the output? For full credit you must also draw two state diagrams. The first should depict the situation just after the `Q.x = 0` statement and the second should depict the situation just after the `P = Point(7,8)` statement.

```
P = Point(3,4)
Q = P
Q.x = 0
print Q.x, Q.y, P.x, P.y
P = Point(7,8)
print Q.x, Q.y, P.x, P.y
```

(c) If the following is executed, then what is the output?

```
x = [10,20,30,40]
y = x
for k in range(4):
   print "x is", x
   print "y is", y
   print "...."
   x[k] = y[3-k]
print x
```

5 Dictionaries

(a) Complete the following function so that it performs as specified

```
def  F(s,D):
   """ Returns True if s is a key for D and every element in D[s] is also
   a key in D. Otherwise returns False.

   Precondition: s is a nonempty string and D is a dictionary whose keys are strings
   and whose values are lists of strings.

   """
```

6 Methods and Lists of Objects

Assume the availability of the following class:

```
class City(object):
    """
    attributes:
        name      the name of a city [str]
        high:     the record high temeratures [length-12 list of int]
        low:      the record low temperatures [length-12 list of int]
    """

    def __init__(self,cityName,theHighs,theLows):
        """Returns a reference to a City object
        PreC: cityName is a string that names a city.
        theHighs is a length 12 list of ints.
        theHighs[k] is the record high for month k (Jan is month 0)
        theLows is a length 12 list of ints
        theLowss[k] is the record high for month k (Jan is month 0)
        """
        self.name = cityName
        self.high = theHighs
        self.low  = theLows
```

(a) Complete the following method for the class `City` so that it performs as specified.

```
    def HotMonths(self):
        """ Returns the number of months where the record high
        is strictly greater than 80.
        """
```

(b) Complete the following method for the class `City` so that it performs as specified. Your implementation must make effective use of the method above.

```
    def Hotter(self,other):
        """Returns True if the city encoded in self has strictly more
        hot months than the city encoded in other.

        A month is hot if the record high for that month is > 80

        PreC: other is a city object
        """
```

(c) Complete the following method for the class `City` so that it performs as specified.

```
    def Variation(self):
        """ Returns a length 12 list of ints whose k-th entry
        is the record high for month k minus the record low for month k.
        """
```

(d) Complete the following method for the class `City` so that it performs as specified.

```
    def Exaggerate(self):
        """ Modifies self.high so that each entry is increased by 1
        and modifies self.low so that each entry is decreased by 1
        .
        """
```

(e) Complete the following function so that it performs as specified. Assume that the methods in parts (a) and (b) are available; your implementation must make effective use of them.

```
def Hottest(C):
    """ Returns an item from C that represents the city that
    has the most hot months.
    PreC: C is a list of references to City objects
    """
```