1 Review Exercises

A study guide for exam 3 is available on the course website. This handout presents some review exercises.

1. Write a function that accepts a file name as a parameter (a string), and prints each line of the file in order of longest line to shortest line. The function does not need to return anything.

Solution:

```
def sort_by_length(filename):
    f = open(filename)
    data = []
    for line in f:
        data.append([len(line), line.strip()])

    data.sort()
    for datum in data:
        print datum[1]

sort_by_length('example.txt')
```

2. Write a function `pruned_list` that takes a string `frag` a list of strings `words` and returns a new list that contains all words that start with `frag`.

```
>>> L = ['cat', 'horse','cattail', 'dog']
>>> prune_list('cat', L)
['cat', 'cattail']
```

Solution:

```
def pruned_list(frag, words):
    new_words = []
    for i in range(len(words)):
        if words[i].startswith(frag):
            new_words.append(words[i])
    return new_words

L = ['cat', 'horse','cattail', 'dog']
print pruned_list('cat', L)
```

3. Same idea as previous question but *modify* the list rather than return a new list. Write a function `prune_list` that takes a string `frag` a list of strings `words` and *modifies* the list, removing all words that do not start with `frag`. The funtion should return None.

```
>>> L = ['cat', 'horse','cattail', 'dog']
>>> prune_list('cat', L)
>>> print L
['cat', 'cattail']
```

Solution:

```
def prune_list(frag, words):
    i = 0
    while i < len(words):
        if not words[i].startswith(frag):
            words.pop(i)
        else:
            i += 1

L = ['cat', 'horse','cattail', 'dog']
prune_list('cat', L)
print L
```

4. Al and Buffy go out birding. Each returns with a list of bird sightings. For example, Al returns with `['hawk', 'jay', 'hawk', 'cardinal', 'jay', 'jay']` and Buffy returns with `['kestrel', 'kestrel', 'jay']`. Write a function `eagle_eye_al` that takes in two lists, Al's and Buffy's, and returns the kind that Al saw the most frequently but Buffy did not see it at all.

   In the above example, the correct answer is `'hawk'` because Al saw two hawks whereas Buffy saw none. (Al saw jays more frequently than hawks, but Buffy saw a jay so jays are excluded from consideration.)

Solution:

```
def histogram(L):
    hist = {}
    for item in L:
        if item not in hist:
            hist[item] = 1
        else:
            hist[item] += 1
    return hist

def eagle_eye_al(a_list, b_list):
    a_hist = histogram(a_list)
    b_hist = histogram(b_list)

    max_key = None
    max_count = -1
    for key in a_hist:
        if key not in b_hist and a_hist[key] > max_count:
            max_count = a_hist[key]
            max_key = key
    return max_key


L = ['hawk', 'jay', 'hawk', 'cardinal', 'jay', 'jay']
L2 = ['kestrel', 'kestrel', 'jay']
print eagle_eye_al(L, L2)
```

5. Write a function called `reverseDict` that takes a dictionary as a parameter and returns a new dictionary in which the keys and values of the original dictionary are inverted. In particular, if `k:v` is an entry in the original dictionary, then `v` should be a key in the new dictionary with `k` as a corresponding value.

   Because dictionary values are not necessarily unique, values in the new (inverted) dictionary will need to be a list. The list assigned to key `x` in the new dictionary should contain all the keys from the original dictionary for which the value is `x`.

   For example,

   ```
   reverseDict({'apple':3, 'potato':2, 'avocado':3})
   ```

   should return the dictionary

   ```
   {2:['potato'], 3:['apple','avocado']}.
   ```

   Solution: Look at Handout `dic2` we did it

6. Consider these two programs.

   (a) What is printed?

```
def f(t):
    t = t.upper()
    return t

s = 'abc'
f(s)
print s
```

   (b) What is printed?

```
def g(L):
    for i in range(len(L)):
        L[i] = L[i].upper()
    return L

a_list = ['abc', 'xyz']
g(a_list)
print a_list
```

   (c) Explain the difference between the two functions in terms of aliasing.

Solution: In both cases, the argument and the parameter are aliases (i.e., `s` is aliased with `t` in the first and `a_list` is aliased with `L` in the second). However, an important difference is that strings are immutable whereas lists are mutable. Since lists are mutable, the line `L[i] = L[i].upper()` actually modifies the list object that `a_list` refers to, thus `a_list` is changed by the function. Since strings are immutable, `t = t.upper()` simply makes a new string which `t` refers to but the original string that `s` refers to is unmodified.

7. For each of the following segments of code, show what the output will be. There are no errors in either code segment.

   (a) What is printed?

```
y = ['f', 'x']

def woof(y):
   y[0] = y[-2]
   x = y[0::2]
   x[-1] = '?'
   print "X =", x
   print "Y =", y
   return y
   return x

x = ['a', 'r', 'f']
woof(x)
print "x=", x
print "y=", y
```

   (b) What is printed?

```
def bark(x, y):
   print "len(X)=", len(x)
   print "Y=", y
   x['a'].append(3)
   del x['b']
   y.append('!!')


w = [1,2]
y = {'a': w, 'b': [2,3]}
bark(y, w)
print 'w=' , w
print 'y=', y
```