1 Review

Exam 3 covers units 1-5, and does not include unit 6 (recursion). Since exam 2 covered units 1-4, you should focus your studying on unit 5 but be aware that anything from units 1-4 is fair game. The handouts for unit 5 are on Moodle and on the course schedule website.

Exercises

Some solutions are presented in class and also included in the moodle version of this handout.

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 `prune_list` that takes a string `frag` a list of strings `words` and returns a new list that contains all the words that start with `frag`.

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

Solution:

```
def prune_list0(frag, words):
    less_words = []
    for word in words:
        if word.startswith(frag):
            less_words.append(word)
    return less_words
```

```
L = ['cat', 'horse','cattail', 'dog']
print prune_list0('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 function 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
```

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. 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.