SI 106 – Winter 2014 **Name: __Solution Set____________**

Written Final Exam

This exam is closed book - you are allowed only one page of notes (double-sided). If a question seems unclear - please write down any assumptions you feel are needed. If you think that there is a just-plain mistake/typo - check with an instructor.

Anywhere we ask you what will be printed out, if you think an error will be generated, you may write "error". You do not need to write out what the whole error message would be.

1. (4 points) You are working collaboratively with a friend on writing some code for a project. Your friend stayed up later than you did and left you a note, saying "Made great progress! Put all my changes in our bitbucket repo." Which git command do you need to do to get your friend's latest updates?

    a. git add
    b. git commit
    c. git pull
    d. git push

2. (4 points) You've made some changes to the code but haven't done any git operations this morning. Your friend is awake now and would like you to share your changes. In which order would you do these git operations?

    a. add, commit, push
    b. add, push, commit
    c. add, push, pull
    d. commit, add, push
    e. commit, add, pull
    f. pull, push

3. (5 points) You have been part of an open source project from its founding. A new leader has emerged and, well, let's just say you don't get along. The leader has convinced the rest of the core group to kick you out. The project is still remaining an open source project. Which of the following things will you not be able to do? Circle all that apply

    a. Download the latest version of the code
    b. Run the code
    c. Modify the code
    d. Get your modifications incorporated into the project's bitbucket repository
    e. Distribute a modified version, including much of the original code but with some of your modifications, through a different bitbucket repository

4. (5 points) You sense a commercial opportunity in the heartbleed fiasco. You would like to make a company that sells a better version of the OpenSSL software. You plan to hire people to do extensive code reviews, run lots of automated security tests, and post bounties for anyone who can find vulnerabilities in the code and report them to you. This will cost you lots of money. But you plan to earn that and more in revenue from selling your software rather than giving it away for free. Your customers will, for a fee, get access to your better version of the software and will have the right to run it but not to modify it or give it away to anyone else.
   Before starting your company, you decide to check out the license under which the existing OpenSSL software is released. Would it be better for your business plan if you find that it is a GPL license, or a BSD-style license? Explain why, in 2-3 sentences.

   A BSD-style license would be better, because it doesn't put any restrictions on your ability to distribute a more-restricted version (the one you'll charge for). The GPL would require that any modified version that you distribute also has the GPL (that's the viral nature of the GPL), and thus you would not be able to distribute a version that your customers can't modify or redistribute.

Assume that we have already executed the following code (recall that """ is the delimiter for strings that are on more than one line):

```
s = """<entry>
    <id>tag:search.twitter.com,2005,1142881099</id>
    <published>2009-01-23T20:04:53Z</published>
  </entry> """
```

5. (4 points) What will the following code print out?

```
print s.find(">")
```

6

6. (4 points) What will the following code print out?

```
print len(s.split(':'))
```

4 (there are 3 colons)

7. (5 points) What will the following code print out?

```
print len(s.split('>')[2].split(':'))
```

2 (there is one colon in tag:search.twitter.com,2005,1142881099</id)

8. (5 points) Define a function f that takes a list of strings and returns a list containing the first letter of every word that contains the letter z. Make it pass the test below. Your function must use a list comprehension to create the list it returns, or use map and/or filter.

```
def f(L):
    return [item[0] for item in L if 'z' in item]
```

```
test.testEqual(f(['Amazing', 'corny', 'zany']), ['A',
'z'])
```

9. (5 points) Define a function g that takes a list of strings and returns a list of them, sorted in alphabetic order by their **last** character. Make it pass the test below.

```
def g(L):
    return sorted(L, key = lambda x: x[-1])
```

```
test.testEqual(g(['Amazing', 'corny', 'zanier']),
['Amazing', 'zanier', 'corny'])
test.testEqual(g(['good', 'good on ya', 'good on
you'], ['good on ya', 'good', 'good on you'])
```

10. (5 points) Fill in the second line **using a string interpolation** (with the % operator), in order to make the tests pass.

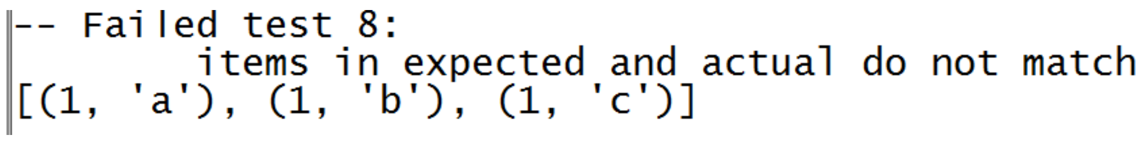
```
def interp(x, y):

  mystr = "That is %d in a row, %s. Congratulations!" 
% (x, y)




  return mystr


test.testEqual(interp(5, "sir"), "That is 5 in a row,
sir. Congratulations!")
test.testEqual(interp(6, "your highness"), "That is 6
in a row, your highness. Congratulations!")
```

11. (5 points) Fill in the parameter list for the function h below so that the tests pass.

```
def h(x, y= 3, z = 4          ):

    return[x, y, z]

test.testEqual(h(1, 2), [1, 2, 4])
test.testEqual(h(1, z = 5), [1, 3, 5])
```

12. (5 points) You have defined a function enum. It takes a list as input and is supposed to produce a list of tuples that number the original items: the first item is paired with 1, the second with 2, etc. Unfortunately, your code isn't working correctly yet. The code generates the output shown in the screenshot. Rewrite the definition of enum so that it passes the test.

```
def enum(L):
    res = []
    for item in L:
        n = 1
        res.append((n, item))
        n = n + 1
    return res

test.testEqual(enum(["a", "b", "c"]), [(1, "a"), (2,
"b"), (3, "c")])
print enum(["a", "b", "c"])
```



```
-- Failed test 8:
         items in expected and actual do not match
[(1, 'a'), (1, 'b'), (1, 'c')]
```

```
def enum(L):
     ####write your new definition below

    res = []
    n=1
    for item in L:
        # n = 1
        res.append((n, item))
        n = n + 1
    return res
```

For the next three questions, the function count_guesses has been defined as below, slightly modified from the ps10 solutions.

```
def count_guesses(next_letter, guesses):
    """guesses is a list of guesses to be made, in
order. Returns the number of guesses that will be made
in order to guess next_letter, or None if it's not
among the guesses"""
    try:
        return guesses.index(next_letter) + 1
    except:
        print "%s not among guesses" % next_letter
```

The comment string defines what the correct output should be. The function is implemented correctly, so any test you write should pass.

13. (4 points) In order to write test cases for count_guesses, you would make:

    a. Return value tests

    b. Side effect tests

    Briefly justify your answer, in 1-2 sentences.

14. (5 points) Write a test that checks that the right thing happens when next_letter is among the guesses

    test.testEqual(count_guesses("a", ["b", "c", "e", "a", "f"]), 4)

15. (5 points) Write a test that checks that the right thing happens when next_letter is not among the guesses.

    test.testEqual(count_guesses("a", []), None)

16. (10 points) Recall that we discussed in class that after a sequence of capital letters in a text, intuitively it would have made more sense for the Shannon guesser to guess another capital letter. As one way to do that, you could just take the list of guesses that would otherwise be made, and rearrange them to put all the capitals first, keeping the same order among the capitals that they had in the original guess list. Define a function, caps_first, which does that. It should pass the test below.

```
def caps_first(L):
   """move all the capital letters in L to the front of the list,
maintaining the order"""
   L_caps = [c for c in L if c in caps]
   L_small = [c for c in L if c not in caps]
   return L_caps + L_small
```

```
test.testEqual(caps_first(["A", "i", "K", "e"]), ["A",
"K", "i", "e"])
```

In the ps10 solutions, we provided code that represented each Facebook post as a dictionary, with whatever keys were included in the json dictionary that Facebook returned, plus a new key for 'redundancy' that our code generated. The relevant code from ps10 is below, slightly modified and simplified. You may assume that game() and heuristic_guesser() are defined as in the ps10 solutions, but not shown here.

In ps9, we used instances of a class Post to represent Facebook posts, instead of dictionaries. The next few questions walk you through rewriting the ps10 solution set to use the Post class instead of using a dictionary.

```python
####...code for Shannon guesser omitted
import facebook
access_token = "your token here"
graph = facebook.GraphAPI(access_token)
feed = graph.get_object("245188182322906/feed?limit=150")

posts = [p for p in feed['data'] if 'message' in p]

def find_redundancy(post):
    message = post['message']
    min_guesses, actual_guesses = game(message,heuristic_guesser)
    redundancy = actual_guesses/float(min_guesses)
    post['redundancy'] = redundancy
    return post

for p in posts:
    find_redundancy(p)

surprising_posts = sorted(posts,key=lambda p: p['redundancy'], reverse=True)

print "-----Top 10 most surprising-----"
for p in surprising_posts[:10]:
    print "%.2f: %s" % (p['redundancy'], p['message'])

print "\n----most predictable posts------"
for p in surprising_posts[-10:]:
    print "%.2f: %s" % (p['redundancy'], p['message'])
```

We have provided a skeleton of the rewrite of the code. Follow the instructions to fill in various lines.

```
class Post():
    """object representing one post"""
    def __init__(self, post_dict):
        self.message = post_dict['message']

    #Rewrite the find_redundancy() function to make it a
    method of the Post class.
```

(5 points)

```
    def find_redundancy(self):
        min_guesses, actual_guesses = game(self.message)
        self.redundancy = actual_guesses/float(min_guesses)
```

```
# this replaces line 7; making a list of instances
instead of a list of dictionaries; nothing you need to
do here except understand what it does.
posts = [Post(p) for p in feed['data'] if 'message' in p]
```

(5 points each for the next three questions)

```
# Rewrite lines 16-17 so that find_redundancy
is invoked as a method instead of as a
function.
```

```
# Rewrite line 19 to sort the instances
instead of sorting the dictionaries
```

```
# Rewrite lines 22-23 and 26-27 to print the
same things they printed before, but taking
into account that the posts are instances
instead of dictionaries now.
```

```
surprising_posts = sorted(posts,key=lambda p:
p.redundancy, reverse=True)

for p in surprising_posts[:10]:
    print "%.2f: %s" % (p.redundancy, p.message)

for p in surprising_posts[-10:]:
    print "%.2f: %s" % (p.redundancy, p.message)
```