COSC 101, Exam #3
19 April 2017

Name: _____ Section: 9:55 / 1:20 / 2:45

- Do not open the exam until instructed to do so.
- Write your name and circle your section time.
- You have 60 minutes to complete this exam; use your time wisely.
- There are 5 questions and a total of 50 points available for this exam. Don't spend too much time on any one question.
- If you want partial credit, show as much of your work and thought process as possible.
- Since indentation is important in Python, please be sure that your use of indentation is obvious for any code you write.
- When defining functions, it is not necessary to write docstrings nor is it necessary to write comments.
- You can always define helper functions.
- If you run out of space while answering a question, you can continue your answer on one of the scrap pages at the end of the exam. If you do so, be sure to indicate this in two places: (1) below the question, indicate which scrap page contains your answer, and (2) on the scrap page, indicate which question you are answering.

Question	Points	Score
1	8	
2	12	
3	10	
4	10	
5	10	
Total:	50	

1. (8 points) Assume that the following statements have already been executed:

```
s = 'spring fling'
q = [5, 4, 1, 7]
d = {'s': 1, 't': 7, 'x': 5, 'y': 4}
```

For each of the following expressions, evaluate the expression and write the resulting value, or identify the error in the code that would prevent it from running.

(a) `d[s[0]]`

Solution:

```
1
```

(b) `s[1::2]`

Solution:

```
'pigfig'
```

(c) `s[sum(q[1:])]`

Solution:

```
Error: string index out of range
```

(d) `len(q) in d`

Solution:

```
False
```

(e) `len(d) in q`

Solution:

```
True
```

(f) `s.split()[0][-3:] in s.split()[1]`

Solution:

```
True
```

(g) `max(q) - max(d.values())`

Solution:

```
0
```

(h) `d[s[min(d.values())]]`

Solution:

```
Error: 'p'
```

2. (a) (4 points) What is the output of the following program? (Recall that a `ValueError` is raised when Python attempts to convert something to a number but fails.)

```
def square(s):
    try:
        print('The number is',s)
        print('Its square is',int(s)**2)
        print('All for now')
    except ValueError:
        print('Oops, no can do')
    print('Later!')

square('four')
```

Solution:

```
The number is four
Oops, no can do
Later!
```

(b) (4 points) What is the output of the following program?

```
def mystery(s):
    if len(s) <= 0:
        return ''
    return s[0] + mystery(s[1:]) + s[0]

print(mystery('abc'))
```

Solution:

```
abccba
```

(c) (4 points) What is the output of the following program?

```
def my_range(L):
    return max(L) - min(L)

def short_vals(D, k):
    Q = []
    for key in D:
        if len(key) < k:
            Q += [D[key]]
    return Q

d = {'january': 20,
     'march': 30,
     'june': 70,
     'august': 80,
     'october': 40}

print(my_range(short_vals(d, 6)))
```

Solution:

40

3. (10 points total) You are writing a program to display a timeline. You are given a list of events including most of the years when they occurred. Unfortunately, the format is inconsistent: not all events have years, and the ones with years don't always have them in the same place.

   If there is a year, it is always the only 4-digit substring of the event description. For example:

```
events = [ 'Declaration of Independence July 4, 1776',
           'Founding of Colgate: 1819',
           'In 1819, Florida was ceded to the US',
           'Launch of Sputnik',
           '4/11/1970: Launch of Apollo 13' ]
```

   From this, you want a list of distinct years in reverse-chronological order:

```
[1970, 1819, 1776]
```

   You will write code for this task in two parts.

   (a) (5 points) Write a function `get_year`, which takes a single string as a parameter and returns the first 4-character substring containing only digits. Your function *should* handle strings containing no 4-digit substrings by returning something to indicate that no year was found.

   (*Hint:* if `s` is a string, then `s.isdigit()` is `True` if and only if it consists entirely of digits.)

(b) (5 points) Write a function `order_years`, which takes a list of strings as a parameter. Each string in the list is an event description, and somewhere in that string *might* be a year.

It should return a list of all the distinct years found, ordered from most recent to least recent. "Distinct" means that each year should appear in the list only once.

This function should use the `get_year` function from the previous part. For this part, you may assume that `get_year` works as described, even if your implementation in the previous part is incorrect or incomplete.

(*Hint:* Recall that lists have `sort` and `reverse` methods that you can use.)

Solution:

```
def get_year(s):
    for i in range(len(s)-3):
        x = s[i:i+4]
        if x.isdigit():
            return x
    # if no year is found, return None

def order_years(L):
    years = []
    for s in L:
        y = get_year(s)
        if y is not None and int(y) not in years:
            years.append(int(y))
    years.sort()
    years.reverse()
    return years
```

4. (10 points) Write a function `length_histogram` that takes as a parameter a string representing the name of a file. It returns a word-length histogram, which is a dictionary in which the keys are integers representing word lengths, and each key's value is the number of words in the file of that length.

   If the file cannot be opened successfully (an `IOError` results), just return an empty dictionary.

   Assume that a word is a substring containing no intervening whitespace. Also, don't worry about case or punctuation, meaning that `'there?'`, `'There'`, and `'there'` are all different words.

   As an example, if the file you are given is:

```
one on Monday
two on Tuesday
one on Wednesday
```

   the dictionary you should return is

```
{ 2:3, 3:3, 6:1, 7:1, 9:1 }
```

Solution:

```
def length_histogram(filename):
    histogram = dict()

    try:
        f = open(filename)
    except IOError:
        return histogram

    for line in f:
        words = line.split()
        for word in words:
            length = len(word)
            if length in histogram:
                histogram[word] += 1
            else:
                histogram[word] = 1
    f.close()
    return histogram
```

5. (10 points) You are trying to determine the winner of a monthly fitness prize, who is the person who had the "longest streak" of consecutive days in which a fitness activity was completed. You are given the information in a dictionary, in which the keys are names of people competing for the prize, and values are lists of days of fitness activity for the corresponding person. (You should assume the data is for a single month, and so the days in each list are integers between 1–31, inclusive. You can also assume the days are in ascending order.)

   Write a function `prize_winner` that takes as a parameter a dictionary in this format and returns the names of anyone eligible for the prize. The return type should be a list, because it is possible that more than one person had the longest streak of fitness days. The order of the names in the returned list does not matter.

   For example, if the dictionary given is:

```
fitness = { 'Alex': [1, 2, 4, 5, 12, 13, 14, 20],
            'Pat': [3, 4, 5, 17, 18, 29, 30],
            'Sam': [1, 2, 9, 10, 12, 13, 18, 19] }
```

   then the list returned by `prize_winner` should be `['Alex', 'Pat']` because both have a streak of 3 consecutive days ($12, 13, 14$ for Alex and $3, 4, 5$ for Pat), and the longest streak attained by anyone is 3 days.

   Requirement: Design matters here, so you should have *at least one helper function* for `prize_winner`.

Solution:

```
def streak_length(L):
    '''Returns length of the longest streak'''
    curlen = 1
    maxlen = 0
    for i in range(1,len(L)):
        if L[i-1] + 1 == L[1]:
            curlen += 1
            if curlen > maxlen:
                maxlen = curlen
        else:
            curlen = 1
    return maxlen

def get_keys(D, n):
    '''Returns list of keys with value n'''
    L = []
    for key in D:
        if D[key] == n:
            L.append(key)
    return L
```

```
def prize_winner(D):
    '''Returns person with longest streak'''
    streaks = {}
    maxlen = 0
    for person in D:
        streak = streak_length(L)
        if streak > maxlen:
            maxlen = streak
        streaks[person] = streak

    return get_keys(streaks, maxlen)
```