COSC 101, Exam #3
April 2019

Name: ____________________

Please write your name above. Do not start the exam until instructed to do so.

You have 50 minutes to complete this exam.

There are 5 questions and a total of 51 points available for this exam. Don't spend too much time on any one question.

Since indentation is important in Python, please be sure that your use of indentation is obvious for any code you write.

If you want partial credit, show as much of your work and thought process as possible.

If you run out of space for answering a question, you can continue your answer on one of the blank 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 blank page contains your answer, and (2) on the blank page, indicate which question you are answering.

Question	Points	Score
1	10	
2	13	
3	5	
4	8	
5	15	
Total:	51	

1. (10 points) Assume that the following statements have already been executed:

```
a = 1.1
b = [('one two three','yes no'),[a + -1.5,a -a,-0.7,int(a)],list(range(3)),[Tru
d = {-8:10,2:b,-10:9}
e = {(1,):1, ():0}
```

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) e[0]

Solution:

```
Error
```

(b) [0,1,2] in b == False

Solution:

```
False
```

(c) str((len(b + [int(a),1.7,1.1])))

Solution:

```
7
```

(d) d.get(9,2)

Solution:

```
2
```

(e) b [1:4:2][-3:1:2]

Solution:

```
[[-0.4, 0.0, -0.7, 1]]
```

(f) '1' in str(a)

Solution:

```
True
```

(g) d.get(list (d.keys ())[1])[0]

Solution:

```
('one two three', 'yes no')
```

(h) b [0][1:]

Solution:

```
('yes no',)
```

(i) e.keys ()[0][1:]

Solution:

```
Error
```

(j) b [0][0:1:-1]+(1, 2, 3)

Solution:

```
(1, 2, 3)
```

2. (a) (4 points) What is the output of the following program if the file taxinfo.txt does not exist?

```
try:
    dict=dict()
    with open('taxinfo.txt') as myfile:
      count=1
      for line in myfile:
        dict[count]=line
        count+=1
    print('next file')
except:
    print('didn't find the file!')

print("The result is", dict)
```

Solution:

```
didn't find the file!
The result is {}
```

(b) (4 points) What is printed by running the code of part (a) if the input file taxinfo.txt only contains two blank lines?

Solution:

```
next file
The result is {1: '\n', 2: '\n'}
```

(c) (5 points) What is printed by the following program?

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

def short_vals(D, k):
    D['all']=sum(list(D.values()))
    Q = []
    for key in D:
        if len(key) < k:
            Q += [D[key]]
    return Q

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

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

Solution:

```
170
{'january': 20, 'march': 30, 'june': 70, 'august': 80, 'all': 200}
```

3. (5 points) Write a function that receives a dictionary and prints a dictionary of its items sorted in ascending order. Then write the output of running your function when the following input is given to it.
d={3:'six', 5:'five', 1:'one', 3:'three'}

Solution:

```
def sortDict (d):
    sdict={}
    itemsList=[]
    for i in d.items():
        itemsList.append(i)
    sortedList=sorted(itemsList)
    for i in sortedList:
        sdict[i[0]]=i[1]
    print(sdict)

sortDict ({3:'six', 5:'five', 1:'one', 3:'three'})

{1: 'one', 3: 'three', 5: 'five'}
```

4. (8 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.

   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

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

   The first three and the last two lines of the code are

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

    #the rest of the code

    f.close()
    return histogram
```

   For the rest of the code, select one line of code from each of the pairs of lines of code given in the next page and reorder them to solve the problem:

```
A1        words = line.split()
A2        words = line.strip()

B1        if word in words:
B2        for word in words:

C1                histogram[length] = 0
C2                histogram[length] = 1

D1            else:
D2            elif:

E1    for line in f:
E2    for line in f.readline():

F1            length = len(word)
F2            length = len(words)

G1            if length in histogram.values():
G2            if length in histogram:

H1                histogram[length] += 1
H2                histogram[word] += 1
```

Select only 8 lines of code from above, and only one line from each pair. You may fill in line identifiers (e.g., E2) below, or write out the code.

Solution:

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

5. (15 points) Write a program which checks in a text file and summarizes the events addressed in the file. The program should work as follows:
It assumes that each event in the input file comes with a date in the format mm/dd/yyyy (and anything in this format is a date). It has to have a function isDate (4 points) that receives a string and if the string is a date, it returns True otherwise False. You can assume that in such input files the slashes appears only in a date and as a separator for some non-digit words. It has to have a function getMonth (4 points) that receives a date as a string and returns a string representing the name of the month associated to the date (e.g. August). Your program also has to have a function getSummary (7 points) that receives a file name and returns a dictionary in which the keys are the names of the months and for each month the value is a tuple of two elements, the first element is the number of events occurred in that month and the second element is the list of actual dates of events of that month. You can assume that all events occur during the second half of the year.

   Example:
   Input text file: "It rained in 08/27/2018 and 08/28/2018 while a week later in 09/03/2018 it was sunny!
   The sequence of rainy/sunny/rainy occurred for 12 consecutive weeks."
   Output: {'August':(2,[08/27/2018, 08/28/2018]), 'September':(1,[09/03/2018)}

Solution:

```
def isDate (s):
    parts=s.split('/')
    for part in parts:
        if not part.isdigit():
            return False
    return True

def getMonth (s):
    months=['July','August', 'September', 'October', 'November', 'December']
    parts=s.split('/')
    num=int(parts[0])
    if 7<=num<=12:
        return months[num-7]

def getSummary (filename):
    summary={}
    lists =[[],[],[],[],[],[]]
    months=['July','August', 'September', 'October', 'November', 'December']

    with open(filename) as myfile:
        words=myfile.read().split()
        for word in words:
            if isDate(word):
                if    getMonth(word)== 'July':
                    lists[0].append(word)
                elif getMonth(word)== 'August':
                    lists[1].append(word)
                elif getMonth(word)== 'September':
                    lists[2].append(word)
                elif getMonth(word)== 'October':
                    lists[3].append(word)
                elif getMonth(word)== 'November':
                    lists[4].append(word)
                elif getMonth(word)== 'December':
                    lists[5].append(word)
        for i in range(6):
            if len(lists[i])>0:
                summary[months[i]]=(len(lists[i]),lists[i])
    return summary
```

(This page is intentionally blank. Label any work with the corresponding problem number.)

(This page is intentionally blank. Label any work with the corresponding problem number.)