COSC 101, Exam #3
Fall 2019

Name: _____

Write your name. Do not open the exam until instructed to do so.

There are 5 questions and a total of 50 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 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	10	
3	10	
4	10	
5	12	
Total:	50	

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

```
s = 'thanks giving'
q = [5, 2, 7, 3]
d = {'n':4, 's':q, 'g':3}
```

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[-2]]`

Solution:

```
4
```

(b) `s.split()[1:7:2]`

Solution:

```
['giving']
```

(c) `len(q) in d`

Solution:

```
False
```

(d) `':'.join(sorted(d.keys()))`

Solution:

```
'g:n:s'
```

(e) `len(d) in q`

Solution:

```
True
```

(f) `d[s[2]][3]`

Solution:

```
Error: 'a'
```

(g) `q[d['g']::]`

Solution:

```
[3]
```

(h) `list(d.values()) + 5`

Solution:

```
Error: can only concatenate list (not "int") to list
```

2. For each part, state what the program prints. None of these programs contains an error.

   (a) (3 points) When this program is run, what is printed?

   Solution:

```
A = [20, 60]
a = ['x', 'y']
b = [20, 10]
```

   (b) (4 points) When this program is run, what is printed?

   Solution:

```
A1 = 12
A2 = 9
ax = 6
ay = 12
```

(c) (3 points) When this program is run, what is printed?

Solution:

```
B0 zero
B1 one
alen = 3
```

3. (10 points) Several graduate students work in a research laboratory and track their research expenses in a text file in the following format:

```
Date,Std ID,Last name,First name,Expenses
8/11/19,10001,Beckham,Andrew,CL2S: $239; TTK: $121; JY80: $1023
8/11/19,10003,Haack,John,TTK: $302; RR3Z: $989
8/11/19,10002,Saffold,Sarah,JY90: $860; TTL1: $65
8/12/19,10001,Beckham,Andrew,CL2S: $400
8/12/19,10002,Saffold,Sarah,JY80: $1023
8/16/19,10004,Dahl,Rebekah,RR3Z: $1210; F12A: $420; DODL: $90
8/16/19,10002,Saffold,Sarah,TTK: $109; TTK: $215
8/20/19,10003,Haack,John,XYZ: $85; ABC: $400; W2Q3: $720
```

Notice that each line contains a comma-separated date, a student id, the student's name, and one or more expenses where each expense includes an expense id and a cost. Their research supervisor wants to have a new file with expense cost totals for each student, ordered by student id like the following:

```
10001: 1783
10002: 2272
10003: 2496
10004: 1720
```

The first three lines of a program to solve this problem are as follows:

Select one line of code from each of the 10 pairs of lines of code in the next page and reorder them to solve the above problem.

A1 `data[studentid] = data[studentid] + expense`
A2 `data[studentid] = data.get(studentid, 0) + expense`

B1 `line = studentid + ": " + data[studentid]) + "\n"`
B2 `line = studentid + ": " + str(data[studentid]) + "\n"`

C1 `outfile.write(line)`
C2 `outfile.print(line)`

D1 `for studentid in data.keys().sort():`
D2 `for studentid in sorted(data.keys()):`

E1 `expense = int(item[item.find('$')+1:])`
E2 `expense = int(item.split(':')[1])`

F1 `studentid = line_data[1]`
F2 `studentid = line_data[0]`

G1 `infile.readlines()`
G2 `infile.readline()`

H1 `for item in line_data[4].split(';'):`
H2 `for item in line_data[4].split()`

I1 `for line in infile:`
I2 `for line in infile.readline()`

J1 `line_data = line.split()`
J2 `line_data = line.split(',')`

Select only 10 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:

- G 1

- I 2
- J 1
- F 2
- H 2
- E 2
- A 1
- D 1
- B 1
- C 2

4. (10 points) In the game Alphabear, one wants to find long words that will results in many points. Each character has a *timer*, which is a positive integer. For a string $w_1 w_2 \cdots w_n$, let the *score* of this word to be:

$$(1/c_1 + 1/c_2 + \cdots + 1/c_n) \times 1.1^n$$

where c_1 is the timer of character w_1, and similarly for all other characters.

Write a function that accepts a list of words, and a dictionary of characters as keys and timers as values. Then, return a dictionary with the keys being "valid" words (i.e., those where each character is in the passed in dictionary), and the values being the word's score. For example, suppose that the words were

```
['abcc', 'abbc', 'aaa', 'bad']
```

and the timers were:

```
{'a': 1, 'b': 2, 'c': 3}
```

then `abcc` would receive a score of $(1/1 + 1/2 + 1/3 + 1/3) \times 1.1^4 = 3.17$, because the timer for `a` is 1, the timer for `b` is 2, and the timer for both `c`'s is 3. Since the length of `abcc` is 4, we multiply this by 1.1^4.

Similarly, `abbc` gets a score of $(1/1 + 1/2 + 1/2 + 1/3) \times 1.1^4 = 3.42$, `aaa` gets a score of $(1/1 + 1/1 + 1/1) \times 1.1^3 = 3.99$, and `bad` is an invalid word since the letter `d` is not in the dictionary.

Therefore, this function would return the following dictionary (with the valid words as keys, and scores as the values):

```
{'abcc': 3.17, 'abbc': 3.42, 'aaa': 3.99}.
```

Solution:

```
def alphabear_score(words, timer_dict):
    total_score_dict = {}
    for word in words:
        score = 0.0
        valid_word = True
        for character in word:
            if character in timer_dict:
                score += 1 / timer_dict[character]
            else:
                valid_word = False
                break
        if valid_word:
            total_score_dict[word] = score
        return total_score_dict
```

5. (12 points) Given how cold it has been recently, you are looking forward to warm weather returning. To do so, you decide to investigate weather patterns in Hamilton over the last year. You are given a file containing the temperature for every day of the past year in the following format: `YYYY-MM-DD: high low average`. For example:

```
2019-02-27: 12 2 9.9
2019-05-01: 53 43 48.53
2019-06-12: 76 57 65.71
2019-08-17: 82 64 70
2019-11-01: 54 39 43.35
```

Write a function `days_below` that takes in the filename and a temperature and returns the number of days during the past year where the average temperature was below the given temperature. For example, given the above file and the temperature 50 the function would return 3 (for Feb 27, May 1, and Nov 1.)

Requirement (4 points): your solution must consist of at least two functions, the `days_below` function described above, *plus at least one "helper" function that is called within the body of* `days_below`. This question will be graded both on correctness and program design (SOFA). Docstrings are unnecessary.

Solution:

```
def days_below(fname, temp):
    averages = get_averages(fname)
    day_count = 0
    for day_temp in averages:
        if day_temp < temp:
            day_count += 1
    return day_count

def get_averages(fname):
    temps = []
    with open(fname) as f:
        for line in f:
            data = line.split()
            avg = float(data[3].strip())
            temps.append(avg)
    return temps
```

(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.)