COSC 101, Exam #2
24 October 2017

Name: ______________________________ Section: 8:30 / 9:55 / 1:20 / 2:45

Instructions and advice:

- 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.
- 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	8	
4	10	
5	12	
Total:	50	

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

```
a = 2.68
b = 'four'
c = [3, 0, 1]
d = [a, [1, c, 0], b]
```

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[2][1]`

Solution:

```
'o'
```

(b) `c + a`

Solution:

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

(c) `b[2:3]`

Solution:

```
'u'
```

(d) `len(d[1])`

Solution:

```
3
```

(e) `d[1][0] + a`

Solution:

```
3.68
```

(f) `c + [b[1:3]]`

Solution:

```
[3, 0, 1, 'ou']
```

(g) `d[1][1:3]`

Solution:

```
[[3, 0, 1], 0]
```

(h) `list(b)[int(a)]`

Solution:

```
'u'
```

2. (a) (4 points) What is the output of the following program?

```
word = 'halloween'
number = 12

while number > len(word):
    if number % 2 == 0:
        number -= 3
        print("A:", number, word)
    elif number % 3 == 0:
        print("B:", number, word)
        number -= 1
    else:
        number += 1
        print("C:", number, word)
    word = word[1:]
```

Solution:

```
A: 9 halloween
B: 9 alloween
A: 5 lloween
```

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

```
def edit_list(alist):
    for i in range(len(alist)-1,0,-2):
        if alist[i] in [5, 6, 7, 8, 9]:
            alist[i] = alist[i] % 5
        else:
            alist.insert(0, alist[i] % 5)

a = [7, 1, 6, 4]
b = a
edit_list(b)
print(a[1], b[1])
```

Solution:

```
2 2
```

(c) (4 points) What is the output of the following program if the user enters `'abc'` and `'*-*'`? Note: the quotes are not part of the text entered by the user.

```
def mystery(a, b):
    s = ""
    for i in range(len(a)):
        s2 = ""
        for j in range(len(b)):
            s2 += a[i] + b[j]
        print(s2)
        s += s2
    return s

s1 = input("Give me a string: ")
s2 = input("Give me a string: ")
print(mystery(s2, s1))
```

Solution:

```
Give me a string: abc
Give me a string: *-*
*a*b*c
-a-b-c
*a*b*c
*a*b*c-a-b-c*a*b*c
```

3. (8 points) Write a function called `sum_of_matches` that takes as a parameter a list of numbers. It should return the sum of all numbers in the list that match the previous number in the list.

   For example:

   - `sum_of_matches([5, 5, 7, 9, 9, 7])` returns 14, because 5 and 9 are repeated next to each other in the sequence.
   - `sum_of_matches([9.6, 2.3, 9.6, 2.3])` returns 0 because 9.6 and 2.3 are always different from the previous number in the list.
   - `sum_of_matches([3, 3, 3, 3])` returns 9, becuase all 3 values except the first match a previous number in the list.
   - `sum_of_matches([])` and `sum_of_matches([5])` both return 0, because there are fewer than two elements in each list, and so there can be no matches.

Solution:

```python
def sum_of_matches(L):
    som = 0
    for i in range(1,len(L)):
        if L[i] == L[i-1]:
            som += L[i]
    return som


# an alternate solution using list accumulator
def sum_of_matches_v2(num_list):

    match_list = [] # keep track of all matches
    prev_num = ''   # keep track of number to compare against, start
                    # with something no numbers with match (empty string)

    # check each number in the list against previous number
    for num in num_list:

        # it's a match
        if num == prev_num:
            match_list.append(num)

        # not a match, look for this num next
        else:
            prev_num = num

    return sum(match_list)
```

4. (10 points) Write a function `calculate_grades` that takes two parameters: `test_scores` and `hw_scores`, nested lists containing the names and test or hw scores for each student in a course, for example: `[['Madeline', 85.3, 97, 89, 78], ['Sandra', 92.7, 83.4, 75, 93.7]]`.

   Your function should calculate return a list containing the final course grades for each student in the class, for example: `[['Val', 96.4], ['Matt', 93.7]]`. In this class, all tests and homework assignments are weighted equally. This means that the final grade is the average of all of the scores, both tests and homeworks.

   The list returned by your function should match this example:

```
>>> tests = [['Adi', 100, 90, 95], ['Pavol', 93.2, 80.35], ['Neela', 97, 98]]
>>> hws = [['Adi', 90, 100], ['Pavol', 80, 85, 90], ['Neela', 97.5, 97.5]]
>>> calculate_grades(tests, hws)
[['Adi', 95.0], ['Pavol', 85.71000000000001], ['Neela', 97.5]]
```

   You cannot use the `sum` function. You should assume that both lists contain the students listed in the same order.

Solution:

```
def calculate_grades(test_scores, hw_scores):
    final_grades = []

    for i in range(len(test_scores)):

        name = test_scores[i][0] # student's name
        assignments = 0          # num of scores
        points = 0               # total points

        # add and count all test scores
        for j in range(1, len(test_scores[i])):
            points += test_scores[i][j]
            assignments += 1

        # add and count all hw scores
        for j in range(1, len(hw_scores[i])):
            points += hw_scores[i][j]
            assignments += 1

        # calculate final grade & save to list
        grade = points / assignments
        final_grades.append([name, grade])

    return final_grades
```

```
######## TESTING
##
##exam = [ ['A', 100, 100, 100, 100],
##          ['B', 50, 50, 50, 50],
##          ['C', 100, 50, 100, 50]
##        ]
##
##hw = [ ['A', 90, 90, 90, 90],
##        ['B', 60, 60, 60, 60],
##        ['C', 100, 50, 100, 50]
##      ]
##
### final scores should be: A=95.0, B=55.0, C=75.0
##print(calculate_grades(exam, hw))

tests = [['Adi', 100, 90, 95], ['Pavol', 93.2, 80.35], ['Neela', 97, 98]]
hws = [['Adi', 90, 100], ['Pavol', 80, 85, 90], ['Neela', 97.5, 97.5]]
print(calculate_grades(tests, hws))
```

5. This is a two-part question, the second part is on the next page. Part (a) is a helper function for part (b). Part (b) can be completed even if you have not finished part (a) correctly.

   (a) (5 points) Write a function called `csv_format` that takes a list of lists of strings (e.g., `[['a','b'], ['c'], ['d','e']]`) as a parameter. The function will return a string in comma separated value (CSV) each sublist is on a new line (separated by the `\n` character), and each string in the sublist is separated by a comma. For example, the CSV format of the above list is: `'a, b, \n c, \n d, e, \n'`.

   A second example:

```
>>> alist = [ ['apple', 'banana'], ['pear', 'mango'] ]
>>> csv_format( alist )
'apple, banana, \n pear, mango, \n'
```

   Note: The above examples are shown with spaces after each new line character and comma to make them more readable, your code *should not* produce these spaces.

Solution:

```
def csv_format( in_list ):
    csv = ''
    for sublist in in_list:
        for string in sublist:
            csv += string + ','
        csv += '\n'
    return csv

#### TESTING
##>>> alist = [ ['a', 'b'], ['c'], ['d', 'e'] ]
##>>> csv_format(alist)
##'a,b,\nc,\nd,e,\n'
##>>> blist = [ ['apple', 'banana'], ['pear', 'mango'] ]
##>>> csv_format(blist)
##'apple,banana,\npear,mango,\n'
```

(b) (7 points) You are tasked with analyzing website accesses by employees at your company. You have been provided with a list of website accesses, each sublist contains the IP address of a device and a website accessed from that device. For example:

```
[ ["192.168.21.3", "food.com"], ["192.168.23.21", "amazon.com"],
 ["192.168.21.3", "amazon.com"], ["192.168.23.45", "colgate.edu"] ]
```

You also got a list of devices and the IP addresses assigned to them. For example:

```
[ ["macbook1", "192.168.23.21"], ["chromebook2", "192.168.23.4"],
 ["jtquad", "192.168.21.3"] ]
```

Write a function called `sites_accessed_by_device` that takes these two lists as parameters and returns a string in CSV format. Each line of the CSV will have a device name followed by the websites accessed from that device. The result of the above examples is:

```
'macbook1,amazon.com,\n chromebook2,\n jtquad,food.com,amazon.com,\n'
```

You are required to use the `csv_format` function from part (a) and can assume the function works as described (regardless of whether your answer is correct or not). Note that `list.index(item)` returns the index of the given `item` if that `item` is found in the `list`.

Solution:

```
def sites_accessed_by_device( device_ip_pairs, site_accesses):
    site_accesses_by_device = []
    for dl_index in range(len(device_ip_pairs)):
        device_accesses = [ device_ip_pairs[dl_index][0] ]
        for wa_index in range(len(site_accesses)):
            if device_ip_pairs[dl_index][1] == site_accesses[wa_index][0]:
                device_accesses += [ site_accesses[wa_index][1] ]
        site_accesses_by_device += [ device_accesses ]
    print (site_accesses_by_device)
    return csv_format(site_accesses_by_device)


# Alternatve solution with two separate for loops
# instead of nesteed loops
def sites_accessed_by_device_v2( devices, sites ):
    sites_by_device = []
    device_names = []
    device_ips = []

    for device in devices:
        device_ip = device[1]
        device_name = device[0]
        device_ips.append(device_ip)
        device_names.append(device_name)
        sites_by_device.append([device_name])

    for site in sites:
        ip_addr = site[0]
        site_name = site[1]
        if ip_addr in device_ips:
            idx = device_ips.index(ip_addr)
            sites_by_device[idx].append(site_name)

    return csv_format(sites_by_device)


#### TESTING
##websites_accessed = [ ["192.168.21.3", "food.com"], ["192.168.23.21", "amazon.co
##                      ["192.168.21.3", "amazon.com"], ["192.168.23.45", "colgate
##device_ips = [ ["macbook1", "192.168.23.21"], ["chromebook2", "192.168.23.4"],
##               ["jtquad", "192.168.21.3"] ]
##
##print ('VERSION 1:')
##print (sites_accessed_by_device(device_ips, websites_accessed))
##
##print('VERSION 2:')
##print(sites_accessed_by_device_v2(device_ips, websites_accessed))
```

(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.)