COSC 101, Exam #3
14 November 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:

```
w = 'winter is coming'
x = ['is', 3, 5, 10]
y = {'s': 1, w:len(x), 'x':len(w), 2:x[1:]}
```

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) `sum(x[-1:0:-1])`

(b) `y[len(x[0])]`

(c) `w[:x[-1]]+'cold'`

(d) `len(w) > sum(y[2])`

(e) `x[0] in y.keys()[:3]`

(f) `len(y)-len(x) in y`

(g) `w[:(x[2]+y['s'])]`

(h) `w.split()[1] in x`

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

```
def change_it(start,check):

    ind1 = 0
    ind2 = 0
    start = start[:]
    limit = len(check)
    while len(start) >= limit and len(start) < 10:
        if start[ind1] in check[ind2]:
            start = start + start[:check[ind2][0]%len(start)]
            ind1 = ind1+1 % len(start)
        else:
            start = start[check[ind2][1]%len(start):]
            ind1 = ind1+1 % len(start)
        ind2 = (check[ind2][1]+1) % len(check)

    return start

start = [3, 5, 4, 2]
check = [(1, 3), (0, 4), (3, 3), (0, 4)]
result = change_it(start,check)
print(start, result)
```

(b) (4 points) Suppose the following text is in a file called `test.txt`: (there are no blank lines)

```
I love thanksgiving
          it makes me happy
because I love eating all day
   and seeing my family is my favorite.
```

What is the output of the following program?

```
afile = open('test.txt')
count = 0
for line in afile:
    split = line.split()
    if len(split) % 6 == 0:
        print(" ".join(split[::3]), end=" they are ")
    elif len(split) % 3 == 0:
        print(" ".join(split[::3]), end=" ")
    elif 'happy' in split:
        print("am angry", end=" ")
    else:
        print(" ".join(split[2:4]) + ".")
afile.close()
```

(c) (4 points) What is the output of the following program if the user enters `'bean'` then `'turkey'` then `'cranberry'` then `'bean'`? Note: the quotes are not part of the text entered by the user.

```
def myfunction(string,info):

    new_info = info
    for i in range(len(string)):
        string = input("Enter a string: ")
        if info.get(string):
            new_info[string] += len(string)
        else:
            new_info[string] = len(string)
    return new_info

info = {'turkey':6}
string = 'corn'
info = myfunction(string,info)
print(string, info)
```

3. (8 points) Write a function called `summarize_contents` that takes a filename, opens the file, finds all the lines in which the first and last characters in the line are `'#'`, and writes those lines to a new file called `filename_summary.txt`. The lines added to the file should not contain the `'#'`s or any spaces between the first `'#'` and the remainder of the line. (If the filename were `a.txt` the output filename would be `a_summary.txt`.)

   For example, if the input file contains:

```
# some information #
definition

        # loop

#more info#
statements
```

   The output file would contain:

```
some information
more info
```

4. (10 points) Write a function `accumulate_values` that takes two parameters: `list1` and `list2`. Both lists contain a number of different items that can be of different types. The function returns a dictionary with pairs of items from `list1` and an accumulation of that item.

   Items from `list1` will only appear in the dictionary if they also appear in `list2`. The associated values are the items accumulated however many times the item appears in `list2`. You can assume that none of the items in `list1` is a list.

   For example:

```
>>> list1 = [4, 'six', 'alpha', 2, 4.8]
>>> list2 = [4, 8, 4, 'three', 'alpha', [5, 6], 4.8, 'alpha']
>>> accumulate_values(list1, list2)
{4: 8, 'alpha': 'alphaalpha', 4.8: 4.8}
```

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 `get_course` that prompts the user to enter the number of a course they want to take and returns the course number as a string. The number will only return valid course numbers, which have four-character department codes followed by a space and then a three digit number. For example, `'COSC 101'` is valid but `'Computer Science 1'` is not. Whenever a user enters an invalid course number, the function should display an error message and reprompt them.

(b) (7 points) Write a function called `course_registration` that takes a dictionary of courses enrollments as a parameter. The course enrollment dictionary has course numbers as keys and values are the number of seats available in the course. The user will be prompted to enter the course number for each course they wish to enroll in. (If they enter an invalid course number they will be re-prompted until they provide a valid course number.) The program will return the list of courses that the user was able to enroll in. Students can only enroll in a course if there is at least one seat available. The course enrollment dictionary will be updated to reflect the updated numbers of seats available.

For example, if the course enrollment dictionary is originally {`'COSC 101':2, 'COSC 102':5, 'COSC 201':-1`} and the user indicates they would like to enroll in `'COSC 102'` and `'COSC 201'`, the returned list would be `['COSC 102']` and the dictionary is now {`'COSC 101':2, 'COSC 102':4, 'COSC 201':-1`}.

You are required to use the `get_course` function from part (a) and can assume the function works as described (regardless of whether your answer is correct or not). You may use **`for`** loops in this program.

(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.)

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