Computer Science 1 — CSci 1100
Fall Semester, 2019
Final Exam Overview and Practice Questions

Overview

- The final exam will be held **Monday, December 16, 2019** from 6:30 pm - 9:30 pm. Note that this will be a **two-hour exam**. Students who have provided Profs. Turner or Mushtaque with an accommodation letter and therefore require extra time will either start earlier or end later depending on what arrangements we can make with the registrar. You should watch your email for a message from Shianne Hulbert with the accommodations details.

- For the rest of you, please check Submitty for your room and section assignment. Students taking the test in the wrong room or section may be subject to as much as a 20 point penalty.

- Once you start the exam, you will not be excused from the room for any reason until you turn your exam in. Please make sure you take care of any biological needs before sitting down to take the final.

- You **MUST BRING YOUR ID** to the exam. Missing ids may cause an immediate 20 point penalty if we cannot validate your identity on Submitty. We are sorry for any inconvenience this may cause.

- Exam coverage is the entire semester, except for the following:
  - JSON data format,
  - Images, and
  - Conversion to C++

  You do not need to know the intricacies of **tkinter** GUI formatting, but you should understand the GUI code structure we outlined, be able to trace through event driven code and write small methods that are invoked by the GUI. Consider the lecture exercises for Lecture 22 and the modifications you made to the BallDraw class during Lab 11 as types of questions you might need to answer. Please review lecture notes, class exercises, labs, homework, practice programs, and tests, working through problems on your own before looking at the solutions.

- Each student may bring ONE double-sided 8.5x11 crib sheet to the test.

- Please refer back to the Test 1, Test 2 and Test 3 practice problems for further instructions.

- We will hold two a review sessions on **Thursday, December 12**. We are still trying to confirm a room and a time, but our current guess is that Prof, Turner will hold a review from 9:30-11:00 and Prof. Mushtaque from 11:00-12:30. We are hoping to book DCC 308, but this is out of our hands. We will publish the confirmed time and location to the Submitty discussion forum as soon as we have them. These will be question-and-answer sessions, so bring your questions!

- There are often study events held on campus, for example the Red/White Association often holds a STUDY DAYS event on this coming weekend and UPE often holds tutoring sessions. I do not know of any specific events right now, but we will post anything we learn to the Submitty discussion forum. Please monitor the channel if you are looking for help.

- What follows are a few additional practice problems. These are by no means comprehensive, so rework problems from earlier in the semester. All the material from tests 1, 2, and 3 are also fair game. This is a comprehensive final exam.

- We have separately provided Spring 2017's final exam.

Questions

1. Write a version of `merge` that does all of the work inside the `while` loop and does not use the `extend`.
2. Using what you learned from writing the solution to the previous problem, write a function to merge three sorted lists. For example

   ```
   print(three_way_merge( [2,3, 4,4, 4, 5], [1, 5, 6,  9], [ 6, 9, 13]))
   ```

   Should output

   ```
   [1, 2, 3, 4, 4, 4, 5, 5, 6, 6, 9, 9, 13]
   ```

3. Given a list of test scores, where the maximum score is 100, write code that prints the number of scores that are in the range 0-9, 10-19, 20-29, ... 80-89, 90-100. Try to think of several ways to do this. Outline test cases you should add.

   For example, given the list of scores

   ```
   scores = [ 12, 90, 100, 52, 56, 76, 92, 83, 39, 77, 73, 70, 80 ]
   ```

   The output should be something like

   ```
   [0,9]: 0
   [10,19]: 1
   [20,29]: 0
   [30,39]: 1
   [40,49]: 0
   [50,59]: 2
   [60,69]: 0
   [70,79]: 4
   [80,89]: 2
   [90,100]: 3
   ```

4. Given a list of floating point values containing at least 10 values, how do you find the 10 values that are closest to each other? In other words, find the smallest interval that contains 10 values. By definition the minimum and maximum of this interval will be values in the original list. These two values and the 8 in between constitute the desired answer. This is a bit of a challenging variation on earlier problems from the semester. Start by outlining your approach. Outline the test cases. For example, given the list

   ```
   values = [ 1.2, 5.3, 1.1, 8.7, 9.5, 11.1, 2.5, 3, 12.2, 8.8, 6.9, 7.4,\
         0.1, 7.7, 9.3, 10.1, 17, 1.1 ]
   ```

   The list of the closest 10 should be

   ```
   [6.9, 7.4, 7.7, 8.7, 8.8, 9.3, 9.5, 10.1, 11.1, 12.2]
   ```

5. Consider the following recursive function:

   ```
   def mystery( L, v ):
       if v < len(L):
           x = L[v]
           mystery( L, x )
           print(x)
       else:
           print(v)
   ```

   (a) Show the output from the call:

   ```
   mystery( [2, 5, 4, 7, 1], 0 )
   ```

(b) Write a Python call to the function `mystery` containing a list and an index that causes the recursion to never stop (until the stack overflows). Do this with as short a list as you possibly can.

(c) Write a Python call to the function `mystery` that causes the program to crash (without the problem of infinite recursion):

6. You are given the following function definition with doctest comments embedded.

```
def f(x, y, z = 0, w = False):
    '''
    >>> f([1,2,3], [4,6,8,5,11], 3, True)
    True
    >>> f([1,2,3,4,5], [0,2,6], -1, True)
    False
    >>> f([1,2,3,4,5], [2,3,4])
    False
    >>> f([5,4,5,8], [6,7,8,9,10,12], 2, True)
    False
    '''
    count = 0
    for item in x:
        if w and (item+z in y):
            count += 1
    if count == len(x):
        return True
    elif count == len(y):
        return False
```

When the lines below are executed, indicate whether the tests above will pass or fail and, for the failures, indicate what is returned by `f`.

```
import doctest
doctest.testmod()
```

7. You are given a dictionary called `clubs` where each key is the name of a club (a string), and each value is the set of id strings for the students in the club. Write a segment of Python code that creates a set of all student ids (if any) that are in all the clubs in the dictionary.

8. Write a function called `framed` that takes a string storing a name and creates output with a framed, centered greeting. For example, if the name string is `'Tim'` the output should be

```
**********
* Hello! *
*  Tim   *
**********
```

and if the name string is `'Anderson'` the output should be

```
************
*  Hello!  *
* Anderson *
************
```

9. Consider a file called `addresses.txt` that contains a name and an address on each line of the file. The parts of each name and address are separated by a `'|'`. For example,

```
John Q. Public |  1313 Mockingbird Way  | Walla Walla, Washington 98765
Samuel Adams  | 1431 Patriot Lane | Apartment 6 |   Washington, Oregon 12345
Dale President | 1600 Pennsylvania Ave. | Washington, D.C. 234781
Thomas Washingon | 153 Washington Street | Anywhere, Anystate 53535
```

Notice there is a variable number of '|' in each line, and the string "Washington" appears in many places. Fortunately, the input that follows the last '|' on each line is in the form City, State Zip-code, and there are no commas in the name of the City.

Write a function that is passed the name of a city and returns the number of people stored in addresses.txt that live in the given city. For the above version of addresses.txt and for the city Washington, the function should return the value 2.

10. Write a piece of code that reads from the user an amount between 0 and 1 inclusive, and calculates the number of coins with different value (1 cent, 5 cents, 10 cents, 25 cents) that would be needed to make this amount of cash. Your solution should use the smallest number of coins possible that sum to the number entered. These four coins are the only values you have to worry about and, therefore, you are allowed to "hardcode" these values in. You may also assume that the value entered by the user is a non-negative float and will not have more than two decimal places.

    Three example runs of the program are given below:

    ```
    Please enter an amount between 0 and 1 ==> 0.76
    You need ==> 25 cent: 3   10 cent: 0   5 cent: 0   1 cent: 1

    Please enter an amount between 0 and 1 ==> 0.81
    You need ==> 25 cent: 3   10 cent: 0   5 cent: 1   1 cent: 1

    Please enter an amount between 0 and 1 ==> 0.67
    You need ==> 25 cent: 2   10 cent: 1   5 cent: 1   1 cent: 2
    ```

11. Write a piece of code that asks the user for a string using input that has multiple underscore characters and prints the string in "kerned" form such that the extra underscore characters are removed. For example, given the input:

    ```
    ___a__b_cd__e f_g___h_
    ```

    The program should output (note the space between e and f):

    ```
    a_b_cd_e f_g_h
    ```

    Remember that split returns all strings before and after a given delimiter. For example, '_a__b_c'.split('_') returns ['', 'a', '', 'b', 'c']. As practice, try to solve this using join and a list comprehension.

12. Assume the variable got contains the characters in a TV show and a score of how much they are loved in the form of a list of lists. Return the name of the most loved character (or characters), i.e. with the highest score in the list got. Do not use the sort or the max functions. Given:

    ```
    got = [ ['tyrion',9], ['cersei',3], ['jon',8], ['daenerys',5], ['arya',9], ['sansa',6], ['tywin',4] ]
    ```

    Your program should print:

    ```
    tyrion arya
    ```

13. You are given the class Die that is defined below:

    ```
    import random

    class Die(object):
        def __init__(self, sides):
            sides = int(sides)
            if sides < 1:
                sides = 1
            self.sides = sides
            self.roll()
    ```

```
    def roll(self):
        # random.randint(a, b) returns a random value, x, where a <= x <= b
        self.value = random.randint(1, self.sides)
```

Write a program that uses the `Die` class to create two Die objects. The first die object should be a 6-sided die. The second die object should be a 10-sided die.

Roll the two dice until they both have a value of 1, otherwise known as "snake eyes", and print out the total number of rolls it took for both dice to have a value of 1. Assume that eventually both of the dice will have a value of 1, terminating your program.

14. Write a function that reads integers from a file (call it `integers.txt`), storing them in a list. It should stop and return the list when there are no more integers or when a negative integer is found. Each input line contains just one integer.

15. Write a function that takes as input an unordered list, and returns the number of values that would remain in the list after all duplicates are removed. Examples:

```
>>> find_dup([1, 5, 3, 6, 6, 3, 1])
4
>>> find_dup([3, 1, 2, 4])
4
>>> find_dup([2, 1, 2, 1, 1, 2, 2])
2
>>> find_dup([])
0
```

Note that this problem is very easy to solve using sets, but you can solve it without sets as well. Try to come up with both solutions.

16. Write a function called `notused` that takes a list of words as its single parameter, and returns a set containing the letters of the English alphabet that are not used by any of the words in the input list. Your function must use sets. Here is an example of how your function should work:

```
>>> notused([ "Dog", "pony", "elephant", "Tiger", "onyx", "Zebu" ])
set(['c', 'f', 'k', 'j', 'm', 'q', 's', 'w', 'v'])
```

Hint: you can use the following set in your solution:

```
all_letters = set("abcdefghijklmnopqrstuvwxyz")
```

17. In the iterative version of merge sort we discussed in class, the code starts by dividing the list to be sorted into a list of lists, each of length 1. The actual Python sort function starts instead by looking for "runs" — consecutive values in the list that are already in order. Each run is an initial list for merge sort. For example, the list

```
  L = [ 15, 3, 6, 19, -1, 7, 9, 3, 5 ]
```

would be divided into the list of lists

```
  lists = [ [15], [3,6,19], [-1,7,9], [3,5] ]
```

Write a function called `runs` that takes `L` as an argument and returns `lists`.

18. Write a version of binary search called *trinary search* where the search interval is split into thirds instead of in half. As with binary search, this function should return the index of the location where the first value `x` either is stored in the list or where it should be inserted if `x` is not there. Before starting on this, please make sure you can do the hand simulations of binary search so that you understand the importance of `low`, `mid` and `high` in the code.

19. (This problem was essentially given as a lecture exercise, but it had appeared on an earlier exam.) Below you will find the merge function from class. Show how to modify it so that when a value that appears in both lists L1 and L2 it only appears once in L. You may assume that there are no duplicate values in L1 and no duplicate values in L2. As an example, if

```
L1 = [ 1, 3, 5, 6, 7, 8, 10 ]
L2 = [ 5, 6, 10, 13, 14 ]
```

then after the merge

```
L = [ 1, 3, 5, 6, 7, 8, 10, 13, 14 ]
```

```
def merge(L1,L2):
    i1 = 0
    i2 = 0
    L = []
    while i1<len(L1) and i2<len(L2):
        if L1[i1] < L2[i2]:
            L.append(L1[i1])
            i1 += 1
        else:
            L.append(L2[i2])
            i2 += 1
    L.extend(L1[i1:])
    L.extend(L2[i2:])
    return L
```

```
def merge(L1,L2):
```

20. The list sort function takes an optional comparison function as an argument. For example, after the following code list L is in *decreasing order.*

```
def rev(x):
    return -x
L = [1, 10, 3, 6]
L.sort(key=rev)
print(L)
```

Would print

```
[10, 6, 3, 1]
```

In order to achieve this, the comparison function negates the key value.

Write a function cmp_string that can be used to order a list of words (comprised of only lower case letters) primarily by length. Words that are the same length should be in alphabetical order. For example, a correct version of this function should result in the following output

```
>>> L = [ "apple", "ball", "car", "card", "basket", "car", "honey" ]
>>> L.sort(key=cmp_string)
>>> print(L)
['car', 'car', 'ball', 'card', 'apple', 'honey', 'basket']
```

Your key function will need to return a tuple.

21. In order to sort a list of 2d point coordinates, `[x,y]`, the Python sort function sorts by the first, `x`, coordinate in the list, and then breaks any ties by looking at the second, `y`, coordinate. If you want instead to sort by the `y` coordinate and then by the `x` coordinate, you need to specify an alternate `key` function to the sort. Write two different `key` routines – the first, called `return_y` that given a 2 element list returns just the `y` coordinate and a second called `flip_coordinates()` that given a 2 element list returns a new list with the `x` and `y` coordinates swapped. Now using each of these key functions, write code to sort a list by the `y` coordinate first and then the `x` coordinate. Note that for the `return_y()` function you will need to take advantage of the **stable sort** property of the Python sort.

As examples,

```
print return_y( [1,3])   # outputs 3
print return_y( ([1,6])   # outputs 6
print flip_coordinates( [1,3])   # outputs [3,1]
print flip_coordinates( [1,6])   # outputs [6,1]
```

Using either of your sorts, the list

```
L = [ [11,2], [5,8], [5,2], [12,3], [1,3], [10,2], [12,1], [12,3] ]
```

will be sorted as:

```
... Sort Code ...
print(L)

[[12, 1], [5, 2], [10, 2], [11, 2], [1, 3], [12, 3], [12, 3], [5, 8]]
```

Now rewrite your solution to the previous question to eliminate `return_y` and `flip_coordinates()` by using lambda functions.

22. Given a list of point coordinates, write a function that converts from cartesian `(x, y)` to polar `(angle, radius)`, then use `map` to covert each entry in a list of points from cartesian to polar. Note that to get angle in degrees you need to calculate `atan2(y, x) * 180 / p`.

```
points = [(-1, 1), (3, 4), (2, -3)]
```

your code would generate:

```
[(135.0, 1.4142135623730951), (53.13010235415598, 5.0), (-56.309932474020215, 3.605551275463989)]
```

Now replace the cartesian_to_polar function with a `lambda` function to generate the same answer, and then use a `list comprehension` to get the same result.

23. Given a list of words as strings, use `filter` and a `lambda` function to return all of the words that contain the string `'ue'`. For example, given:

```
words = ['python', 'queue', 'blue', 'coconut', 'true', 'grail']
```

your code would generate:

```
['queue', 'blue', 'true']
```

Then use a `list comprehension` to get the same result.

24. Write a recursive function to generate the greatest common denominator of two numbers using Euclid's algorithm which says that: Given two numbers `x` and `y` with `x > y`, return `y` if `y` divides `x`, otherwise, calculate the greatest common deniminator of `y` with the remainder of `x / y`.

Once you have the recursive version written, rewrite it not using recursion.

25. Given the following code, what is the output of the call `x(3)`? What is the base case?

```
def x(y):
    if y < 10:
        print(y*'=')
        x(y+1)
        print(y*'+')
    else:
        print(y*'*')
```