EDITED FOR FALL 2018 PRACTICE FINAL EXAM

University of California, Berkeley – College of Engineering
Department of Electrical Engineering and Computer Sciences
Spring 2018 Instructor: Prof. Gerald Friedland 2018-05-09

Computational Structures in Data Science, CS88 Final Exam

Last Name (Please print clearly)		
First Name (Please print clearly)	SOLUTIONS	
Student ID Number		
What time is your lab on Wednesday?		
Name of the person to your: Left | Right		
All my work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS88 who haven't taken it yet. **(please sign)**		

Instructions

- Don't Panic! This booklet contains 13 pages including this cover page. Put all answers on pages 2-13; you can use page 13 for extra/doodle space. Please don't hand in any stray pieces of paper. If we provide lines, they are a hint of how long our solution is. Your solution can be longer or shorter.
- Please turn off all pagers, cell phones and beepers. Remove all hats and headphones.
- You have 120 minutes to complete this exam. The final is closed book, no computers, no PDAs, no cell phones, no calculators, but you are allowed two double-sided sheets of notes, the midterm study guide and the final study guide. There may be partial credit for incomplete answers; write as much of the solution as you can. When we provide a blank (different from multiple lines), please fit your answer within the space provided.
- You are allowed to use standard Python data structures for programming questions.
- Remember: Whatever your score in this exam – it counts 20% of the total grade. If you are caught cheating, however, it's an F and we will have to report it. Good luck!

Question	1	2	3	4	5	6	7	**Total**
Points	5	5	6	6	6	12	6	46

Student ID: _____

Warm-up Questions with short answers (1pt each)

Please write your answer within the designated boxes.

Question 1a: Describe the concept of inheritance and state which programming paradigm is it part of.

Classes can inherit methods and attributes from parent classes to extend into their own class. It is a part of object oriented programming.

Question 1b: Give an example of a grammar that cannot be expressed with a regular expression.

AB -> A

There are many examples. See also Lecture #13, Slide 11.

Question 1c: Fill in the gaps with some of the following words so that the sentences are correct: class, object, instance, object, general, specialized, constructor, destructor, inherits, overloads.

An Object is an ___instance____ of a class. The first method that needs to be called in a new object is called __constructor_____. A child class ____inherits____ from a super class or parent class. The child class is more __specialized_____ than the parent class.

Question 1d: Give an example of a situation where it would be advantageous to use a set instead of a list. Explain why.

When you want to remove duplicates from your data. This is because sets only keep unique items while lists keep all items. Additionally, sets are unordered and sets can only contain hashable items so lookup is more efficient, sets have built-in functions like union and intersect that lists don't have.

Student ID: _____

Question 2: *Generating a Few Fibs* (5 pts)

Define an infinite generator function that generates the Fibonacci series.

The Fibonacci series is defined as follows:
F(n) = F(n-1) + F(n-2) with the seed values F(0) = 0 and F(1) = 1
This produces a sequence of numbers: 0,1,1,2,3,5,8,13, …

```
def Fibonacci_sequence():
    a, b = 0, 1
    while True:
        yield a
        b = a+b
        yield b
        a = a+b

    Alternate:
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a+b
```

Student ID: _____

Question 3: *Picking Your New Roommates* (6 pts)

Use the two tables below, potential roommates *p_roommates* and current roommates *c_roommates*, for the questions in this section.

p_roommates

names	niceness	cleanliness	talkativeness
Anna	5	2	3
Jean	3	5	5
Katy	2	1	2
Bailey	5	5	4

c_roommates

name	niceness	cleanliness	talkativeness
Andrea	2	4	2
Jane	4	2	3
Eda	3	3	0
Sherry	1	5	1
Lauren	2	4	3
Sheila	3	2	2
Joyce	4	1	4

Student ID: _____

a. (2 pts) Fill in the blanks for the SQL query below. The query returns a table with 2 columns: name of one potential roommate and the name of one current roommate who can be matched together. The rule is this: the new roommate match's cleanliness and niceness score must add up to above 4. The column names don't matter.

```
SELECT p.names, c.name
FROM p_roommates as p, c_roommates as c
WHERE p.niceness + p.cleanliness > 4
GROUP BY _____;
```

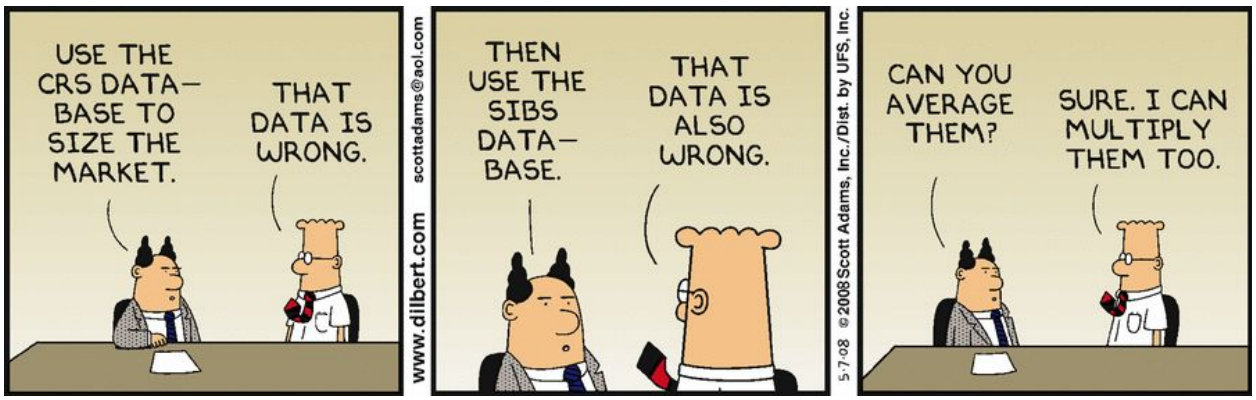
b. (2 pts) What does the following query return? Write down all output values AND column names. You may not need all the lines.

```
SELECT names
FROM p_roommates
WHERE cleanliness > avg(cleanliness);
```

Name
Jean
Bailey

c. (2 pts) Write a query to find the current roommate with the lowest niceness score.

```
SELECT c.name
FROM c_roommates as c
WHERE c.niceness = min(c.niceness);
```



Student ID: _____

Question 4: *Bugs, Bugs, Bugs!* (6 pts)

Each of the functions below has an error that causes the function to have different behavior than what is specified in the docstring. The output of the error is given to you. Use the code and error output to explain the bug in 2 sentences or less. *Note:* Violation of an assert statement does not count as a valid bug. For example, calling countdown with a negative number should error due to the assert statement. This is not a bug.

a. (2 pts)

```
def convert_to_tuple_list(d):
"""
>>> convert_to_tuple_list({“hello”:1, “world”:2})
[(“hello”, 1), (“world, 2)]
"""
    assert type(d) == dict
    tuple_lst = [(0, 0) for key in d]
    i = 0
    for key in d:
        tuple_lst[i][0] = key
        tuple_lst[i][1] = d[key]
        i += 1
    return tuple_lst
```

Error message: TypeError: 'tuple' object does not support item assignment

Explanation:

Tuples are immutable, so you cannot mutate the tuples inside tuple_lst.

b. (2 pts)

```
def countdown(n):
"""
>>> countdown(3)
3
2
1
Blastoff!
```

Student ID: _____

```
    """
        if n == 0:
            print("Blastoff!")
            return
        assert n > 0
        print(n)
        return countdown(n+1)
```

Error message: RuntimeError: maximum recursion depth exceeded while calling a Python object

Explanation:

The value of n is being incremented in the recursive call, instead of decremented.

c. (2 pts)

```
    def vending_machine():
     """
     >>> vm = vending_machine()
     >>> vm(10)
     items sold: 10
     >>> vm(1)
     items sold: 11
     """
        num_sold = 0
        def vend(count):
            num_sold += count
            print("items sold: " + str(num_sold))
            return vend
```

Error message: UnboundLocalError: local variable 'num_sold' referenced before assignment

Explanation:

The value of num_sold cannot be modified inside the inner function without a nonlocal statement.

Student ID: _____

Question 5: *WWPP* (12 pts, 2 each)

Consider the following class definitions:

```
class Phone:
	def __init__(self, owner, number):
		self.owner = owner
		self.number = number
		self.power = 100
		self.apps = []

	def send_call(self, other):
		print("Calling " + other.owner + " at " + str(other.number))
		other.receive_call(self)

	def receive_call(self, other):
		print("Hello, this is " + self.owner +".")

class Cellphone(Phone):
	def __init__(self, owner, number):
		Phone.__init__(self, owner, number)

	def receive_call(self, other):
		if other.power:
			print("Hello, this is " + self.owner + ".")
		else:
			print("... There was no ring.")

	def charge_phone(self):
		self.power = min(self.power + 10, 100)
		if self.power == 100:
			print(self.owner + "'s phone fully charged.")
		else:
			print(self.owner + "'s phone at " + str(self.power)
				+ " percent.")

class Smartphone(Cellphone):
	def __init__(self, owner, number):
		Cellphone.__init__(self, owner, number)
		self.apps = ["messaging", "maps", "web browser", "address book"]

	def surf(self):
		if "web browser" in self.apps:
			print("Surfing the web")
			self.power = max(self.power-10, 0)

gerald = Cellphone("Gerald", 1365478910)
jobel = Smartphone("Meghna", 5554446666)
```

Student ID: _____

For each of the following, fill in what Python would print. Some might take more than a line, some might not need every line. If there is an error, specify what kind of error. If Python wouldn't print anything, leave the lines blank. The commands are entered in alphabetical sequence on the Python command line.

a. `>>> gerald.charge_phone()`

```
Gerald's phone fully charged.
```

b. `>>> gerald.apps`

```
[]
```

c. `>>> jobel.surf()`
`>>> jobel.surf()`
`>>> jobel.charge_phone()`

```
Surfing the web
Surfing the web
Meghna's phone at 90%.
```

d. `>>> gerald.surf()`

```
Error: Cellphone object does not have method surf
```

e. `>>> jobel.apps()`

```
TypeError: 'list' object is
not callable
"Can't call attribute" is okay
```

f. `>>> gerald.send_call(jobel)`

```
Calling Meghna at 5554446666
Hello, this is Meghna.
```

Student ID: _____

Question 6: *Stem and Leaf* (6 pts)

We want to implement a function **stem_and_leaf**, which takes a string of sorted integers. A stem-and-leaf plot is a special table where each data value is split into a "stem" (the tens digit) and a "leaf" (the ones digit). **stem_and_leaf** should return a dictionary, where the keys are stems and values are lists of leaves. For example:

23, 25, 25, 27, 28 32, 36, 36, 45, 50

The stem-and-leaf plot for the above data:

Stem	Leaf
2	3 5 5 7 8
3	2 6 6
4	5
5	0

`stem_and_leaf("23 25 25 27 28 32 36 36 45 50")` would return the following dictionary:

```
{
      2: [3, 5, 5, 6, 8]
      3: [2, 6, 6]
      4: [5]
      5: [0]
}
```

However, we can't assure that users will only input integers with two digits. Instead, we write a helper function called **len_two_only** that will eliminate all of the non 2 digit numbers from the string and output a list of integers containing only the 2 digit numbers. For example, given the following string of numbers: "1 23 25 25 27 28 32 360 909 1552 22011", **len_two_only** will return [23, 25, 25, 27, 28, 32]. We've written most of the helper function but need your help to think a pattern to remove from the string. Hint: you can use built-in methods String methods like `len` and `split`.

Student ID: _____

a. (2 pts) Fill in the `len_two_only` function that takes a string of numbers as input and returns a list of two digit numbers. Your solution will return a list of integers (not strings).

```
def len_two_only(numbers):
    """
    >>> two = len_two_only("6 10 12 21 21 21 85 99 100 619")
    >>> two
    [10, 12, 21, 21, 21, 85, 99]
    >>> type(two[0])
    <class 'int'>
    """
    list_of_nums = numbers.split()
    # find all numbers of length 2 and convert values to int
    return [int(x) for x in list_of_nums if len(x) == 2]
```

b. (4 pts) Complete the function definition for **stem_and_leaf**. Assume that you have a correctly-written **len_two_only** function.

```
def stem_and_leaf(numbers):
    """
    >>> d = stem_and_leaf("6 10 12 21 21 21 85 99 100 619")
    >>> d
    {1: [0, 2], 2: [1, 1, 1], 8: [5], 9: [9]}
    >>> d[1]
    [0, 2]
    >>> 3 in d
    False
    """
    result = {}
    for num in len_two_only(numbers):
        if num // 10 in result:
            result[num // 10] += [num % 10]
        else:
            result[num // 10] = [num % 10]
    return result
```

Student ID: _____

Doodle/Notes/Extra Space: