Lecture 17, Thu 12/06

Final Review

CS 8 Final Exam Review

- Final Exam: Tuesday 12/11, 8am - 11am, BUCHN 1920
- Seating is assigned: check the **seat assignment for F18**
- Check the **layout of the room** to locate your seat
- Link to code written in lecture: **https://github.com/ucsb-cs8-f18/cs8-f18-lecture-code**
- Exam will be longer than the midterm (~ twice as long, ~ 2 hours)
- Exam is cumulative (covers everything from print statements to complex data structures)
- Logistics
    - Bring writing utensil (dark led or pen)
    - Bring your student ID
    - No electronic devices
    - Closed book andonly one sheet of notes. No electronic devices
- Structure of final is similar to the midterms. Types of questions:
    - Evaluate expressions
    - Evaluate types
    - Given code, what is the output
    - Short answer / definitions
    - Read / write assert statements and test code
    - Complete function definitions / write python statements
    - Fill-in-the-blank
        - complete function definition key terms
        - pass in specific paramaters
        - etc.
    - General Advice
        - Read instructions carefully - pay close attention to what is being asked
        - Double-check your work

```
'''
## Advice on how to prepare
    - Lecture notes (important to know topics, examples,
    concepts).
    - Review code written in lecture
      - Type out the code and understand the output
    - Understanding labs and being able to implement them
    - Reading the textbook for additional details and
    understanding
        - Do the examples to solidify understanding
    - Homework exercises
    - Prototyping - "I wonder how python behaves when ..."
        - write a simple example
        - Helps with code practice as well as understanding the language and edge cases

## Overview of topics covered after midterm 2

* File I/O
    - Read file (infile = open('example.txt', 'r'))
        - infile.read()
        - infile.read(n)
        - infile.readlines()
        - infile.readline()
        - for a_line in infile
    - Write file (outfile = open('example.txt', 'w'))
    - Append file (outfile = open('example.txt', 'a'))
        - outfile.write(somestring)
```

```
* Dictionaries
    - key / value pairs
    - Creating a dictionary
    - Adding to a dictionary
    - dictionary methods
        - .pop(key)
        - .get(key)
        - .keys()
        - .values()
        - .items()

* Sets
    - Collection of items with no duplicates
    - Creating an empty set or a set from a list
    - Set operators
        - in, not in, combine(|), intersection(&), difference(-), unique(^)
    - Set comparisons
        - ==, !=, proper subset (<), subset (<=)
    - Set methods
        - .add, .clear


* Recursion
    - Properties of recursion
        - base case
        - recursive calls getting "closer" to base case
    - Examples in class and lab
        - print
        - reconstruct lists and strings (reverse a string or list)
        - computing values (factorial, fibonacci, ...)
        - etc.

## Topics covered before midterm 2 (refer to previous lecture notes)
- Python data types
- Arithmetic
- Python built-in functions
- Comparison operators
- Boolean operators
- Strings
- Lists
- Tuples
- User-defined functions
- Namedtuples
- Testing (assert / pytest)
- For loops
- Nested control structures
- Accumulator Patterns
- Nested (double) for loops
- While loops
    - Break, continue, pass
- 2D Lists
- String functions and formatting
- Random
'''
```

Name: *(as it would appear on official course roster)*
Umail address: **@umail.ucsb.edu**
Optional: name you wish to be called if different from name above.
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone"

1

h13: Perkovic 10.1 and 10.2 (Recursion)

h13
CS8 F18

ready?	assigned	due	points
true	Tue 11/27 07:00AM	Wed 12/05 09:00AM	100

You may collaborate on this homework with AT MOST one person, an optional "homework buddy".

MAY ONLY BE TURNED IN IN THE LECTURE/LAB LISTED ABOVE AS THE DUE DATE.
There is NO MAKEUP for missed assignments, and you may not submit work in advance, or on behalf of another person.
In place of that, we drop the four lowest scores (if you have zeros, those are the four lowest scores.)

Read Chapter 10, sections 10.1 and 10.2 and the lecture notes. Then complete these problems and turn in your completed homework during lecture

Please:

- **No Staples**.
- **No Paperclips**.
- **No folded down corners**.

1. (10 pts) Please fill in the information at the top of this homework sheet, as usual. WRITE DARK, and remember, if you MUST submit it on multiple sheets, JUST write your name at the top of both sheets and turn in both sheets UNCONNECTED. No staples, paper clips, fold/tear etc or anything that would jam up the scanner.

2. (10 pts) What is the significance of a “base case” in a recursive function?

3. On page 332, the author talks about two important properties of recursive functions: * There exist one or more base cases which provide the stopping condition for the functions * One or more recursive calls on arguments that are “closer” to the base cases (progress to the base case)

   For each of the following implementations of `count(n)`, indicate which of the two properties are satisfied by circling the appropriate option. Then write the output when the function is called as `count(4)`. If no output is produced, circle the “No output” option. If the execution never ends, circle “execution never ends” and write the first four characters printed. You may circle multiple options as appropriate. Finally write if either there is no output or execution never ends (or both), explain why that happened in light of the recursive properties that the function did not satisfy

   a. (10 pts)

```
def count(n):
  print(n)
  count(n-1)
```

Base case(s) exist Progress to base case

No output produced Execution never ends Output:

b. (10 pts)

```
def count(n):
  count(n-1)
  print(n)
```

Base case(s) exist Progress to base case

No output Execution never ends Output:

c. (10 pts)

```
def count(n):
  if n < 0:
    return
  count(n-1)
  print(n)
```

Base case exists Progress to base case

No output Execution never ends Output:

d. (20 pts)

```
def count(n):
  if n <= 0:
    print("Blast off!")
    return
  print(n)
  count(n)
```

Base case exists Progress to base case

No output produced Execution never ends Output:

e. (20 pts)

```
def count(n):
  if n <= 0:
    return 0
  result = n + count(n-1)
  print(result)
  return result
```

Base case exists Progress to base case

No output Execution never ends Output:

4. (10 pts) Consider the code in part(e) of the previous question. Show all the variables (and their values) in the program stack when the execution of the function reaches the base case, specifically right before the `return` statement in the `if n<=0:` clause is executed.

Name: *(as it would appear on official course roster)*	
Umail address:	**@umail.ucsb.edu**

1

EXAM: le01: Lab Exam 01 Version B

le01

CS8 F17

ready?	date	points
true	Mon 11/13 10:00AM	50

You may not collaborate on this exam with anyone. If you need to use the restroom, you must leave your cell phone with the exam proctor before leaving the room.

- Write your name at the top of every page
- Double check that you turned in ALL pages; look for "End of Exam" on the last page.
- This exam is **closed book, closed notes**, **closed mouth, cell phone off.**
- You are permitted **one sheet of paper** (max size 8.5x11") on which to write notes.
- This sheet will be collected with the exam, and might not be returned.
- Please write your name on your notes sheet.

Please refer to the general instructions on the handout for this exam.

1. (25 pts) For `EXAM_lf01`, please submit a file called `lf01.py`. This file should contain the definition of a Python function called `startsWithZ` that takes one parameter, `name`. If `name` is a string that starts with the letter `Z` or `z`, the function should return a Boolean value True. If name is not a string, or if it is an empty string or if it does not start with lower or upper case 'z', the function should return False.

   Be sure to test your code thoroughly with pytest before submission. You may add test code to the same file. Here are a few sample test functions.

```
def test_startsWithZ_1():
    assert(startsWithZ("") ==  False)

def test_startsWithZ_2():
    assert(startsWithZ("Xerox") ==  False)

def test_startsWithZ_3():
    assert(startsWithZ("zoo") ==  True)

def test_startsWithZ_4():
    assert(startsWithZ("Zoo") ==  True)

def test_startsWithZ_5():
    assert(startsWithZ("oz") ==  False)
```

2
le01
CS8 F17

2. (25 pts) For `EXAM_lf02`, please submit a file called `lf02.py`.

   This file should contain the definition of a Python function called `combineNameLists` that takes two parameter, `flist` and `slist`.

   If `flist` and `slist` are of type `list`, and contain only strings, then `flist` is expected to be a list of strings containing first names, and `slist` is expected to be a list of strings containing last names. Your function should return a new list, containing full names that are constructed by combining each first name in `flist` with the corresponding last name in `slist` separated by a space.

   If either `flist` or `slist` is not of type `list`, or contains anything at all that is not a string, the function should return the value `False`.

   The function should also return False if `flist` and `slist` don't have the same number of elements

   For example, the function should pass the following test cases:

```
def test_combineNameLists_0():
    assert(combineNameLists("James", "Bond") ==  False)

def test_combineNameLists_1():
    assert(combineNameLists(["James"], "Bond") ==  False)

def test_combineNameLists_2():
    assert(combineNameLists(["James", "Joe"], ["Bond"]) ==  False)

def test_combineNameLists_3():
    assert(combineNameLists(["James", "Joe"], ["Bond", 123]) ==  False)

def test_combineNameLists_4():
    assert(combineNameLists(["James", "Joe"], ["Bond", "Smith"]) ==  ["James Bonc

def test_combineNameLists_5():
    assert(combineNameLists(["Al", "Hillary", "Barrack"], ["Gore", "Clinton", "Ob
```

End of Exam

Name: *(as it would appear on official course roster)*
Umail address: @umail.ucsb.edu

1

EXAM: le01: Lab Exam 01 Version C

le01

CS8 F17

ready?	date	points
true	Mon 11/13 12:00PM	50

You may not collaborate on this exam with anyone. If you need to use the restroom, you must leave your cell phone with the exam proctor before leaving the room.

- Write your name at the top of every page
- Double check that you turned in ALL pages; look for "End of Exam" on the last page.
- This exam is **closed book, closed notes**, **closed mouth, cell phone off.**
- You are permitted **one sheet of paper** (max size 8.5x11") on which to write notes.
- This sheet will be collected with the exam, and might not be returned.
- Please write your name on your notes sheet.

Please refer to the general instructions on the handout for this exam.

1. (25 pts) For `EXAM_lf01`, please submit a file called `lf01.py`. This file should contain the definition of a Python function called `endsWithA` that takes one parameter, `name`. If `name` is a string that ends with the letter `A` or `a`, the function should return a Boolean value True. If `name` is not a string, or if it is an empty string or if it ends with a letter other than lower or upper case `a`, the function should return False.

   Be sure to test your code thoroughly with pytest before submission. You may add test code to the same file. Here are a few sample test functions.

```
def test_endsWithA_0():
    assert(endsWithA("Diba") ==  True)

def test_endsWithA_1():
    assert(endsWithA("") ==  False)

def test_endsWithA_2():
    assert(endsWithA("Xerox") ==  False)

def test_endsWithA_3():
    assert(endsWithA("hoopla") ==  True)

def test_endsWithA_4():
    assert(endsWithA("ANNA") ==  True)

def test_endsWithA_5():
    assert(endsWithA(123) ==  False)
```

2. (25 pts) For `EXAM_lf02`, please submit a file called `lf02.py`.

This file should contain the definition of a Python function called `addNumLists` that takes two parameter, `alist` and `blist`.

If `alist` and `blist` are of type `list`, and contain only numeric types (int or float), then return a new list where each element in the new list is the arithemetic sum of the elements in `alist` and `blist` at the same index.

If either `alist` or `blist` is not of type `list`, or contains anything at all that is not numeric, the function should return the value `False`.

The function should also return False if `alist` and `blist` don't have the same number of elements

For example, the function should pass the following test cases:

```
def test_addNumLists_0():
    assert(addNumLists(10, 20) ==  False)

def test_addNumLists_1():
    assert(addNumLists([10], [20]) ==  [30])

def test_addNumLists_2():
    assert(addNumLists([10, 15], [20]) ==  False)

def test_addNumLists_3():
    assert(addNumLists([10, 15.5], [20, 52.6]) ==  pytest.approx([30, 68.1]))

def test_addNumLists_4():
    assert(addNumLists(["James", 10], ["Bond", 20]) ==  False)

def test_addNumLists_5():
    assert(addNumLists([-1, -2, -3], [1, 2, 3]) ==  [0, 0, 0])
```

End of Exam

Name: *(as it would appear on official course roster)*	
Umail address:	**@umail.ucsb.edu**

1

e02

CS8 F18

EXAM: e02: Midterm 2 Exam

ready?	date	points
true	Thu 11/15 08:30AM	100

You may not collaborate on this exam with anyone. If you need to use the restroom, you must leave your cell phone with the exam proctor before leaving the room.

- Write your name at the top of this page **AND EVERY ODD NUMBERED PAGE.**
- Double check that you turned in ALL pages; look for "End of Exam" on the last page.
- This exam is **closed book, closed notes**, **closed mouth, cell phone off.**
- You are permitted **one sheet of paper** (max size 8.5x11") on which to write notes.
- This sheet will be collected with the exam, and might not be returned.
- Please write your name on your notes sheet.

NOTE: All references to **Python** on this exam mean **Python 3**, so you should answer accordingly.

1. Please refer to the handout that came with this exam, and find the **Example `pytest` test cases from lab03** at the top of page 1 of the handout.

   Provide brief answers to these questions:

   a. (2 pts) What is the specific problem that arises with the third test case, but not with the first two?

   b. (3 pts) Rewrite the third test case below to address that problem?

2. Read the following functions:

```
def foo(x):
  for c in x:
      print(c*2)

def bar(x):
  for c in x:
      return c*2

def fooBar(x, y):
  result = y
  for c in x:
      result = result + c*2
  return result
```

2

e02

CS8 F18

Write the output of each of the following calls to the functions `foo`, `bar` and `fooBar` in the table below. Write "Error" if the call results in an error.

	Expression	Result		Expression	Result
(2 pts)	`foo([10, 20])`		(2 pts)	`fooBar([10, 20], 0)`	
(2 pts)	`foo('ape')`		(2 pts)	`fooBar('ape', '')`	
(2 pts)	`foo(10)`		(2 pts)	`fooBar(10, 0)`	
(2 pts)	`foo((4, 2, 0))`		(2 pts)	`fooBar((4, 2, 0), 0)`	
(2 pts)	`bar([10, 20])`		(2 pts)	`fooBar('ape', 0)`	
(2 pts)	`bar('ape')`		(2 pts)	`fooBar([10, 20], 5)`	
(2 pts)	`bar((4, 2, 0))`		(2 pts)	`fooBar(['ape','car'], '')`	

3. (10 pts) Assume that `wizard` is a variable that contains the string 'Harry Potter is back!'. Write the result of each of the following Boolean expressions (either True or False)

Points	Expression	Result
(2 pts)	`type(wizard)==str`	
(2 pts)	`not (len(wizard)> 5)`	
(2 pts)	`wizard.count('t') < wizard.count('r')`	
(2 pts)	`len(wizard[0:]) > 1 and (wizard[-1]=='!')`	
(2 pts)	`wizard[0] or (wizard[1] > wizard[2])`	

4. (5 pts) What is the output of each of the following code?

```
fruits = ["apple","banana","pear","grape"]
for i in range(9, 13):
   print('{0:2} {1:6}s'.format(i,fruits[i-9]))
```

Name: *(as it would appear on official course roster)*	
Umail address:	**@umail.ucsb.edu**

3

e02

CS8 F18

5. (20 pts) Write the definition of a Python function called `containsNVowels` that takes two parameters, `word` and `N`. If `word` is a string that contains exactly `N` vowels (any of the characters `aeiuo`in either upper case or lower case), the function should return `True`, otherwise it should return `False`. If `word` is not a string, or `N` is not an integer or negative, the function should return `False`.

4

e02

CS8 F18

6. On the handout, you'll find several attempts at writing a function that returns the index of the smallest odd integer in a list. These attempts are labeled as `indexOfSmallestOdd_a`, `indexOfSmallestOdd_b`, etc. Please locate these implementations on your handout.

   These functions are intended to operate as follows:

   - If the parameter `alist` is not a list, is empty, or contains anything that is not of type `int`, then return `False`.
   - Otherwise, return the index of the smallest odd number in the list.
   - If there are multiple copies of the smallest odd value in the list, return the index of the one that appears earlier.
   - If there are no odd numbers in the list, return `-1`.

   That is how the functions are supposed to work. However, any or all of them **may or may not contain bugs**.

   Your job is to *do what Python would do* with this code, i.e. indicate the output of the function call shown.

   Assume that it has been loaded into `idle3` and that we've selected `Run Module` (or pressed F5.)
   Then we typed in the Python shell, the print statement that contains the function call shown, and something is printed as a result.

   Which of the answers shown matches what is printed? Put a check mark (✔) in the appropriate column. The first is done for you as an illustration.

Points	Function Call	0	1	2	3	None	False	Python error message	something else
(0 pts)	`print(indexOfSmallestOdd_a([9,10,20,40]))`	✔							
(2 pts)	`print(indexOfSmallestOdd_a([2,10,11,7]))`								
(2 pts)	`print(indexOfSmallestOdd_a([2,7,5,10]))`								
(2 pts)	`print(indexOfSmallestOdd_a([2,4,6,8]))`								

Points	Function Call	0	1	2	3	None	False	Python error message	something else
(2 pts)	`print(indexOfSmallestOdd_b([9,10,20,40]))`								
(2 pts)	`print(indexOfSmallestOdd_b([2,10,11,7]))`								
(2 pts)	`print(indexOfSmallestOdd_b([2,7,5,10]))`								
(2 pts)	`print(indexOfSmallestOdd_b([2,4,6,8]))`								

Points	Function Call	0	1	2	3	None	False	Python error message	something else
(2 pts)	`print(indexOfSmallestOdd_c([9,10,20,40]))`								
(2 pts)	`print(indexOfSmallestOdd_c([2,10,11,7]))`								
(2 pts)	`print(indexOfSmallestOdd_c([2,7,5,10]))`								
(2 pts)	`print(indexOfSmallestOdd_c([2,4,6,8]))`								

End of Exam

Example `pytest` test cases from lab03

```
import pytest

def test_perimRect_1():
   assert perimRect(4,5)==18

def test_perimRect_2():
   assert perimRect(7,3)==20

def test_perimRect_3():
   assert perimRect(2.1,4.3)==12.8
```

Code for `indexOfSmallestOdd_a`

No hints on this one.

```
def indexOfSmallestOdd_a(alist):
    if type(alist)!=list:
        return False
    if alist==[]:
        return False

    soFar = 0
    for i in range(0,len(alist)):
        if type(alist[i])!=int:
            return False
        if alist[i] %2 == 1:
            soFar = i
    return soFar
```

1

Midterm 2 Handout for e02 CS8 F18

Handout for CMPSC 8, Midterm 2, F18, Page 2

Code for `indexOfSmallestOdd_b`

Hint: Pay attention to the indentation of line 13

```
1  def indexOfSmallestOdd_b(alist):
2      if type(alist)!=list:
3          return False
4      if alist==[]:
5          return False
6
7      soFar = False
8      for i in range(0,len(alist)):
9          if type(alist[i])!=int:
10             return False
11         if alist[i] %2 == 1:
12             soFar = i
13             return soFar
```

Code for `indexOfSmallestOdd_c`

```
1  def indexOfSmallestOdd_c(alist):
2      if type(alist)!=list:
3          return False
4      if alist==[]:
5          return False
6
7      soFar = False
8      for i in range(0,len(alist)):
9          if type(alist[i])!=int:
10             return False
11         if alist[i] %2 == 1:
12             if soFar == False:
13                 soFar = i
14             if alist[i] < alist[soFar]:
15                 soFar = i
16     return soFar
```

End of Handout