Student ID Number: ________________________

University of California, Berkeley – College of Engineering

Department of Electrical Engineering and Computer Sciences

Spring 2018 Instructor: Prof. Gerald Friedland 2018-02-22

CS88 Midterm Exam

Last *Name (Please print clearly)*		
First *Name (Please print clearly)*		
Student ID Number		
What time is your lab on Monday?		
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 8 pages including this cover page. Put all answers on pages 2-7; you can use page 8 for extra/doodle space. Please don't hand in any stray pieces of paper.
- Please turn off all pagers, cell phones and beepers. Remove all hats and headphones.
- You have 50 minutes to complete this exam. The midterm is closed book, no computers, no PDAs, no cell phones, no calculators, but you are allowed one double-sided sheets of notes and the midterm study guide. There may be partial credit for incomplete answers; write as much of the solution as you can. When we provide a blank, please fit your answer within the space provided.
- Remember: Whatever your score in this exam – you can clobber it with the finals. If you are caught cheating, however, it's an F and you will not be able to clobber.

Good luck!

Question	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Total**
Points	**2**	**2**	**2**	**2**	**2**	**14**	**12**	**4**	**40**

Warm-up Questions with short answers (2pts each)

Please write your answer within the designated boxes. We drop the lowest-scoring question in this section.

Question 1: TRUE or FALSE: “In general, recursion is more powerful than a for loop”. Explain why.

Question 2: TRUE or FALSE: “A function does not allow repeated execution of statements”. Explain why.

Question 3: Look at the box below. There is a list of problems on the left (labeled 1–4) and a list of the concepts that solve these problems (labeled A–D). **Write** the correct letter to the left of each number to match each problem to the Internet component that you could use to solve that problem.

______ 1. Step through a list

______ 2. Manipulate a Literal

______ 3. Shield a user from a concrete implementation

______ 4. Create a function that returns a function

A. Operator

B. Iterator

C. Higher Order Functions

D. Abstraction

Question 4: When analyzing an algorithm’s running time, we count the number of operations or “frames” instead of timing it with a stopwatch. List one benefit and one disadvantage of this decision.

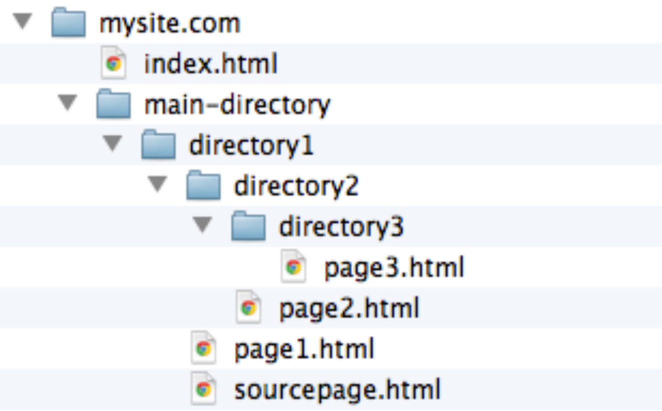
Question 5: Provide two practical reasons why Python allows to define a function within a function

Question 6: *File Search…* (14 pts)

In today’s computers, directories (also called folders) and files are used to structure information on hard disks and memory cards. Directories contain files and also other directories. Directories can also be empty. The uppermost directory is called /.

Write a function **`searchFile(name, dir)`** that searches for the file **`name`** in an arbitrary large and arbitrary shaped directory structure starting in directory **`dir`**, and reports **`true`** if the file exists or **`false`** if it doesn’t. Note that it is possible for a file and a directory to have the same name.

Example of a Simple Directory Structure:



```
▼ mysite.com
      index.html
  ▼ main-directory
    ▼ directory1
      ▼ directory2
        ▼ directory3
              page3.html
            page2.html
          page1.html
          sourcepage.html
```

For example in the above picture:

> **`searchFile(page1.html, mysite.com)`** should return **`true`.**
> **`searchFile(page4.html, mysite.com)`** should return **`false`.**

Here are two helper functions to help create your solution:

1. **`listContent(dir)`** – gives you all the files and directories in a directory as a list of names (as Strings) in the specified directory (no recursion)
2. **`isDir(name)`** – reports **`true`** if **`name`** is a directory and **`false`** if it is a file

(6a) Let’s start by creating a helper function **`getFiles(dir)`** that takes in a directory and reports a list of just the files in that directory. The solution should fit in one line.

```
def getFiles(dir):

    return ( ____________________________________________________________ )
```

(6b) Create another helper function **`getDir(dir)`** that takes in a directory and returns a list of sub-directories, and [] if none exist. You may not need all the lines.

```
def getDir(dir):

    ______________________________________________________

    ______________________________________________________

    ______________________________________________________

    ______________________________________________________

    ______________________________________________________

    ______________________________________________________
```

(6c) Now assume your helper functions **`getFiles(dir)`** and **`getDir(dir)`** work correctly. You may use them along with any other blocks to create the **`searchFile(name, dir)`** solution.

```
def searchFile(name, dir):

     _____________________________________________________

     _____________________________________________________

     _____________________________________________________

     _____________________________________________________

     _____________________________________________________

     _____________________________________________________
```

(6d) What is the estimated “runtime” counted in number of frames of **`searchFile(name, dir)`** as a function of the number of files and directories?

Student ID Number: ________________________

Question 7: *Checksum…* (12 pts)

A checksum is a function that calculates a particular value for a set of characters in order to make sure there are no errors. *Luhn's Algorithm* is a way of verifying whether a credit card number is valid. It does this by calculating a *checksum* of the card number that is only correct when all digits are also correct. Checksum algorithms are used every time you use the internet, to make sure the data being sent is transferred correctly.

Let’s define a very simple checksum algorithm: *Count every ‘a’ as 1, ‘b’ as 2, …, ‘z’ as 26, sum the values together. For example, the string “Cab” would be 3(C) + 1(a) + 2(b) = 6 points.*

We will give you two helper functions:

1. **`letterTovalue(letter)`**, which if given **`a`** or **`A`** returns 1, **`b`** or **`B`** returns 2, etc.
2. **`wordTolist(word)`**, which if given the word **`Cab`**, it would return the list **`[‘C’,‘a’,‘b’]`**

You can also use higher-order functions and arithmetic operations as usual.

(7a) Write **`checksum(word)`** that returns the checksum of the word. E.g., **`checksum(Cab)`**➔**`6`**

```
def checksum(word):

              ____________________________________________

              ____________________________________________

              ____________________________________________

              ____________________________________________
```

(7b) Give an example where this checksum algorithm would fail (i.e. where two different strings will give the same checksum value) and explain why.

Example 1 (two strings):

Why it fails:

(7c) Now assume that there are only 9 letters in the alphabet (a – i), each with the same “letter-value” as before (*‘a’ as 1, ‘b’ as 2, …, ‘i’ as 9*). Also assume that a given word cannot have any duplicate letters. Implement an improved algorithm in which every valid input reports a unique output. Explain verbally how your algorithm works, and how it fixes the problem from our original checksum.

In addition, you may use these two functions useful:

1. **pow(x,y)** – takes in numbers **`x`** and **`y`**, and reports **`x`** to the power of **`y`**
2. **pos(word, letter)** – takes in a letter and a word, and reports the position of the letter in the word

```
def uniqueChecksum(word):
            ____________________________________________________
            ____________________________________________________
            ____________________________________________________
            ____________________________________________________
            ____________________________________________________
            ____________________________________________________
            ____________________________________________________
```

How does your algorithm work, and how does it fix the problem from our original checksum?

Question 8: *Recursion to Iteration* (4 pts)

Rewrite the following function as an iterative function (Note you cannot use ** or the pow function):

```
def power(x, n)
    if n = = 0:
           return 1
    else
           return x * power(x, n-1)


def power_iter(x, n):

    ______________________________________________________________________

    ______________________________________________________________________

    ______________________________________________________________________

    ______________________________________________________________________

    ______________________________________________________________________

    ______________________________________________________________________
```

Student ID Number: _________________________

Doodle/Notes/Extra Space: