REMINDEERS

• Exam 2 will be held Monday, April 6, 2015. Most of you will take the test from 6:00-7:50PM in DCC 308.

• Students who have provided Prof. Goldschmidt with an accommodation letter requiring extra time will take the exam starting at 4:00PM in DCC 239.

• You MUST BRING YOUR ID to the exam. Missing IDs will result in a 20-point penalty.

Overview

• Primary coverage includes all lectures from March, Labs 4-7, and Homeworks 2-4. Material from strings and lists are part of this exam. Sets will not be covered on this exam. Images may be covered, but any needed functions will be provided to you.

   Important topics to keep in mind are as follows. Lists and tuples are very important for this exam. Make sure you are very comfortable with all the different list functions (append, insert, remove, concatenation, replication, and slicing) and the differences between tuples, string, and lists. They are indexed similarly, but assignment creates copies for every object type we have seen thus far except for lists and images. Study how to pass lists as arguments to functions (i.e. pass by reference). We will also ask elaborate questions on if statements, while and for loops, and range. Make sure you know how to do basic operations on lists using loops, such as compare items from two separate lists, find specific items in a list, find the index of a specific item in a list (min, max, last value with some property), add up values in a list, etc. Write functions that iterate over lists, such as to check if the list contains a specific item, returning True if so and False otherwise. These are examples of the type of problems we have solved and you should become very comfortable with. They are by no means the full list of possible questions!

• No calculators, books, electronics of any kind, etc.! You may bring a one-page, double-sided, 8.5” x 11” “crib sheet” sheet with you. You may prepare this as you wish. Feel free to work in groups to prepare a common crib sheet. Of course, each of you must have your own copy during the exam.

• We will assume you know (perhaps via your crib sheet) the following mathematical functions, some of which are in the math module: abs, ceil, float, int, max, min, sqrt, trunc

• We will assume you know the following functions associated with the data of type str: len, +, * (by an integer), capitalize, replace, str, upper, and lower.

• Python syntax is still important on this exam, though minor mishaps will be okay (sometimes!).

• Below are many sample questions, far more than will be on the exam. Solutions to most of the problems will be posted on the course website on Saturday, April 4. These posted solutions will not include problems involving output from Python — try those out yourself!

• Please note that your solution to any question that requires you to write Python code will be at most 10-12 lines long, and may be much shorter. Focus on writing short programs, as it will save you time during the exam.

• The exam questions will be closely related to the practice problems below and problems from the homeworks, labs, and lecture exercises.

• Remember that the exam is timed. You should be able to solve each programming problem in about 5-8 minutes. Time yourself when solving them, find out where you are spending too much time, and practice the related material.
• How to study?

– First, make sure you know the syntax of all the different programming constructs we have learned. You cannot construct solutions if you do not know the building blocks. Practice them like you would a foreign language; memorize them. The best approach is to type them into the Python shell and try new variations of code. This will also help when you are trying to track down syntax errors.

– Work through the sample questions, writing out solutions by hand (since you won’t have laptops at all for the exam!). Read solutions only as a last resort. Remember that reading a solution is much easier than actually writing one, and you are graded on the writing part.

– You are encouraged to work with other students as you study, but ask yourself if you understand the questions and material enough to solve problems on your own. Try to replicate a solution you worked on with someone a little while later without looking at any solutions.

– Review and re-do lecture exercises, lab, and homework problems.

– Identify the problems that cause you difficulty and review lecture notes and background reading on these topic areas. Go to office hours and ask to review a concept you did not understand.

– Use the Wing IDE / Python interpreter extensively to help you understand what is happening, but practice writing solutions out without using your laptop.

• Advance warnings:

– We will not answer questions from students in the middle of the exam unless there is a mistake in the question itself. In other words, do not ask what a certain line of code means or if your answer is correct.

Questions

1. Assume v is a list containing numbers. Write Python code to find and print the highest two values in v. If the list contains only one number, print only that number. If the list is empty, print nothing. For example, if we assigned

   v = [ 7, 3, 1, 5, 10, 6 ]

then the output of your code should be something like

7 10

If we are given that

v = [ 7 ]

then the output of your code should be

7

2. Consider a simplified version of the Yelp lab data, where just the name of the restaurant, the type of restaurant, and the ratings are provided. Assume these values have already been read into a list of lists of the form below:

   restaurants = [ [ 'Acme', 'Italian', 2, 4, 3, 5],
                 [ 'Flintstone', 'Steak', 5, 2, 4, 3, 3, 4],
                 [ 'Bella Troy', 'Italian', 1, 4, 5] ]

Write a segment of Python code that prints all Italian restaurants in the restaurants list that have no ratings of value 1 and at least one rating of value 5. In the above example, Acme would be printed in the output, but Flintstone and Bella Troy would not. Flintstone is not Italian and Bella Troy has a 1 rating. Your code should work for any legal version of restaurants.

3. Continuing with the Yelp lab, assume that you have the code
in_file = open('yelp.txt')

for line in in_file:
    p_line = parse_line(line)
    print p_line

and that you parse_line will return a list that looks like

["Meka's Lounge", 42.74, -73.69, "Bars", [5, 2, 4, 4, 3, 4, 5], 3.857142857142857 ]

Modify the for loop above to create a list called high that stores the names of all restaurants that have an average rating of at least 4.0. You do not have to print high.

4. In the game of chess you can often estimate how well you are doing by adding the values of the pieces you have captured. The pieces are Pawns, Bishops, Knights, Rooks and Queens. Their values are

<table>
<thead>
<tr>
<th>Piece</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P - (P)awn</td>
<td>1</td>
</tr>
<tr>
<td>B - (B)ishop</td>
<td>3</td>
</tr>
<tr>
<td>K - (K)night</td>
<td>3</td>
</tr>
<tr>
<td>R - (R)ook</td>
<td>5</td>
</tr>
<tr>
<td>Q - (Q)ueen</td>
<td>9</td>
</tr>
</tbody>
</table>

Write a Python function called chess_score that takes a single string as an argument and returns the combined values represented by the pieces in the string. You may assume that only 'P', 'B', 'K', 'R', and 'Q' appear in the string. You may not use any if statements and you may not use any loops. As an example,

print chess_score('BQBP')

should output the value 16 because there are 2 Bishops (3 points each), 1 Queen (9 points each), and 1 Pawn (1 point each).

5. Write a function that takes as a parameter a list, v. Using a while loop, it should count and return the number of values in v that are greater than the previous value in the list. For example, if v is

[ 21.5, 32.0, 16.5, 18.7, 33.2, -1.1, -0.5, 6.7, 19.4, 14.2 ]

the returned value should be 6 because of the 32.0, 18.7, 33.2, -0.5, 6.7 and the 19.4. After you are finished, rewrite the solution to using a for loop.
6. Write the output for each of the following sections of Python code:

Part a

for i in range(10,1,-2):
    print i

Part b

pickle = 2
for j in range(2):
    for k in range(2):
        pickle = pickle + 2
        pickle = pickle * 2
        print pickle

Part c

i = 0
while i<10:
    if i < 5:
        i += 5
    elif i%4 == 0:
        i -= 1
    else:
        i += 3
    print i

7. You are given a file that contains, on each line of input, three integers separated by commas. Write a
Python program that sums all of the first integers, the second integers, and the third integers, outputting
the resulting sums all on one line, separated by commas. As a simple example, if the input is

2, 5,7
3, 6, 10
1, 2, -3
2, 4, 1

Then the output should be

8, 17, 15

8. Write Python code to generate the following ranges

(a) (100,99,98,...,0)
(b) (55,53,51,...,-1)
(c) (3,5,7,9,...,29)
(d) (-95,-90,-85,...,85,90)

9. Write a while loop to add all of the numbers in a list v until it reaches a negative number or until it
reaches the end of the list. Store the sum in the variable result. Your code should work for any version
of v containing only numbers. For example, the value of result should be 25 after the loop for both of
the following lists:

v = [ 10, 12, 3, -5, 5, 6 ]

v = [ 0, 10, 3, 6, 5, 1 ]
10. Write Python code that takes a list of numbers, \( v \), and outputs the positive values that are in \( v \) in increasing order, one value per line. If there are no positive values, then the output should be the string 'None'. You may assume there is at least one value in the list. As an example,

\( v = [17, -5, 15, -3, 12, -5, 0, 12, 22, -1] \)

Then the output of your code should be

12  
12  
15  
17  
22

As a second example, if

\( v = [-17, -5, -15, -3, -12, -5, 0, -12, -22, -1] \)

then the output should be just

None

11. What is the output of the following operations:

```python
>>> mylist = [1,4,8,12,6]  
>>> x = mylist.sort()  
>>> print x
>>> mylist = [1,4,8,12,6]  
>>> slice1 = mylist[2:4]  
>>> slice1[0] = 20  
>>> print slice1  
>>> print mylist
```

12. Suppose you are given a list of integers, \( v \). In other words, \( v \) is already provided with a value in your program, such as

\( v = [126, -1, 347, -14, 14, 15, 29, 12, 9, 8] \)

Write Python code that:

- asks the user for one integer, reads it in and stores it in the variable \( x_0 \),
- asks the user for a second integer, reads it in and stores it in the variable \( x_1 \), and
- prints all values in \( v \) that are above \( x_0 \) and below \( x_1 \). For example, if the user gave the input -14 and 15 then the output should be

-1  
14  
12  
9  
8

Note: if the input was 15 and 5, then the output is empty! There is no number greater than 15 and less than 5.
13. Using the methods from the images lab, do the following:
   
   (a) Resize an image so that it is the same height but half the width.
   
   (b) Take an image, and create a new image that repeats this image m times horizontally and n times vertically.

14. Write a Python for loop to print out the values from the list v that are positive (0 is NOT a positive number).

15. What is the output of the following code?

   ```python
   L1 = [0,1,2,4,1,0]
   s1 = set(L1)
   L1.pop()
   L1.pop()
   L1.pop()
   L1[0] = 5
   s1.add(6)
   s1.discard(1)
   print L1
   for v in sorted(s1):
       print v
   ```

16. v is a list containing an odd number of values. Write Python code (no need for a whole function) to compute and print the average of the first, middle and last values in v, accurate to two decimal places. For example, if we assigned

   ```python
   v = [ 6, -5, 3.1, 7, 12, -5, 9 ]
   ```

   then the output of your code should be

   Average: 7.33

   Note that your code needs to work for any odd-length v.

17. Write a Python program that copies an input file called in.txt to an output file called out.txt. All lines from in.txt should be copied to out.txt except the first line and the last line. Try to think of some different ways to approach this.
18. What is the output of the following program?

```python
def spam(a1, b1, a2, b2):
    if (a1 == a2) and (b1 > b2):
        return 1
    else:
        return 0

def egg(a1, b1, a2, b2):
    if (a1 > a2) and (b1 == b2):
        return 0
    else:
        return 1

a1 = 3
b1 = 4
a2 = 6
b2 = 4

print spam(a2, b2, a1, b1)
print egg(a1, b1, a2, b2)
c = spam(a1, b2, a2, b1)
print c
c += egg(a1, b2, a2, b1)
print c
```

19. (16 pts) Write a function called `copy_half` that takes the name of two files as arguments. The function should copy the first, third, fifth, etc. lines (i.e. odd lines only) from the first file to the second file. For example, if the file names are `in.txt` and `out.txt` and if `in.txt` contains

```
starting line
not this line
middle line is here
    skip this line too
    I like this line
```

then after the call

```python
copy_half( 'in.txt', 'out.txt' )
```

the file `out.txt` should contain

```
starting line
middle line is here
    I like this line
```
20. Write a program that extracts all of the names of functions from the Python program stored in a file called `prog.py`. For example, if `prog.py` contains

```python
def a1 (y):
    def b__(x):
        return x*x
    return b__(y) * 3
def _a134():
    print "This is _a134"
print a1(3)
```

then your program should output

```
a1
b__
_a134
```

21. Write a segment of code that outputs the indices and the values in a list, v. For example, the output for the list

```python
v = [ 15, 'bicycle', 'xray', -3.14 ]
```

should be

```
0: 15
1: bicycle
2: xray
3: -3.14
```

22. Write a function called `count_in_range` that counts and returns the number of values in a list that are between the values stored in parameters `x0` and `x1`. (Values equal to `x0` or `x1` should not be included in the count.) The function begins with

```python
def count_in_range( v, x0, x1):
```
23. Write a segment of code that reads integers from a file called **test2.txt** and stores the positive values in one list, the negative values in a second list, and skips blank lines and zeros. The order of the values in each list should match the order of the input. Each line of input will contain either spaces or spaces and an integer. For example, if **test2.txt** contains

```
11
-3
5
0
```

Then after your code, the list **P** should be [11, 5] and the list **N** should be [-3].

24. Give the output of each of the following

(a)

```python
i = 4
L = [0, 12, 3, 5, 2, -1]
while 0 <= i and i < len(L):
    if L[i] < 0:
        break
    else:
        i = L[i]
print i, L[i]
```

(b)

```python
tough = 2
for i in range(2):
    s = 1
    for j in range(i, tough):
        s += tough
    print s
    print tough
    tough = s
print tough
```

25. Suppose time is represented by a tuple containing the hour and minute, both as integers. (We are ignoring AM and PM in this problem.) For example,

```
t1 = (12, 3)
```

is 12:03. Write a function that takes two time tuples and returns `True` if and only if the first tuple represents an earlier time in a day than the second. As examples,

```python
>>> earlier_than((12, 3), (1, 5))
True
>>> earlier_than((5, 45), (1, 5))
False
>>> earlier_than((5, 45), (5, 45))
False
```
26. Suppose a list of words in alphabetical order has been assigned to the variable called `words`. For example, we might have the assignment

```python
words = [ 'aardvark', 'abaka', 'expedite', 'experience', 'shoetrees', 'tastetest', 'test' ]
```

Write code to find and output the first and the last string in `words` that start and end with the same letter and are at least 8 characters long. You may assume that at least one word in `words` satisfies this condition. You may write a function if you wish. For the above example, the output should be

```
expedite
tastetest
```

27. Recall that our Sudoku board from lab was represented as a list of lists. For example,

```python
bd = [ [ '1', '.', '.', '.', '2', '.', '.', '3', '7' ],
      [ '.', '6', '.', '.', '.', '5', '1', '4', '8' ],
      [ '.', '5', '.', '.', '.', '.', '2', '9' ],
      [ '.', '.', '.', '9', '.', '.', '4', '.', '.' ],
      [ '.', '.', '4', '1', '.', '3', '7', '.', '.' ],
      [ '.', '.', '1', '.', '.', '4', '.', '.', '.' ],
      [ '4', '3', '.', '.', '.', '.', '.', '1', '.' ],
      [ '.', '1', '7', '5', '.', '.', '.', '8', '.' ],
      [ '2', '8', '.', '.', '4', '.', '.', '.', '6' ] ]
```

Your goal in this problem is to write a function called `min_and_max_filled` that outputs the minimum and maximum number of values that are filled in the rows. For the above example, the output values for `bd` would be 2 and 5 because rows 3 and 5 both have just two values ("9" and "4" in row 3 and "1" and "4" in 5), while row 1 has 5 values ("6", "5", "1", "4" and "8"). Solve this in 2 parts:

(a) Write a short function called `num_filled` that takes a single list from a row, such as

```python
[ '.', '6', '.', '.', '.', '2', '4', '8' ]
```

and returns the number of values filled in (in this case 5). This can be done using just two lines of code, without a `for` loop. If you need to use a `for` loop, slightly less credit will be given.

(b) Now write function `min_and_max_filled`, which takes a Sudoku board as its only argument. It must call function `num_filled` for each row, and make use of the values returned by these function calls to solve the problem. The output from `min_and_max_filled` for the example board above should simply be

```
2 5
```
28. Please show the output from the following code?

```python
def get_min(v):
    v.sort()
    return v[0]

def get_max(v):
    x = max(v)
    return x

v = [14, 19, 4, 5, 12, 8]
if len(v) > 10 and get_min(v) > 6:
    print "Hello"
    print v[0]
    print v[4]
else:
    print "So long"
    print v[0]
    print v[-1]
    if len(v) < 10 or get_max(v):
        print get_max(v)
        print v[0]
        print get_min(v)
        print v[0]
```

29. Write code that uses a `range` (and NO loops) to generate the following lists:

```python
v0 = [10, 9, 8, 7, 6, 5, 4, 3]
v1 = [-10, -3, 4, 11, 18, 25, 32, 39]
```

30. Consider the following list of lists of strings:

```python
wordy = ['impala', 'malibu', 'camry', 'jetta'],
        ['zebra', 'impala', 'lion', 'impala', 'malibu', 'zebra'],
        ['tiger', 'lion', 'cowboy', 'jet', '49er'],
        [],
        ['five', 'seven', 'nine']
```

(a) Show the output:

```python
print wordy[2][1]

print wordy[1][2][3]

print len(wordy)

print len(wordy[1])

print sorted( set(wordy[0]) & set(wordy[1]) )
```
(b) Write a loop to print the last word of each list in \texttt{wordy}, stopping when either an empty list is found or when there are no more lists. For the above example, the output should be

\begin{verbatim}
jetta
zebra
49er
\end{verbatim}

31. Suppose census data has been read into a list called \texttt{census}. Here is a short version of this list.

\begin{verbatim}
census = [['Albany', 'county', 294571, 304204 ],
         ['Albany', 'city', 94444, 97856],
         ['Berne', 'town', 2853, 2794],
         ['Coeymans', 'town', 8161, 7418],
         ['Cohoes', 'city', 15607, 16168],
         ['Spamalot', 'county', 81100, 99999 ],
         ['Colonie', 'town', 79327, 81591],
         ['Green Island', 'town', 2283, 2620] ]
\end{verbatim}

Recall that the first number in each list in \texttt{census} is the population in 2000 and the second number is the population in 2010. Write code to determine the average town size in 2000 and the average town size in 2010. Integer arithmetic is fine. For the version of \texttt{census}, the average town sizes are \((2853 + 8161 + 79327 + 2283)/4 = 23156\) in 2000 and \((2794 + 7418 + 81591 + 2620)/4 = 23605\), so the output should be

\begin{verbatim}
2000: 23156
2010: 23605
\end{verbatim}

32. Show the output from the following code:

\begin{verbatim}
def elephant(height):
    time_step = 1
    steps = 0
    while steps < height:
        steps += time_step
        steps -= time_step/3 # note: this is integer division
        time_step += 1
    print "%d, %d" %(time_step, steps)

elephant(0)
elephant(5)
elephant(6)
\end{verbatim}
33. Show the output of the following code. Make sure we can determine what is output and what is scratch work.

```python
def remove_something(z):
    z.remove( z[z[0]] )

v = [ 1, 8, [12, 8], 'hello', 'car' ]
x = 'salad'

if len(v[2]) >= 2:
    if x > v[3]:
        if v[0] == 1:
            print 'Three'
        else:
            print 'Two'
    else:
        print 'One'
elif len(v) == 5:
    print 'Six'
else:
    v.append('five')
    print 'Ten'

remove_something(v)
print v[1]
print v[2]
v.append(x)
print len(v)
```

34. Write a function that takes as input a string and returns a new string that has a plus sign in the input string after every two characters. Example runs are given below.

```python
>>> addplus('abracadabra')
'ab+ra+ca+da+br+a'

>>> addplus('steven')
'st+ev+en+'
```

35. Write a function

```python
compare_semester(sem1,sem2)
```

that takes as input two tuples `sem1,sem2` representing two academic semesters, each given in the form of a semester (either `Fall` or `Spring`) and year pair.

Your function should return 1 if `sem2` comes after `sem1`, -1 if `sem1` comes after `sem2`, and 0 otherwise. Remember, Fall 2014 comes after Spring 2014.

```python
>>> compare_semester(('Fall',2013), ('Spring',2014))
1
>>> compare_semester(('Fall',2013), ('Spring',2012))
-1
>>> compare_semester(('Fall',2012), ('Spring',2012))
-1
>>> compare_semester(('Fall',2012), ('Fall',2012))
0
```
36. You are given in variable $x$ a list of lists represented as an $N \times N$ grid in which each list corresponds to one row of the grid. For example, a $4 \times 4$ grid is given by:

$x = [[1, 2, 3, 4], [4, 3, 2, 1], [2, 1, 4, 2], [2, 1, 4, 5]]$

Write a piece of code to print the grid in the following format with a vertical and horizontal line right in the middle:

```
1 2 | 3 4
4 3 | 2 1
----|----
2 1 | 4 2
2 1 | 4 5
```

37. Write a piece of code that repeatedly asks the user for numbers using `raw_input` until the user enters 'stop'. Then, the program reports the sum of the values entered by the user and the total number of values strictly greater than zero. You can assume that the user enters a valid number until she enters stop.

An example run of this code is given below.

```
Enter a value ==> 1.2
Enter a value ==> 0
Enter a value ==> 2
Enter a value ==> -1
Enter a value ==> stop
Sum: 2.2
Values > 0: 2
```

38. You are given a list of restaurants in the RPI region in a variable called `rest` in the form:

```
rest = [ "De Fazio's", "Troy", "Pizza", 4.5],
       ["Dinosaur BBQ", "Troy", "BBQ", 3.5],
       ["Van's", "Albany", "Vietnamese", 4],
```

For each restaurant, the first item is the name of a restaurant, the second item is the location, the third item is the type of restaurant it is, and the fourth item is the score.

Write a piece of code that uses this variable to find all restaurants in Troy with a score of at least 4 and print the name and the category. For example, for the above list, it would print:

```
De Fazio's (Pizza)
Lucas Confectionery (Wine Bar)
```

39. Write a function `remove_val(l, val)` that removes all copies of `val` from list `l`.

Suppose you are given a variable $x$ containing numbers as shown below:

```
x = [1, 4, 2, 1, 2, 4, 4, 2, 5, 5, 2]
```

Then, your function should work as follows:

```
>>> remove_val(x, 4)
>>> x
[1, 2, 1, 2, 5, 5, 2]
```

Note: if your function returns a new list with this content instead of modifying it as given, you will lose 3 points.
40. Suppose you are given the scores of two athletes in various competitions, given in two separate lists. Assume there are unknown number of competitions numbered 1, 2, 3, etc. and the length of the two lists is the same.

\[ a_1 = [11, 8, 11, 9] \]
\[ a_2 = [11, 9, 8, 12] \]

For example according to this list, both athletes got a score of 11 in competition 1. Print the index of all the competitions in which \( a_2 \) did better. For example, for the above lists, we would print:

\( a_2 \) is better in 2 4

If there is no value in which \( a_2 \) is better, then you should print:

\( a_2 \) is never better

41. What is the output of the following programs:

<table>
<thead>
<tr>
<th><strong>Part a</strong></th>
<th><strong>Scratch area:</strong></th>
</tr>
</thead>
</table>
| a = 25  
b = 11  
while True:  
    print a, b  
    if a <= 0 or b <= 0:  
        break  
    if a > b:  
        a = a - b  
    else:  
        b = b - a  
    b -= 1  
a += 1  
Output: | Scratch area: |
Part b

def spam(l,s):
    m = len(s)/2
    s1 = s[:m]
    s2 = s[m:]
    if l.count(s1) == 0:
        l.append(s1)
    if l.count(s2) == 0:
        l.append(s2)

l = ['ab','cd','de','fg']
s1 = 'abcde'
s2 = 'fghi'
spam(l,s1)
print s1
print l
l = spam(l,s2)
print s2
print l

Output: