CS 115 Exam 2, Sections 5–8 Thu. 4/2/2015

Name: ____

Rules and Hints

- You may use one handwritten 8.5×11" cheat sheet (front and back). This is the only additional resource you may consult during this exam. *No calculators.*
- Explain/show work if you want to receive partial credit for wrong answers.
- As long as your code is correct, you will get full credit. No points for style.
- When you write code, be sure that the indentation level of each statement is clear.

Grade

Problem	**Your Score**	**Max Score**
Problem 1: Trace snippets of code		15
Problem 2: Trace functions		15
Problem 3: Define functions		30
Problem 4: Write a complete program		40
Total		100

Problem 1: Trace snippets of code (15 points)

What will print to the screen when each of the following snippets of code is executed in IDLE or in the Online Python Tutor?

Be very clear with spacing, line breaks, etc.

Treat each sub-problem as an independent question.

Problem 1A

```
x = 2
y = 15
while x < y:
    x *= 3
print(x)
```

Problem 1B

```
s = 'CS115'
print(len(s))
print(s[1:3])
```

Problem 1C

```
strings = [ 'Los Angeles', 'San Diego', 'San Jose',
            'San Francisco', 'Sacramento' ]
print(len(strings))
print(strings[3])
print(strings[3][2])
```

Problem 2: Trace functions (15 points)

What will print to the screen when each of the following snippets of code is executed in IDLE?

Be very clear with spacing, line breaks, etc.

Note: the parts of this problem are independent.

For all parts of this problem, assume that the following functions have been defined.

```
def f1():
    return 10

def f2(x, y):
    return x + y - 2

def f3(y, x):
    return 3 * x + y

def f4(z):
    return f2(f3(1, z), f1())
```

Problem 2A

```
print(f1())
```

Problem 2B

```
print(f2(7, 8))
```

Problem 2C

```
print(f4(10))
```

Problem 3: Define functions (30 points)

Define functions to perform the following tasks, obeying these requirements:

- Assume that the math library has been imported for you.
- Do NOT ask the user for input unless the specification explicitly requires it.
- Do NOT print anything unless the specification explicitly requires it.
- Do NOT call `sys.exit()` to terminate the program.

Problem 3A

Define a function named `semi` that:

- Takes 3 parameters: the three side lengths of a triangle
- Returns the *semi*-perimeter (one-half of the total perimeter)

Problem 3B

Define a function named `tri_area` that:

- Takes 3 parameters: the three side lengths of a triangle
- Uses Heron's formula to compute the area, and returns the result. Heron's formula is as follows:

  $$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

  In this formula, s is the semi-perimeter of the triangle, and a, b, and c are the side lengths.
- For full credit, this function should call the function you wrote in Part A to compute the semi-perimeter, rather than doing that computation directly.

Problem 3C

Define a function named `list_is_square` that:

- Takes 1 parameter: a nested (2D) list
- Returns `True` if the list represents an NxN grid for some value of N, and `False` otherwise.

For example, the following list is an NxN grid, where N = 3:

```
L1 = [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ]  # True: 3x3
```

The following list is not:

```
L2 = [ [1, 2, 3], [4, 5, 6], [7, 8, 9, 10] ]  # False: irregular shape
```

Nor is the following list:

```
L3 = [ [1, 2], [3, 4], [5, 6] ] # False: 3x2
```

It doesn't matter what the actual values in the lists are; we are just looking for the number of elements in each.

Problem 4: Write a complete program (40 points)

For this problem, you must write a complete program. That includes a docstring, a `def main()`, any necessary library imports, etc.

Each function you write should obey the rules given in Problem 3. **To get full credit, your functions should call each other where appropriate to avoid duplicating work!**

Your program should contain the following functions:

- `read_names`: Takes no parameters. Prompts the user for names until the user enters a blank line. Returns a list of the names the user entered (not including the blank line). Each line of user input should count as a single name, even if it has multiple words.
  For example:
  ```
  Enter a name:  Ilana Wexler
  Enter a name:  Abbi Abrams
  Enter a name:
  ```
  should return the list `[ 'Ilana Wexler', 'Abbi Abrams']`
- `is_sorted`: Takes a list of strings as a parameter. Should return `True` if the list is in alphabetical order (A-Z) and `False` otherwise. Remember that, if strings `s1` and `s2` are the same case (upper or lower), `s1 < s2` will be `True` if `s1` comes first alphabetically. Don't try to pull out the last names – just sort the strings as the user typed them.
- `average length`: Takes a list of strings as a parameter, and returns the average length (in characters) of the strings in the list. Spaces count as characters.
- `main`: Calling your other functions whenever possible, this function should get a list of names from the user (stopping when the user enters a blank line). It should then print the following:
  - The string that comes last alphabetically (only if the list is sorted)
  - All of the strings whose lengths are greater than the average length

Problem 4 code

Problem 4, continued