Spring, 2019 Name: ______________________

(Please *don't* write your id number!)

# Exam 1: Python Programming

## Directions

For this exam you are permitted one sheet (8.5 x 11 inches) of paper of notes on one side. It is a good idea to condense your notes into a small amount of ready reference material.

If you need more space, use the back of a page. Note when you do that on the front.

Before you begin, please take a moment to look over the entire test so that you can budget your time.

Clarity is important; if your answers are sloppy and hard to read, you may lose some points.

### For Grading

| Question: | 1 | 2 | 3 | Total |
|---|---|---|---|---|
| Points: | 30 | 35 | 35 | 100 |
| Score: | | | | |

1. (30 points) [Programming] Define a Python function, `box_volume(length,height,depth)`, which takes three numbers, `length`, `height`, and `depth` and returns the volume of a box with those dimensions. (Assume that the units are all the same, say inches.) Recall that the formula for the volume of a box with length $\ell$, height $h$, and depth $d$ is $\ell \cdot h \cdot d$.

   Tests for this problem appear in Figure 1.

```
# $Id: test_box_volume.py,v 1.1 2019/01/24 02:59:08 leavens Exp leavens $
from box_volume import box_volume
from math import isclose
def test_box_volume():
    assert isclose(box_volume(1,1,1), 1)
    assert isclose(box_volume(5,10,20), 1000)
    assert isclose(box_volume(2.0,4.0,6.0), 48.0)
    assert isclose(box_volume(55.12,39.44,92.1254), 200274.42877311996)
```

Figure 1: Tests for `box_volume`.

2. (35 points) [Programming] Define a Python function, `strictly_increasing(n1,n2,n3,n4)` that takes four numbers, `n1`, `n2`, `n3`, and `n4`, and returns true just when these arguments are in strictly increasing order, and false otherwise. That is, it returns true just when `n1` < `n2` < `n3` < `n4`.

   Tests for this problem appear in Figure 2.

```
# $Id: test_strictly_increasing.py,v 1.1 2019/01/24 16:04:20 leavens Exp leavens $
from strictly_increasing import strictly_increasing
def test_strictly_increasing():
    assert strictly_increasing(43, 44, 45, 46)
    assert strictly_increasing(65, 500, 4000, 30000)
    assert strictly_increasing(-2.1, -1.0, 0, 1.345687912)
    assert strictly_increasing(-20, -10, -1, 0.532)
    # however, note:
    assert not strictly_increasing(0, 0, 1, 2)
    assert not strictly_increasing(91, 92, 92, 93)
    assert not strictly_increasing(99, 3, 55, 1)
    assert not strictly_increasing(999, 888.9, 777.885, 777.888)
    assert not strictly_increasing(23, 42, 59, 59)
    assert not strictly_increasing(39, 39, 39, 40)
    assert not strictly_increasing(1, 0.5, 0.25, 0.125)
```

Figure 2: Tests for `strictly_increasing`.

3. (35 points) [Programming] Define an interactive Python procedure, `formletter()`, that prompts to stdout and reads from stdin. When called, `formletter()` prompts with the string “`first name? `” (without the quotes) on stdout, and reads a first name from stdin. It then prompts with the string “`last name? `” (also without the quotes) on stdout, and reads a last name from stdin. Then it prompts with “`amount owed? `” on stdout and reads an amount (a string) on stdin. Note that each prompt has exactly one space following the question mark. It then prints a form letter asking for payment of the amount as shown below. The following is a sample interaction with this program, in which the non-blank text following the question mark is input by the user (on stdin) and the texts up to the question mark (and the following blank) on the first 3 lines are prompts, and after the first 3 lines is the output (all on stdout) of the letter.

```
first name? Frank
last name? Drebbin
amount owed? 2,000,000.00 euros
Frank Drebbin

Dear Frank,

You owe us 2,000,000.00 euros. Please pay by tomorrow!

Sincerely, The Collection Company
```

Another interaction would be the following.

```
first name? Melinda
last name? Gates
amount owed? $6,000,000.00
Melinda Gates

Dear Melinda,

You owe us $6,000,000.00. Please pay by tomorrow!

Sincerely, The Collection Company
```

Your program should assume that the user gives a string in response to each prompt.