**Com S 127x**
**Lab 5 - Exam review problems**

The purpose of this lab is to provide some practice in preparing for the exam. You will need some blank paper and a pencil. The only checkpoint is to review what you've done with the TA by the end of the period. You will not be penalized if you don't finish all the problems during the lab period.

*Remember, your TA is there to help and answer questions! After the lab period, you can post questions on Piazza or post your own solutions for other people to review and check.*

## How do I study for an exam like this?

You can become a good at problem-solving and coding with practice and experience.

Given a problem, think of a concrete example to be a "test case". That is, if you wrote the code and wanted to know whether it was correct, how would you check? Often this involves some hand calculation, and often the hand calculation shows you how to write the code. Try to describe to yourself in words the steps in solving the problem.

After writing the code on paper, try to read it. Does it make sense to you? Trace through it by hand and make sure it is really doing what you intended at every step.

Type up your solution in Wing or activecode, and try it using your concrete examples as test cases.

If you weren't successful, get some help from the TA or Piazza or a fellow student. But ask yourself: In the future, how can I remember the way to solve this kind of problem? How can I recognize a similar problem? Can I make up more problems that are like this one? What information can I stash in my brain to make it easier the next time?

*Test-taking also involves some strategy*. Ask yourself: What kind of problem am I being asked to solve? Am I reading existing code, answering a question, writing a few statements, writing a function, or what? Am I printing output? Am I reading input? If I am writing a function, what are its parameters? Does it return a value, or does it perform some other action (like drawing or printing)?

1) Mousetraps are $3.00 each or $50.00 for a box of 20. Shipping is $4.00 per box or .25 for an individual mousetrap. In addition, sales tax is added to the total order cost (including shipping). Write a Python function **`mousetrap_cost`** that returns the total cost of an order of mousetraps, given the number of mousetraps *and* the local sales tax rate. (Assume the tax rate is given as a decimal, e.g. "6.5%" would be given as .065.)

- What kind of problem is this? A complete program? Just a function? Does it read input, or print output, or both, or neither? If it's a function, what are its arguments? Does it return a value? If so, what type does it return?
- Calculate by hand the result for 50 mousetraps using a sales tax rate of .10.
- Write the code on paper.
- Code it in Wing 101. Test it on a few values.

2) Assume that the function **`mousetrap_cost`** from the previous problem is already implemented in a module called **`mice.py`**. Write a Python program that *uses* that function. Your program should input the number of mousetraps from the user and print the order cost, using a fixed sales tax rate of 0.05. (*Don't rewrite or modify the existing function* **`mousetrap_cost`**.)

3) Eggs are 3.00 per dozen or 1.75 per half dozen.

a) How much would you have to spend for 30 eggs? How about for 31 eggs? Do this part by hand and show how you got your answer.

b) Write a function **`egg_cost`** that takes a number of eggs and returns amount you'd have to spend for at least that many eggs.

4) Suppose we have the function **`foo`** below:

```
def foo(x, y):
  result = False
  if x > y:
    if y != 0:
      result = True;
    if x == 0:
      result = True
  return result
```

What is the value *returned* by the function in each of the following cases?

```
foo(2, 1)

foo(0, -1)

foo(1, 1)
```

5) What's the output of the code below that is supposed to calculate the hypotenuse of a right triangle? Where is the error?

```
import math

def hypotenuse(a, b):
  c = a * a + b * b
  math.sqrt(c)
  return c

result = hypotenuse(3, 4)
print(result)
```

6) Suppose we have the function definition

```
def g(x):
  return 2 * x + 1
```

Trace the execution of the following sequence of statements. *After each line, show the values of the variables foo and bar after the line executes*.

```
foo = 5
```

| foo | |
|---|---|
| bar | |

```
bar = g(foo)
```

| foo | |
|---|---|
| bar | |

```
foo = g(bar - 1)
```

| foo | |
|---|---|
| bar | |

```
bar = g(g(foo))
```

| foo | |
|---|---|
| bar | |

6) Write a function **`next_multiple`** that takes one number and returns the smallest number that is greater than or equal to it, *and* is an exact multiple of 100. For example, **`next_multiple(302)`** is 400, and **`next_multiple(300)`** is 300.

7) Rewrite the following code so that it only uses one **`if`**-statement.

```
if score >= 0:
  if score <= 100:
    print("ok")
else:
  print("error")
```

8) Rewrite the function in problem 4 so that it has at most one `if`-statement. Can you write it with no `if`-statements at all?

9) Try this function on some values and figure out what it does, and describe its purpose in words.

```
def mystery(x, y, z):
  w = x;
  if y > w:
    w = y
  if z > w:
    w = z
  return w
```

10) In class we wrote the function grade_calculator below:

```
def grade_calculator(score):
    if score >= 90:
        grade = "A"
    elif score >= 80:
        grade = "B"
    elif score >= 70:
        grade = "C"
    else:
        grade = "F"
    return grade
```

Modify this function so that for grades other than F, it adds a "+" or "-" to the letter when the score is within the top three or bottom three scores for its range. For example: scores of 87, 88, 89 would be "B+" and 80, 81, 82 would be "B-". (Hint: you can do this without adding a bunch of additional cases. Look at the remainder when the score is divided by 10.)