Your name:________ ***SOLUTION*** _________________

1. You are about to write a method that does some complex math. As usual, ***you start by doing a concrete example by hand.*** What are some things you should do? Check all that apply:

    _____ Choose your numbers to be big and complicated.

    YES Choose numbers that avoid symmetry.

    YES Give names to the relevant items.

    YES Track how you calculate the answer by hand.

    YES Use this example as a unit test that you write first before writing your code.

In the following problems, consider the two code snippets below. They are intended to indicate whether or not the given grade is a Passing or Failing grade, where 70 is the lowest Passing grade

Snippet A:	***Snippet B***:

`if grade >= 70:`
`    print "pass"`
`if grade < 70:`
`    print "fail"`
	`if grade >= 70:`
	`    print "pass"`
	`else:`
	`    print "fail"`

2. Given the same value for ***grade***, the snippets produce the same output. ***True*** **False** (circle your choice)

3. Given the same value for ***grade***, which snippet ***runs faster***? (Circle your choice.)

    The one on the LEFT ***The one on the RIGHT*** ***Neither (they are equally fast)***

4. Which snippet is ***better***? (Circle your choice.)

    The one on the LEFT ***The one on the RIGHT*** ***Neither (they are equally good)***

5. One of the most important reasons to use the appropriate form in problems like the above is (check all that apply):

    YES If you use the inappropriate form at a job interview, you will embarrass yourself and not get the job.

    YES If you use the inappropriate form at a job interview, you will embarrass yourself and not get the job.

    YES If you use the inappropriate form at a job interview, you will embarrass yourself and not get the job.

6. In the space to the right, write code for a function that has a single parameter whose value must be one of the following letter grades:

   "A" "B" "C"

(everyone gets a passing grade in this function!)

The function returns the value of the letter grade (4 for an A, 3 for a B, 2 for a C).

```
def grade(letter):
    if grade == "A":
        return 4
    elif grade == "B":
        return 3
    else:
        return 2
```

[Doing the IF’s in a different order, for example “C” then “A”, is also correct.]

7. In the box below, write an implementation of a function ***is_odd*** that takes a non-negative integer and returns ***True*** if the integer is odd, and ***False*** otherwise.

   Hint: Think about the expression **X % 2** and what it evaluates to when **X** is odd, and when **X** is even. For example, to what do the following evaluate?

   17 % 2 **18 % 2** **371934 % 2**? **12345 % 2**

```
def is_odd(n):
    if n % 2 == 1:
        return True
    else:
        return False
```

A better solution is:

```
    return n % 2 == 1
```

Make sure that you see why this solution is equivalent to the above solution. (Think about the two cases, and what gets returned in each case.)

8. Suppose that you are given a function ***sum_of_digits*** that takes a non-negative integer and returns the sum of the digits in that integer. For example:

   sum_of_digits(81323) returns 8 + 1 + 3 + 2 + 3, which is 17.

   Suppose that you are also given a function ***is_odd*** that takes a non-negative integer and returns ***True*** if the integer is odd, and ***False*** otherwise. For example:

   is_odd(17) returns ***True*** and **is_odd(147204)** returns ***False***

   In the box below, write an implementation of a function ***count_odd_digit_sums*** that takes a non-negative integer ***m*** and returns the number of integers ***X*** from ***m*** to (***2 times m) + 1***, inclusive, for which the ***sum-of-digits of (X squared) is odd***. For example:

   count_odd_digit_sums(5) returns ***5*** because:

   5 squared is ***25***, whose sum of digits is ***7***, which is odd.

   6 squared is ***36***, whose sum of digits is ***9***, which is odd.

   7 squared is ***49***, whose sum of digits is ***13***, which is odd.

   8 squared is ***64***, whose sum of digits is ***10***, which is NOT odd.

   9 squared is ***81***, whose sum of digits is ***9***, which is odd.

   10 squared is ***100***, whose sum of digits is ***1***, which is odd.

   11 squared is ***121***, whose sum of digits is ***4*** which is NOT odd.

   Your solution MUST use (i.e. call) the functions ***sum_of_digits*** and ***is_odd*** appropriately. As in ALL problems through Exam 1, you may NOT use the multiple-argument form of RANGE.

```
def count_odd_digit_sums(m):
    count = 0
    for k in range(((2 * m) + 1) - m) + 1):
        number = k + m
        number_squared = number ** 2
        sum_digits = sum_of_digits(number_squared):
        if is_odd(sum_digits) == True:
            count = count + 1
    return count
```

The above solution is more verbose than an experienced software developer would write. However, for more complicated problems it is often helpful to compute intermediate values (per your solved-examples-by-hand), give them names, and build up a solution from those named values. The next page shows a "cleaner" solution.

Here is a "cleaner" solution:

```
def count_odd_digit_sums(m):
    count = 0
    for k in range(m + 2):
        if is_odd(sum_of_digits((k * m) ** 2):
            count = count + 1
    return count
```