fullName:____________________________andrewID:____________________________ recitationLetter:______

15-112 S23

Quiz1 version A (25 min)

You **MUST** stop writing and hand in this **entire** quiz when instructed in lecture.

- You may not unstaple any pages.
- Failure to hand in an intact quiz will be considered cheating. Discussing the quiz with anyone in any way, even briefly, is cheating. (You may discuss it only once the quiz has been posted to the course website.)
- You may not use your own scrap paper. If you must use additional scrap paper, raise your hand and we will provide some. You must hand any scrap paper in with your paper quiz, and we will not grade it.
- You may not use any concepts (including builtin functions) which we have not covered in the notes in week 1 / unit 1.
- You may not use strings, loops, lists, indexing, tuples, dictionaries, sets, or recursion.
- We may test your code using additional test cases. Do not hardcode.
- Assume almostEqual(x, y) and rounded(n) are both supplied for you. You must write all other helper functions you wish to use, unless we specify otherwise.

True or False [3pts ea]

Mark each of the following statements as True or False.

1. ○ True ○ False

I understand that it is an Academic Integrity Violation to discuss anything at all about this quiz with anyone other than current 112 TAs and faculty, regardless of which lecture they attend, until the quiz is posted on the course website. Hint: We hope your answer is True.

2. ○ True ○ False

If Python tries to evaluate the following line, it will crash due to a syntax error:

```
print(1/0)
```

3. ○ True ○ False

Because of the 'Approximate Value of Floating-Point Numbers', you should generally use almostEqual(x,y) instead of (x == y) if either x or y are floating-point values.

4. ○ True ○ False

The following line prints True:

```
print(isinstance(3, int) == isinstance("3", int))
```

CT1: Code Tracing [9pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def ct1(m):
    a = m % 10
    b = m // 100
    c = a > b
    if type(c) == str:
            print("c =", c)
    elif type(c) == bool:
            if b < 10:
                b += 1
            print("a * b =", a * b)
    if c == False:
            print("a =", a)
    else:
            print("b =", b)
    return b + a

print(ct1(368))
```

CT2: Code Tracing [9pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def f(x, y):
  y += 2
  print(y)
  return x - x // y

def ct2(x, y):
  x = f(x, y - 1)
  print(x / (y - 1))

print(ct2(7, 5))
```

Free Response 1: isDuplicatedNumber(n) [35pts]

We say that a number is duplicated if the left and right halves are the same. For example, 55 and 123123 are both duplicated numbers, but 1221 and 12312 are not.

With this in mind, write the function isDuplicatedNumber(n) which takes a non-negative int n and returns True if n is a duplicated number and False otherwise.

Notes:

- The numbers can be arbitrarily large. For example, 123456789123456789 is a duplicated number.
- We will say the 0 is not a duplicated number.
- Remember, do not use strings, loops, lists, or anything else prohibited on page 1 of this quiz.

Here are some test cases:

```
assert(isDuplicatedNumber(11) == True)
assert(isDuplicatedNumber(55) == True)
assert(isDuplicatedNumber(123123) == True)
assert(isDuplicatedNumber(123) == False)
assert(isDuplicatedNumber(1221) == False)
assert(isDuplicatedNumber(12312) == False)
assert(isDuplicatedNumber(121212) == False)
assert(isDuplicatedNumber(1) == False)
assert(isDuplicatedNumber(0) == False)
```

We have provided the digitCount(n) helper function below, which you may call in your solution (without rewriting it) if you wish.

```
def digitCount(n):
    n = abs(n)
    if n == 0:
        return 1
    return 1 + math.floor(math.log10(n))
```

Begin your answer on the next page

Begin your FR1 answer here

Free Response 2: commonFactor(a, b, c) [35pts]

Write the function commonFactor(a, b, c) which takes three non-negative integers and returns True if the **smallest** value of a, b, and c is a factor of the other two. Consider these test cases carefully:

```
assert(commonFactor(3, 6, 12) == True)    # 3 is a factor of 6 and 12
assert(commonFactor(10, 2, 112) == True)  # 2 is a factor of 10 and 112
assert(commonFactor(14, 7, 7) == True)    # 7 is a factor of 14 and 7
assert(commonFactor(11, 11, 11) == True)  # 11 is a factor of 11 and 11
assert(commonFactor(3, 10, 20) == False)  # 3 is not a factor of 10 or 20
assert(commonFactor(3, 5, 7) == False)    # 3 is not a factor of 5 or 7

# 0 is smallest, but not a factor. Don't crash!
assert(commonFactor(2, 10, 0) == False)

# This is the only case with zero that returns True:
assert(commonFactor(0, 0, 0) == True)     # 0 is a factor of 0 and 0
```

Notes:

- **We will not consider zero to be a factor of any number except for itself.**
- Remember, do not use strings, loops, lists, or anything else prohibited on page 1 of this quiz.

Begin your FR2 answer here or on the following page

You may begin or continue your FR2 answer here, if you wish

bonusCT: Code Tracing [2pts]

This question is optional, and intentionally quite difficult. Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def f(x): return x+5
def g(x): return f(x-3)
def h(x): return g(g(x)%f(x))
def bonusCt1(f, g, x):
    if (x > 0):
       return bonusCt1(g, h, -f(x))
    else:
       return f(g(h(x)))
print(bonusCt1(g, f, 4))
```