fullName: _____ andrewID: _____ section: ____

15-112 F26 Combined Comp2 and Quiz2 Version A
Time: 35 Minutes

Note: this contains both Comp2 and Quiz2.

- You have 35 minutes total to complete everything included here.
- The first 10 problems (7 CTs and 3 MCs) are Comp2.
- Comp2 also counts for 20 points on Quiz2.

You must write your name on this paper and hand this back in immediately after the assessment. If we do not receive it immediately, you will receive a zero on the assessment. Do not unstaple any pages. All pages must be handed in intact.

Do not use your own scrap paper. You should not need it, but if you must absolutely have scrap paper, raise your hand and we will provide some. Write your andrewID clearly on it and hand it in with your quiz. We will not grade anything on scrap paper.

You may not ask questions during the quiz, except for English-language clarification questions. If you are unsure about a problem, take your best guess.

Before and during the quiz, you may not view any other notes, prior work, websites or resources, including any form of AI. You may not use calculators, phones, laptops, or any other devices. You may not communicate with anyone else except for current 112 TAs or faculty during the assessment. All syllabus policies apply.

You may not discuss this test with anyone else, even briefly, in any form, until we have released grades. Failure to abide by these rules may result in an academic integrity violation.

Do not use material we have not yet covered. Thus, do not use strings, lists, tuples, sets, dictionaries, OOP, or recursion.

For all problems, assume that the required imports (such as 'import math') are present.

Do not open this or look inside (even briefly) before you are ready to begin. Do not spend more than the specified time noted above on this assessment.

Comp2

For each CT: Indicate what the code prints. Write your answer (and nothing else) in the box.
For each MC: Write your answer (as a single uppercase letter) in the box.

 # CT1:

```
x = 41
for x in range(3):
    x += 5
    print(x, end=',')
print(x)
```



 # CT2:

```
x = 24
for x in range(2, 2):
    print(x, end=',')
print(x)
```



 # CT3:

```
x = 47
for x in range(2, 5):
    print(x, end=',')
print(x)
```



 # CT4:

```
x = 44
for x in range(2, 10, 2):
    print(x, end=',')
print(x)
```



 # CT5:

```
x = 21
y = 50
for x in range(2, 5):
    for y in range(x, 5):
        print(10 * x + y, end=',')
print(10 * x + y)
```



 # CT6:

```
x = 5
while x <= 7:
    print(x, end=',')
    x += 2
print(x)
```



 # CT7:

```
x = 29
for x in range(5):
    if x == 1:
        continue
    elif x == 4:
        break
    print(x, end=',')
print(x)
```



 # MC1:

What is NOT an advantage of using good style?

A. Good style makes you more employable.
B. Good style saves time.
C. Good style makes code run faster.
D. Good style avoids bugs.

 # MC2:

What are magic numbers?

A. Random numbers.
B. Constants like math.pi and math.e.
C. Very large and hard-to-read integers.
D. Constants in your code that hide their computation.

 # MC3:

Which of the following is NOT in our recommended style guide?

A. Do not use test functions (in the age of AI, they are not necessary).
B. Format your code cleanly, basically so it looks like our sample code.
C. If you have unused code, at least comment it out or, even better, just delete it.
D. Do not write repetitive code.

Quiz2 (Note: Comp2 counts for 20 pts on Quiz2.)

Fill in the Blank [20 pts, 10 pts each]

FitB1: [10 pts]
Fill in the blank so f(n) == g(n) for all integers n.

```
def f(n):
    total = 0
    for x in range(n):
        for y in range(n):
            total += 1
        total += 1
    return total

def g(n):
    # hint: what happens if n is negative?

    return _____________________________________________________________
```

FitB2: [10 pts]
Fill in the blank so f(n) == g(n) for all integers n.

```
def f(n):
    n = abs(n)
    result = 0
    while n > 0:
        if n % 10 == 0:
            result += 1
        n -= 1
    return result

def g(n):

    return _____________________________________________________________
```

FR1: nthAddish [30 pts]

Background: we will define the addish numbers (a coined term) as such:

- The 0th addish number is 75.
- If the nth addish number is A, then the (n+1)th addish number is A plus the sum of the digits in A.

Thus, here are the first several addish numbers:

n	nth addish number
0	75
1	87 == 75 + 7 + 5
2	102 == 87 + 8 + 7
3	105 == 102 + 1 + 0 + 2

With that, write the function nthAddish(n) that takes a non-negative int n and returns the nth addish number. Here are some test cases:

```
assert(nthAddish(0) == 75)
assert(nthAddish(1) == 87)
assert(nthAddish(2) == 102)
assert(nthAddish(3) == 105)
```

Write your solution on the next page.

FR2: largestDigitCount [30 pts]

Write the function largestDigitCount(n) that takes a possibly-negative int n and returns the number of times the largest digit of n occurs in n. For example, if n is 511355, the largest digit in n is 5, and there are three 5's in n, thus:

```
assert(largestDigitCount(511355) == 3)
```

Here are some more test cases:

```
assert(largestDigitCount(-199) == 2)
assert(largestDigitCount(1) == 1)
assert(largestDigitCount(1232321) == 2)
```

Write your solution on the next page.

BonusCT

These CTs are optional, and intended to be very challenging. They are worth very few points. Indicate what the following code prints. Place your answers (and nothing else) in the boxes below. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

bonusCT1: [1pt]

```
def bonusCt1(x):
    def f(x, y): return x ** y == y ** x
    def g(x, n): return x // 10 ** n % 10
    def h(x, n):
        if f(g(x, n), g(x, n//2)):
            print(2 * n, end='')
    for n in range(5): h(x, n)
bonusCt1(63245)
```

bonusCT2: [1pt]

```
import math
def bonusCt2(n):
    def f(n):
        return 0 if n < 10**-5 else n + f(n/2)
    return math.floor(f(n))
print(bonusCt2(234))
```