fullName: ______________________________ andrewID: ________________ section: ____

15-112 S26 Quiz2 Version A
Time: 25 Minutes

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.

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.

Multiple Choice [10 pts, 5 pts each]

Indicate your answer by filling in the dot(s). Unless otherwise specified, only fill in one dot for each question.

MC1: Which of the following ranges does not include any values (check all that apply):

○ A) range(-3)

○ B) range(-3, 3)

○ C) range(-3, 3, 10)

○ D) range(10, 3, -3)

○ E) range(10, -3, 3)

MC2: When the following code is run, how many times is the function f called?

```
for x in range(100):
    if x % 2 == 0:
        f(x)
        for y in range(100):
            f(y)
```

○ A) 50 times

○ B) 150 times

○ C) 5000 times

○ D) 5050 times

○ E) 10000 times

○ F) None of these

Fill in the Blank [10 pts, 5 pts each]

Fill in the two blanks in g(x) so that f(x) == g(x) for all integer values of x:

```
import math

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

def getKthDigit(n, k):
    n = abs(n)
    return n // 10**k % 10

def f(x):
    x = abs(x)
    t = 0
    # hint: the following range does not use the default step size:
    for k in range(0, digitCount(x), 2):
        t += getKthDigit(x, k)
    return t

def g(x):
    x = abs(x)
    t = 0
    while x > 0:

        t += __________________________________________ # <-- blank #1

        x //= _________________________________________ # <-- blank #2

    return t
```

Code Tracing [20 pts, 10 pts each]

CT1: [10 pts]
Indicate what the following code prints. Place your answer (and nothing else) in the box below. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

```
def ct1(x):
    t = 0
    while x > 0:
        d = x%10
        x //= 10
        if d % 4 == 0:
            break
        elif d % 2 == 0:
            continue
        else:
            t = 10*t + d
    return t

print(ct1(98765))
```

CT2: [10 pts]

Indicate what the following code prints. Place your answer (and nothing else) in the box below. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

```
def ct2(x):
    while x > 0:
        for y in range(0, 10, x%10):
            if y > 0:
                print(y, end='')
        x //= 10

print(ct2(47))
```

FR1: nthFused (n) [25 pts]
Note: for this problem, you do not have to write isPrime(n). You may assume it is already written for you. However, you must write any other helper functions that you use.

Background: we will say that an integer n is "fused" (a coined term) if:

- n is NOT negative, and
- n is NOT prime, and
- n is NOT the sum of two primes.

For example, -27 is not fused because -27 is negative.

Also, 19 is not fused because 19 is prime.

Finally, 18 is not fused because 18 == 5 + 13 and both 5 and 13 are prime, so 18 is the sum of two primes.

The first several fused numbers are: 0, 1, 27, 35, 51, 57,...

With that in mind, write the function nthFused(n) that takes a non-negative int n and returns the nth fused number.

Hint: you may wish to write the helper function isFused(n).

Here are some test cases for you:
```
    assert(nthFused(0) == 0)
    assert(nthFused(1) == 1)
    assert(nthFused(2) == 27)
    assert(nthFused(3) == 35)
    assert(nthFused(4) == 51)
    assert(nthFused(5) == 57)
```

Write your solution on the next page:

FR2: secondLargestOddDigit(n) [35 pts]
Write the function secondLargestOddDigit(n) that takes a possibly-negative int n and returns the second largest odd digit in n, or None if n contains less than two odd digits.

For example, if n = 12345, the odd digits are 1, 3, and 5, and the second largest of these is 3, so:

```
  assert(secondLargestOddDigit(12345) == 3)
```

Note: if a digit appears twice in a number, it can be both the largest and the second-largest. For example, if n = 1355, the odd digits are 1, 3, 5, and 5 again, so the largest of these is 5, but the second largest is also 5 (since there are two of them). Thus:

```
  assert(secondLargestOddDigit(1355) == 5)
```

Here are some more test cases for you:

```
    assert(secondLargestOddDigit(0) == None)
    assert(secondLargestOddDigit(1) == None)
    assert(secondLargestOddDigit(135) == 3)
    assert(secondLargestOddDigit(12345) == 3)
    assert(secondLargestOddDigit(1355) == 5)
    assert(secondLargestOddDigit(-797) == 7)
```

Write your solution here and 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 f(x, b):
    n = 1
    while n*b < x: n *= b
    return x % n
def bonusCt1(x, b):
    while b: x, b = f(x, b), b//5
    return x
print(bonusCt1(200100, 10))
```

bonusCT2: [1pt]

```
# hint: f(x) computes something familiar....
def f(x):
    r, z = 0, x-1
    while z-1:
        r += not bool(x%z)
        z -= 1
    return bool(r)
def bonusCt2():
    r, x = 0, 1
    while True:
        while True:
            x += 1
            if f(x): break
        r += x
        if not f(r): break
    return r
print(bonusCt2())
```