fullName: _____ andrewID: _____ section: ____

15-112 S26 Quiz1 Version A
Time: 20 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 loops, 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, 2 pts each]

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

MC1: What happens when the following code is run?

```
def f(x):
    if x % 2 == 0:
        return 10 / x
    else:
        return x + 'a'

print(f(3), f(4))
```

◯ A. It crashes with a ZeroDivisionError.

◯ B. It crashes with a TypeError.

◯ C. It crashes with a SyntaxError.

◯ D. It does not crash but has a logical error.

◯ E. It does not crash and prints: `aaa 2.5`

MC2. Given these functions, which of the following will crash? Select all that apply.

```
def f(x):
    return x, 2*x
def g(x, y):
    return x + 2*y
```

◯ A. `x = f(5)`

◯ B. `x, y = f(5)`

◯ C. `x = g(1, 2)`

◯ D. `x, y = g(1, 2)`

MC3. What does `((3 < 5) and (0 == 1//0))` evaluate to?

◯ A. `True`

◯ B. `False`

◯ C. None

◯ D. It crashes.

◯ E. None of the above.

MC4. What does `((3 < 5) or (0 == 1//0))` evaluate to?

◯ A. `True`

◯ B. `False`

◯ C. None

◯ D. It crashes.

◯ E. None of the above.

MC5. What does (1 and (0 or 3)) evaluate to?

◯ A. `True`

◯ B. `False`

◯ C. `0`

◯ D. `1`

◯ E. `3`

Code Tracing [40 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, y):
    print(x * 6)
    print(y * '7')
    print(x % 7)
    print(7 % x)

print(ct1(11, 3))
```

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(q, r):
    print(-q//10)
    print(abs(r - 5))
    print(1-2**3+4*5//6)
    return type(20/10) == type(20//10)

print(ct2(53, 3))
```

CT3: [10 pts]

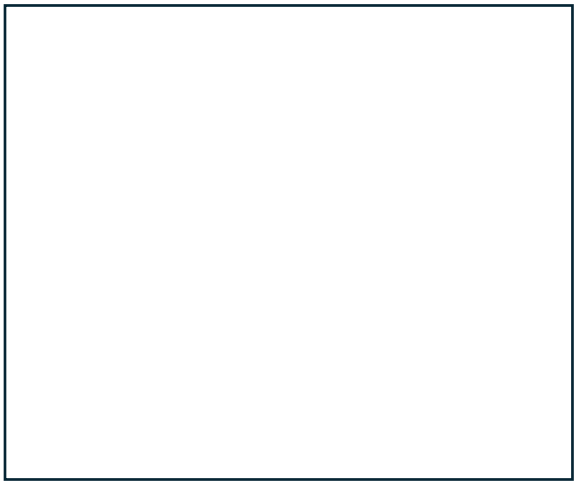
Indicate what the following code prints. Place your answer (and nothing else) in the box below. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

```
import math

def f(x):
    x **= 2
    x += 4
    return x/10 if x < 30 else x//10

def ct3(x):
    y = f(x + 1)
    print(y) # don't miss this!
    x += 1
    return f(x + math.ceil(y))

print(ct3(4))
```

CT4: [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 ct4(f):
    if f < 0:
        f += 3
    else:
        f -= 2
    if f < 0:
        f += 3
    z = f**2
    print('z:', z)
    if z > 3:
        return z**2%3
    elif z > 0:
        return 'z' * (z+1)
    else:
        return z

print(ct4(-4))
print(ct4(-3))
print(ct4(3))
```



FR: computeEncodedOperation [50 pts]

Background: this problem takes a spec which is a single non-negative integer that encodes an operation that can be +, -, or min.

The spec is 7 digits (though it can include leading 0's). The leftmost 3 digits are x, the middle digit is the opcode, and the rightmost 3 digits are y.

Thus, both x and y are integers between 0 and 999 (inclusive), and the opcode is an integer between 0 and 9 (inclusive).

So if the spec is 3451024, then x is 345, the opcode is 1, and y is 24.

Here are the meanings of the opcodes:

- 1 means plus
- 2 means minus
- any other digit means min

With that, write the function computeEncodedOperation(spec) that takes a legal spec as just described and returns the result of applying the operator associated with that opcode to the values x and y.

For example, consider computeEncodedOperation(3451024):

- Here, x is 345, the opcode is 1, and y is 24
- an opcode of 1 means plus
- so this returns 345 + 24, which is 369

As another example, consider computeEncodedOperation(3452024):

- Here, x is 345, the opcode is 2, and y is 24
- an opcode of 2 means minus
- so this returns 345 - 24, which is 321

Next, consider computeEncodedOperation(3453024):

- Here, x is 345, the opcode is 3, and y is 24
- an opcode of 3 means min
- so this returns min(345, 24), which is 24

Finally, consider computeEncodedOperation(31002):

- This example includes leading 0's
- Here, x is 3, the opcode is 1, and y is 2.
- an opcode of 1 means plus
- so this returns 3 + 2, which is 5

Thus:

```
assert(computeEncodedOperation(3451024) == 369) # 345 + 24 == 369
assert(computeEncodedOperation(3452024) == 321) # 345 - 24 == 321
assert(computeEncodedOperation(3453024) == 24)  # min(345, 24) == 24
assert(computeEncodedOperation(31002) == 5)     # 3 + 2 == 5
```

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 bonusCt1(n):
    def f(n):
        return int(str(n%100) * (n%10)) + 1
    def g(n):
        return 100*(n%10) + 10*(n//10%10) + (n//100%10)
    f,g,n = g,f,g(f(n))
    return f(1 + g(1 + n))
print(bonusCt1(12345))
```

bonusCT2: [1pt]

```
import math
def bonusCt2(x, y):
    if math.pow != bonusCt2:
        math.pow = bonusCt2
    return 42 if x > y else (math.pow(4, 2) + min(x, y) + max(x, y))
min = max # don't miss this!
max = min # or this!
print(bonusCt2(1, 3))
```