fullName:_______________________ andrewID:________________ section:___

15-112 F24
Quiz1 version B (30 min)

Read these instructions carefully before starting:

1. Quiz versions are color-coded. You must have a different version (color) of this quiz than the students sitting to your left and right.
2. Stop writing and submit the entire quiz when instructed by the proctor.
    - Do not unstaple any pages.
    - You must submit the entire quiz with all pages intact.
3. Do not discuss the quiz with anyone else until after 4pm.
    - This applies to everyone, including students in either lecture.
4. Do not use your own scrap paper.
    - You should not need scrap paper, there is plenty of room for you on the quiz.
    - However, if you absolutely must use scrap paper, raise your hand and we will provide some. Then, you must write your andrew id clearly on the scrap paper, and hand in the scrap paper with your paper quiz. We will not grade anything on your scrap paper.
5. You may not ask questions during the quiz.
    - The one exception is for English-language clarifications.
    - If you are unsure how to interpret a problem, just take your best guess.
6. Do not use any concepts (including built-in functions) not covered in the notes through week 1 / unit 1.
    - Do not use strings, loops, lists, tuples, dictionaries, sets, or recursion.
7. Do not hardcode your solutions.
    - We may test your code using additional test cases.
    - Hardcoding will receive zero points.
8. 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.
9. Good luck!

Short Answers (SA) [5 pts, ½ pt each]
All Short Answer (SA) answers must be **3 words or less**.
Any SA answers with more than 3 words will be marked as incorrect.

SA1. Rewrite the variable name numberofdogs in snake case.

SA2. Fill in the blank with one word: The * operator means "multiplication" when used with ints, but it means "multiple string concatenation" when used with a string and an int. This is an example of types affecting the ___________ of an operator.

SA3. What is the only right-associative arithmetic operator in Python? For example, + is left-associative because 2+3+4 is the same as (2+3)+4.

SA4. What is the default value returned by a function that does not include a return statement?

SA5. What is the rule Python uses for rounding with the built-in function round(x) when x is a float that ends with .5 (such as 1.5, 2.5, etc)?

SA6. What kind of error occurs if your code runs to completion but produces an incorrect result?

SA7. What kind of error occurs if you omit a close quote in your code?

SA8. What kind of error occurs if your code divides by zero while running?

SA9. Why doesn't (True or (1/0 == 0)) crash in Python?

SA10. Rewrite the variable name numberofdogs in camel case.

Code Tracing (CT)) [15 pts, 5 pts each]
For each CT, indicate what the code prints
Place your answer (and nothing else) in the box below the code.

Note: all the floats that are printed in these CTs have no more than one digit after the decimal point.

CT1:

```
def ct1(m, n):
    print(m // n, n // m)
    print(m / n, n / m)
    print(m % n, n % m)
    print(-m % n, -n % m)

print(ct1(2, 5))
```

CT2:

```
def f(x):
    print('f ', end='')
    return x > 0

def g(x):
    print('g ', end='')
    return x % 2 == 0

def ct2(x):
    print(g(x) or f(x))
    print(f(x) and g(x))
    return 10*x

print(ct2(-5))
print(ct2(6))
```

CT3:

```
def ct3(x, b):
    print(x*2 if not b else x/2)
    if x % 2 == 0:
        x *= 10
    if x > 10:
        x -= 10
    else:
        x = -x
    return f'x = {x}'

print(ct3(6, False))
print(ct3(7, True))
```

Free Response / FR1: isGoofy [40 pts]

Background: we will say that an int n is "goofy" (a coined term) if n's absolute value has 3 digits (so 100 <= |n| <= 999) and the sum of the first two digits equals the third digit.

For example:

- 257 is goofy because 2+5 == 7
- -257 is also goofy because 2+5 == 7
- 431 is not goofy because 4+3 != 1
- 1257 is not goofy because 1257 > 999

Also, a float f is goofy if the integer part of f is goofy. So:

- 257.9 is goofy because 257 is goofy
- -257.9 is goofy because -257 is goofy.

Non-numbers are not goofy.

Write the function isGoofy(n) that takes an arbitrary value n and returns True if n is goofy and False otherwise.

Here are some test cases:

```
    assert(isGoofy(257) == True)
    assert(isGoofy(-257) == True)
    assert(isGoofy(431) == False)
    assert(isGoofy(1257) == False)
    assert(isGoofy(257.9) == True)
    assert(isGoofy(-257.9) == True)
    assert(isGoofy(431.9) == False)
    assert(isGoofy(-431.9) == False)
    assert(isGoofy(112.1) == True)
    assert(isGoofy('do not crash here') == False)
```

Begin your FR1 answer on the next page.

Begin your answer to FR1 here:

Free Response / FR2: swapFirstAndLastDigit [40 pts]

Note: for this problem, assume that these functions are already written for you, so you can use them without writing them yourself:

- `digitCount(n)`
- `getKthDigit(n, k)`
- `setKthDigit(n, k, d)`

Recall:

- getKthDigit and setKthDigit count from the right (so 0 is the ones digit).
- setKthDigit only works for non-negative ints.

With that in mind, write the function swapFirstAndLastDigit(n) that takes a possibly-negative int n and returns an int with the first and last digits swapped. Be sure to preserve the sign of the number.

Here are some test cases:

```
    assert(swapFirstAndLastDigit(1234) == 4231)
    assert(swapFirstAndLastDigit(42) == 24)
    assert(swapFirstAndLastDigit(1) == 1)
    assert(swapFirstAndLastDigit(0) == 0)
    assert(swapFirstAndLastDigit(-1234) == -4231)
    assert(swapFirstAndLastDigit(-42) == -24)
    assert(swapFirstAndLastDigit(-1) == -1)
```

Begin your answer to FR2 on the next page.

Begin your answer to FR2 here:

Bonus Code Tracing (BonusCT)) [Optional, 4 pts, 2 pts each]
Bonus problems are not required.
For each CT, indicate what the code prints
Place your answer (and nothing else) in the box below the code.

BonusCT1:

```
def bonusCt1(x):
    # hint: eval('1+2') returns 3
    s = str(x)
    t = str(1-x) * (x-2)
    u = f'{s}{t}'
    return eval(u)
print(bonusCt1(20))
```

BonusCT2:

```
import math
def f(x, n): return int(str(x)*n)
def g(x): return f(f(x, x), x)
def bonusCt2(x):
    return math.floor(math.log10(g(x)))
print(bonusCt2(24))
```