fullName: _____ andrewID: _____ section: ____

15-112 S26 Quiz3 Version A
Time: 30 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 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.

True/False [10 pts, 2 pts each]

Indicate your answer by filling in the dot. Assume that both s and t are valid Python strings.

TF1: If `(s.startswith(t) == s.endswith(t))` is True, then `(s == t)` is True.

◯ True

◯ False

TF2: If `(s.replace(q, r) == s)` is True, then `(q == r)` is True.

◯ True

◯ False

TF3: `(s == s[:4] + s[4:])` is True for any string s regardless of its length.

◯ True

◯ False

TF4: If `s.isupper()` is True, then s cannot contain any spaces.

◯ True

◯ False

TF5: If `(s.strip() == s)` is True, then s cannot contain any spaces.

◯ True

◯ False

Fill in the Blank [5 pts]

Fill in the blank in g(x) so that f(x) == g(x) for all integer values of x (including negative values):

```
import math

def f(x):
    x = abs(x)
    r = 0
    while x > 0:
        r += x%10
        x //= 10
    return r

def g(x):
    s = str(x)
    r = 0
    for c in s:

        r += ____________________________________________ # <-- here
    return r
```

Code Tracing [20 pts, 5 pts each]

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

CT1: [5 pts]

```
def ct1(s):
    t = s
    u = f's{s}' * 2
    s += 's'
    u = u.replace(s, 'Z')
    return f'{s}-{t}-{u}'

print(ct1('B'))
```

CT2: [5 pts]

```
def ct2(s):
    r = ''
    for c in s:
        d = chr(ord(c) + 1)
        e = str(s.find(d))
        r += '.' if e in r else e
    return r

print(ct2('abbc'))
```

CT3: [5 pts]

```
import string

def ct3(s):
    for c in string.ascii_lowercase:
        if c in s:
            i = s.index(c)
            s = s[:i] + s[i+1:]
    return s

print(ct3('Ab9ba'))
```

CT4: [5 pts]

```
def ct4(s, d):
    r = ''
    for t in s.split(d):
        r += t[::-1]
    return r

print(ct4('ab,cd', ','))
print(ct4('ab,cd', 'c'))
```

FR1: isWordLadder(s) [30 pts]

Background: This problem defines "word ladder" differently than how it was defined in hw3. Here, a word ladder is a hyphen-separated string of at least two words where each word only contains uppercase letters, and the last letter of each word is the first letter of the next word.

For example, 'DOG-GNU' is a word ladder. Note that DOG ends in G and GNU starts with G. Similarly, 'DOG-GNU-URCHIN' is a word ladder.

These are not word ladders:

- `'DOG'      # not enough words`
- `'DOG-CAT'  # CAT does not start with G`
- `'DOG,GNU'  # not hyphen-separated`
- `'dog-gnu'  # not uppercase`
- `'5DOG-GNU' # 5 is not a letter`
- `'DOG--GNU' # word ladders cannot contain two consecutive dashes`

With that, write the function isWordLadder(s) that takes a string s and returns True if it is a word ladder as just defined and False otherwise.

Remember that you cannot use lists except for looping over them.

Write your solution on the next page:

FR2: getDogCount(petReport) [35 pts]

Background: we will say that a petReport is a possibly-empty multiline string listing how many of different kinds of pets a group of friends has, like so:

```
Ann: 1 frog, 2 dogs, 3 cats
Ben: 1 dog
Cam: 1 cat
Dan: 1 zebra, 13 dogs # Not many zebras, but lots of dogs
```

Each line starts with a person's name, a colon, a space, and then a comma-separated list of the number and kind of pets that person has. All animals have a single-word name (like 'dog' or 'cat' but not 'red panda'). Lines can end with comments, which start with a # sign, and should be ignored.

With that, write the function getDogCount(petReport) that takes a petReport string as just described and returns the total number of dogs owned by everyone in the report (so it returns an int).

For example:

```
    petReport = """\
Ann: 1 frog, 2 dogs, 3 cats
Ben: 1 dog
Cam: 1 cat
Dan: 1 zebra, 13 dogs # Not many zebras, but lots of dogs
"""
    assert(getDogCount(petReport) == 16) # 2 + 1 + 13 == 16
```

If there are no dogs, return 0, so:

```
    petReport = """\
Ann: 1 frog, 3 cats
Cam: 1 cat
"""
    assert(getDogCount(petReport) == 0)
```

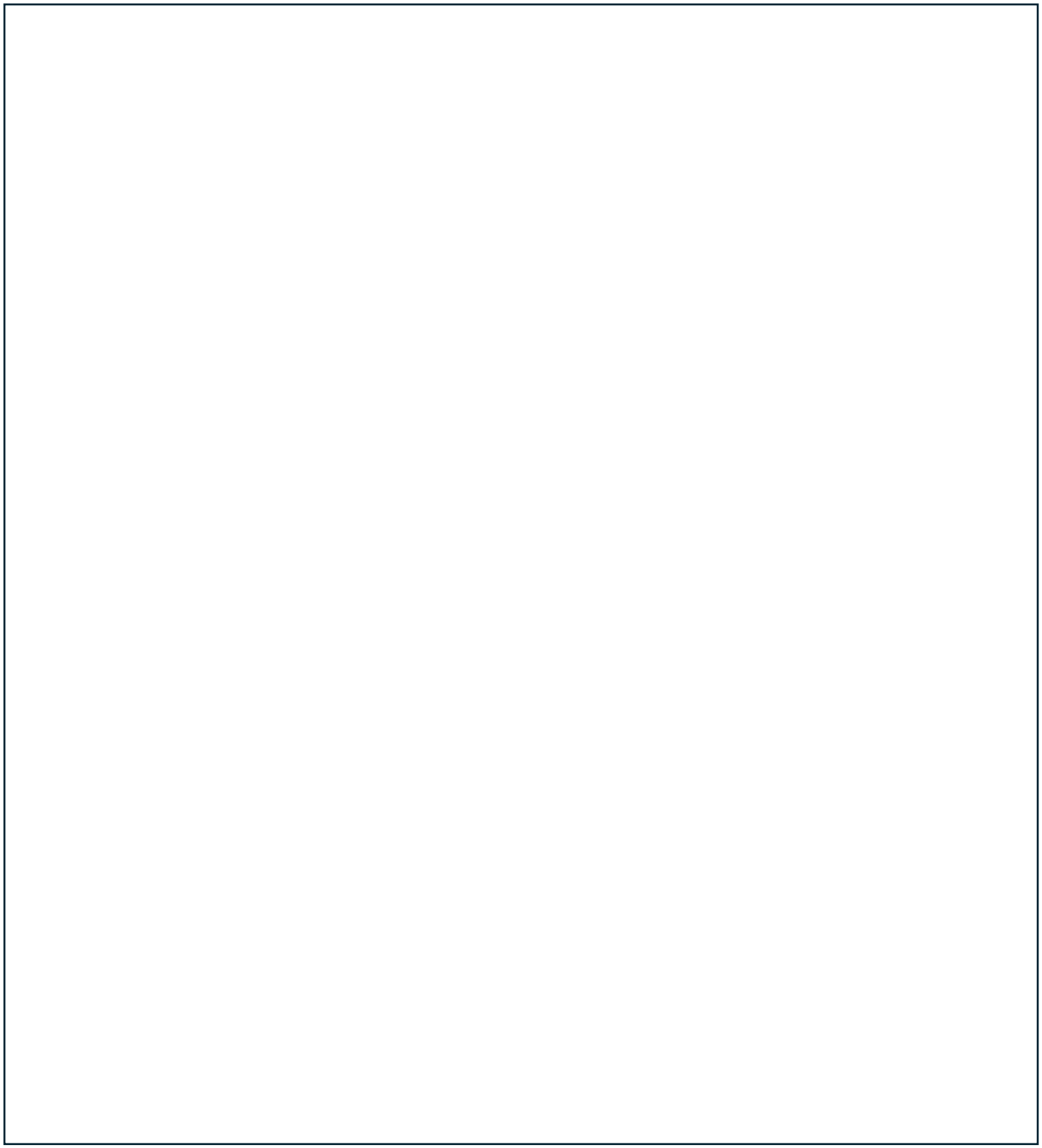
Also, the petReport can be empty, like so:

```
    petReport = ''
    assert(getDogCount(petReport) == 0)
```

Also, while it may seem strange, dogs can appear more than once on a line, like so:

```
    petReport = """\
Ann: 2 dogs, 3 cats, 1 dog, 1 mouse # 3 total dogs
"""
    assert(getDogCount(petReport) == 3)
```

Remember that you cannot use lists except for looping over them.



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 to the right of each code block. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

bonusCT1: [1pt]

```
import string
def bonusCt1(d):
    a = 0
    q = a + ord('a')
    for c in string.ascii_letters.lower(): a += ord(c) - q
    return d + a // len(string.ascii_lowercase)
print(bonusCt1(3))
```

bonusCT2: [1pt]

```
def bonusCt2(s):
    def f(s): return s[-1], s[:-1]
    while len(s) > 1:
        x, s = f(s)
        y, s = f(s)
        z, s = ('+',s) if s[-1].isdigit() else f(s)
        z = int(eval(x+z+y)) % 10
        s += str(z)
    return s
print(bonusCt2('+25*4/38'))
```

Extra space (if needed for scratch or for an FR):