fullName:_____andrewID:_____ recitationLetter:_____

15-112 S23

Quiz3 version B (25 min)

You **MUST** stop writing and hand in this **entire** quiz when instructed in lecture.

- You may not unstaple any pages.
- Failure to hand in an intact quiz will be considered cheating. Discussing the quiz with anyone in any way, even briefly, is cheating. (You may discuss it only once the quiz has been posted to the course website.)
- You may not use your own scrap paper. If you must use additional scrap paper, raise your hand and we will provide some. You must hand any scrap paper in with your paper quiz, and we will not grade it.
- Unless otherwise stated, you may not use any concepts (including builtin functions) which we have not covered in the notes in weeks 1-3. You may not use tuples, dictionaries, sets, or recursion.
- We may test your code using additional test cases. Do not hardcode.
- 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.

CT1: Code Tracing [15pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def f(c, i):
    return chr((ord(c) - ord('a') + i)%26 + ord('a'))

def ct1(s):
    i = 0
    r = ""
    s = s.lower()
    for c in s:
        if c.isalpha():
            r = r + f(c, i)
        else:
            r = r + c
            print(f'i = {i}')
        i += 1
    return r

print(ct1('6Z Ca t'))
```

CT2: Code Tracing [15pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def ct2(s):
    r = ''
    for i in range(len(s)):
        x = s[i]
        y = s[len(s)-1-i]
        if x.isupper() == y.isupper():
            r = y + r
            print(r[0], end = '+')
        else:
            r = x + r
            print(r[0], end = '-')
    print()
    return r

print(ct2('ABCdE'))
```

CT3: Code Tracing [15pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
import copy
def ct3(L):
    A = L
    B = copy.copy(L)
    x = L.pop()
    B += ['z']
    print(f'L = {L}')
    print(f'A = {A}')
    print(f'B = {B}')
    for i in range(len(L)):
        if type(L[i]) == int:
            x += i
    return x

print(ct3(['y', 20, 15, 'x', 3]))
```

Free Response 1: gradeAverages(grades) [55pts]

This problem takes a string (grades) that represents the homework grades for several students in the following format:

```
grades = '''Jimothy, 80, 50, 80
Chee, 99, 100, 102, 91
Jackie, 105
Bundt, 32, 50, 73, 84, 80'''
```

Write the function gradeAverages(grades) which returns a new string containing the average grade for each student in the following format (based on the example above):

```
'''Jimothy, 70
Chee, 98
Jackie, 105
Bundt, 64'''
```

The grade averages in the result should be **whole numbers which are rounded up at 0.5.**

Assume almostEqual(x, y) and rounded(n) are both supplied for you (though you are not required to use them). You must write all other helper functions you wish to use. Hint: It **may** be a good idea to write a helper function that takes a single line (like "Jimothy, 80, 50, 80") and returns a single line with that student's grade average (like "Jimothy, 70").

.split() and/or .splitlines() may be very useful here but you may only loop over the result, and may not index/slice the result or use list functions or list methods. For example, you may not use `len(s.splitlines())` or `max(s.split())` or `s.splitlines()[0]` etc. Do not use other list functions or list methods, or any prohibited concepts or functions which are not in the week 1, 2, or 3 notes.

Clarifications/Notes:

- You are guaranteed that the initial grades string will have a comma and one space between the name and each number, and between each number, e.g. "Bob, 102, 95" and there will not be any extra spaces at the beginning or end of any line.
- The result should have a comma and one space between the name and average, e.g. "Bob, 99"
- There are no blank lines. All grades strings will be properly formatted, with at least one student. Each student will have at least one grade. Student names will only contain letters. Each score is an integer.
- Do not include extra commas/spaces at the end of your lines, or an extra newline at the beginning or end of your result.

```
def testGradeAverages():
    grades1 = '''Jimothy, 80, 50, 80
Chee, 99, 100, 102, 91
Jackie, 105
Bundt, 32, 50, 73, 84, 80'''
    averages1 = '''Jimothy, 70
Chee, 98
Jackie, 105
Bundt, 64'''
    assert(gradeAverages(grades1) == averages1)

    grades2 = '''Jimothy, 0
Chee, 99'''
    averages2 = '''Jimothy, 0
Chee, 99'''
    assert(gradeAverages(grades2) == averages2)

    grades3 = '''Wilfred, 0, 10, 90
Wobbston, 95
Wiggles, 95, 90'''
    averages3 = '''Wilfred, 33
Wobbston, 95
Wiggles, 93'''
    #Note the rounding for Wilfred and Wiggles!
    assert(gradeAverages(grades3) == averages3)
    print('Passed')
testGradeAverages()
```

Begin your FR1 answer here or on the following page

You may begin or continue your FR1 answer here

You may continue your FR1 answer here

bonusCT: Code Tracing [2pts]

This question is optional, and intentionally quite difficult. Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def bonusCt1(r):
    for i in range(128):
        if (chr(i).isalpha()):
            r += ord(chr(i).lower()) - ord('a')
    r = str(r)
    return eval(f'{r[:1]}{r[-1]}/{r[1]}')
print(bonusCt1(4))
```