fullName:_____andrewID:_____ recitationLetter:_____

15-112 F23

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.
- You may not ask questions during the quiz, except for English-language clarifications. If you are unsure how to interpret a problem, take your best guess.
- You may not use any concepts (including builtin functions) which we have not covered in the notes in weeks 1-3 / units 1-2.
- You may not use list indexing or slicing, dictionaries, sets, 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.

```
import string
def ct(s):
    result = ""
    for c in s:
        if c in string.digits:
            result += c
    for c in s[3:5]:
        result += chr(ord(c) - 1)
    return result

print(ct("Dy\tB5f\n9?q0"))
```

CT2: Code Tracing [15pts]

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

```
def ct2(s, t):
    t0 = t
    for i in range(len(s)):
        if (s[i] == t[-1]):
            t = str(i) + t
        else:
            t = s[i].upper() + t
    t = t.replace(t0, 'N')
    return t[::-1]

print(ct2('wxyz', 'wx'))
```

Free Response 1: hasTwoAnagrams(s) [35pts]

Background: Two strings are anagrams if each string can be reordered into the other. Treat uppercase and lowercase as the same letter (so "Abca" and "BAAc" are anagrams).

With this in mind, write the function `hasTwoAnagrams(s)` which takes a single two-line string that may contain only letters and one newline character, and returns `True` if the two lines in the string are anagrams of each other, and `False` otherwise. Consider the following two-line string:

```
s = '''Abca
BAAc'''
```

`hasTwoAnagrams(s)` should return True for the case above, since `'Abca'` and `'BAAc'` are anagrams. This is described by the following test case:

```
assert(hasTwoAnagrams('Abca\nBAAc') == True)
```

As always, you may not use anything we have not yet covered in the week1-3 notes. You may not use `sort()`, `sorted()`, list indexing, or any list-based functions or methods.

Hints:

- We do not recommend reordering the letters in either line!
- There are many ways to solve the problem, but a particular string method makes this easier.

```
assert(hasTwoAnagrams('Abca\nBAAc') == True)
assert(hasTwoAnagrams('Word\nword') == True)
assert(hasTwoAnagrams('CAT\nact') == True)
assert(hasTwoAnagrams('abcd\nabc') == False)
assert(hasTwoAnagrams('abc\nabcd') == False)
assert(hasTwoAnagrams('abc\nabd') == False)
assert(hasTwoAnagrams('bat\nbaAt') == False)
assert(hasTwoAnagrams('abc\ndef') == False)
assert(hasTwoAnagrams('abc\nabc') == True)
assert(hasTwoAnagrams('\n') == True) #Both lines are blank
```

Begin your FR1 answer on the following page

Begin your FR1 answer here

Free Response 2: simpleEval(s) [35pts]

Write the function `simpleEval(s)` which takes a non-empty string consisting of only digits, `'+'`, `'-'`, and `'.'` and evaluates the expression, and returns this result as a float. You are guaranteed that the only operations present in the string are + and -. Any float in the strong will have a 0 in front of the decimal place. See the test cases for examples. Remember, this function should always return a float.

Note: As always, you may not use anything we have not yet covered in the week1-3 notes. You also may not use the built-in function `eval()` or any similar built-in that effectively evaluates the string for you.

```
assert(almostEqual(simpleEval('1+2+3'), 6.0))
assert(almostEqual(simpleEval('1-2-3'), -4.0))
assert(almostEqual(simpleEval('-100+50.5-25'), -74.5))
assert(almostEqual(simpleEval('0.3+0.3+0.3-0.9+1'), 1.0))
assert(almostEqual(simpleEval('-123.45+123.45+5-4+0.12'), 1.12))
```

Begin your FR2 answer here or on the following page

You may begin or continue your FR2 answer here, if you wish