fullName:_____

andrewID:_____

recitationLetter:_____

15-112 F22

Quiz3 version B

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) we have not covered in the notes this semester.
- You may not use list indexing, tuples, dictionaries, sets, or recursion.
- You may use string methods described in the notes that produce lists, but you may only loop over their return values as shown in the notes. (i.e. you may not store any lists in variables, index into them, use list methods etc.)
- We may test your code using additional test cases. Do not hardcode.
- Assume almostEqual(x, y) and roundHalfUp(n) are both supplied for you. You must write all other helper functions you wish to use.
- **Write your answers entirely inside the boxes!**

CT1: Code Tracing [12pts]

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

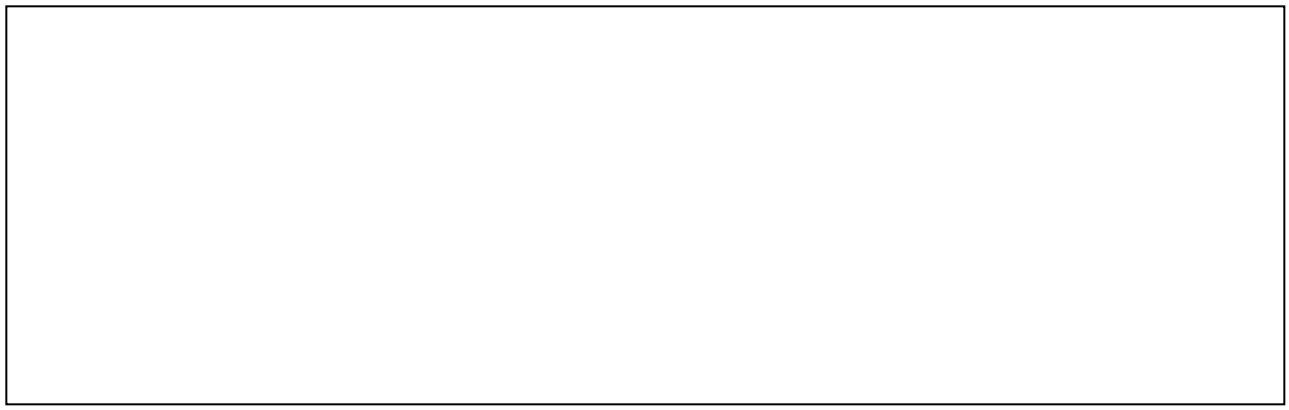
```
def ct1(s):
    for i in range(1,len(s)):
        s = s[i:] + s[:i]
        print(s)
    return s[::-1]

print(ct1('tldr'))
```

CT2: Code Tracing [12pts]

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

```
def ct2(s):
    t = s
    s = s + 'b'
    t = t + 'c'
    print(s + '-' + t)
    s = 'abcdba'
    t = ''
    v = chr(ord('b')-1)
    c = 0
    for w in s.split('b'):
        c += 1
        t = t + str(ord(w[0]) - ord(v))
    print(c)
    return t
print(ct2('a'))
```



Free Response 1: courseAverage(gradebook) [42pts]

This function will use a gradebook, which is a multi-line string where each line contains a student's name followed by their overall grade.

- If a student dropped the course, the score will be '--' and should be ignored.
- Aside from ':' and '--' and newlines, gradebook lines will not have special characters or spaces.
- Grades in the gradebook will only be digits (no floats).

Here is a sample gradebook:

```
gradebook = '''steve:82
mae:79
jimothy:--
sue:92'''
```

The currently-enrolled student grades are 82, 79, and 92. Jimothy has dropped the course.

With this in mind, write the function courseAverage that takes a gradebook, as just described, and returns the rounded average score for the whole course, or None if there are no valid grades in the gradebook. (Note: A 0 is a valid grade.) Remember, if a string method produces a list, you may not index into it. Here are some sample test cases:

```
gradebook = '''steve:82
mae:79
jimothy:--
sue:92'''
assert(type(courseAverage(gradebook) == int))
assert(courseAverage(gradebook) == 84)

gradebook = '''el:0
mae:79'''
assert(courseAverage(gradebook) == 40) # 39.5 rounds up

gradebook = '''jimothy:--
xarl:--'''
assert(courseAverage(gradebook) == None) # No grades

gradebook = '''zarl:0'''
assert(courseAverage(gradebook) == 0) # Watch for zeros
```

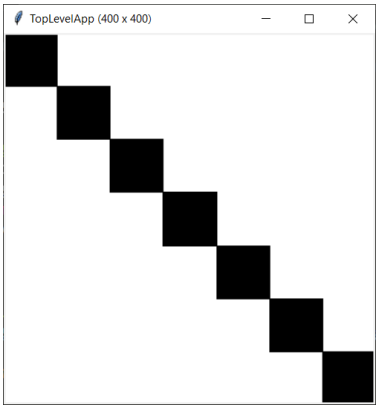
Begin your FR1 answer here

You may continue your FR1 answer here

Free Response 2: drawDiagonalSquares(canvas, w, h, n) [34pts]

Complete the drawDiagonalSquares(canvas, w, h, n) function so that it draws n squares from the top-left of the canvas to the bottom-right of the canvas as shown below. Note:

- w and h are positive integers which represent the width and height of the canvas, and n is a positive integer indicating the number of boxes to be drawn.
- The squares should touch at their corners.
- Assume that the canvas will always be square, so w == h.
- The squares should be filled, but the color and outlines do not matter.
- The example image is for drawDiagonalSquares(canvas, 400, 400, 7) but your function should work for all reasonable integer values of w, h, and n
- Assume cmu_112_graphics is imported, we will call runApp, and redrawAll is defined to call drawDiagonalSquares with reasonable values of w, h, and n
- The top-left and bottom-right squares should touch the corners of the canvas. Sometimes the canvas does not fully fill the window, so ignore the few pixels of whitespace at the edges of the image below.



You may begin your answer here or on the next page

You may begin or continue your FR2 answer here

bonusCT: Code Tracing [2pts bonus]

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

```
import string
def bonusCt1():
  s = string.ascii_letters[2:]
  e = 0
  for c in s:
    for d in s:
      e += ord(c) - ord(d) + 4
  return e//100
print(bonusCt1())
```