15-112 Fall 2022 Lecture 3
Quiz 4
35 minutes

Name: _____

Andrew ID: _____@andrew.cmu.edu

Section: _____

- **You may not use any books, notes, or electronic devices during this quiz.**
- **You may not ask questions about the quiz except for language clarifications.**
- **Show your work on the quiz (not scratch paper) to receive credit.**
- **If you use scratch paper, you must submit it with your Andrew ID on it, and we will ignore it.**
- **All code samples run without crashing unless we state otherwise. Assume any imports are already included as required.**
- **Do not use these topics: sets/dictionaries and recursion.**
- **You may use almostEqual() and rounded() without writing them. You must write everything else.**

Do not write below here

Question	**Points**	**Score**
1. CT	30	
2. FR: removeLargestValue	30	
3 FR: intersectLines	40	
4. Bonus	5 (bonus)	
TOTAL	100	

1. CT [30 pts]

Indicate what these print. Place your answers (and nothing else) in the box next to each block of code.

```
def ct1(L):

    M = copy.copy(L)
    N = copy.deepcopy(L)
    L[0] = L[1]
    M[1] = M[2]
    L[1][1] = 3
    N[0] = L[2]
    return (M, N)
L = [[5],[6,7],[8]]
print(ct1(L))
print(L) # don't miss this!
```

```
def ct2(L):
    rows, cols = len(L), len(L[0])
    M = [ ]
    for i in range(min(rows, cols)):
        M.append(L[i].pop(i))
    L.append(M)
L = [[1,2],[3,4],[5,6]]
ct2(L)
print(L)
```

2. Free Response: removeLargestValue [30 pts]

Write the mutating function removeLargestValue(L) that takes a rectangular 2d list L of integers, and mutates L so that both the row and the column containing its largest value are removed. You are guaranteed that the largest value in L occurs only once. Your function should return None.

Test Cases:

```
L = [ [ 1, 2, 3 ],
      [ 4, 5, 0 ] ]
assert(removeLargestValue(L) == None)
assert(L == [ [ 1, 3 ] ])

L = [ [ 1, 2, 3, 4 ],
      [ 5, 6, 5, 4 ],
      [ 3, 2, 1, 0 ] ]
assert(removeLargestValue(L) == None)
assert(L == [ [ 1, 3, 4 ],
              [ 3, 1, 0 ] ])

L = [ [ -1, -2],
      [ -4, -5 ] ]
assert(removeLargestValue(L) == None)
assert(L == [ [ -5 ] ])
```

This page is blank (for your removeLargestValue solution, if needed).

3. Free Response: intersectLines [40 pts]
Background: we can represent any line like so:

Ax + By = C

We will store the coefficients in a list. So [2,3,5] represents the line:

2x + 3y = 5

With this in mind, write the function intersectLines(L) that takes a 2d list L that contains at least two lines (where each line is represented by 3 numbers, as just noted). If all the lines intersect at a single point, your function should return the x value of that point. However, if the lines do not ALL intersect at that point, your function should return None.

Hint #1: to solve this, first find the point of intersection of the first two lines. Then, make sure the other lines also contain that point.

For example, say:

```
L = [[2,3,7],
     [3,2,8],
     [4,1,9]]
```

Start by intersecting these lines:

2x + 3y = 7
3x + 2y = 8

We did that by dividing each line by its first coefficient to get:

x + (3/2)y = (7/2)
x + (2/3)y = (8/3)

We then subtracted these equations to get:

(3/2)y - (2/3)y = (7/2) - (8/3)

We then solved for y, to get:

y = ((7/2) - (8/3)) / ((3/2) - (2/3)) = 1.0

Actually, we got 1.0000000000000002. Remember that these are floats!

We then substituted y into the first line to solve for x.

We found that these lines intersect at (2.0, 1.0).

We then verified that (2.0, 1.0) lies on the third line:

```
4x + 1y = 9
```

It does, so we returned the x value, 2.0. If it did not, we would have returned None.

Hint #2: We provide you with the function almostEqual(x, y), which you may use in your code. Our test function also uses it. Be sure to use almostEqual rather than == when comparing floats!

Hint #3: we found these two lines of code to be helpful in our helper function that checked if the first two lines intersect, where line1 is L[0] and line2 is L[1]:

```
a,b,c = line1
d,e,f = line2
```

Hint #4: You are guaranteed that none of the lines are parallel, and none of the lines are vertical.

Test Cases:

```
# These intersect at (1.0, 2.0):
assert(almostEqual(intersectLines([[2,3,8],
                                   [3,2,7]]), 1.0))

# These 3 lines all intersect at (2.0, 1.0):
assert(almostEqual(intersectLines([[2,3,7],
                                   [3,2,8],
                                   [4,1,9]]), 2.0))

# These 4 lines all intersect at (2.0, 1.0):
assert(almostEqual(intersectLines([[2,3,7],
                                   [3,2,8],
                                   [4,1,9],
                                   [5,-1,9]]), 2.0))

# The first two intersect at (1.0, 2.0) but the third does not:
assert(intersectLines([[2,3,8],
                       [3,2,7],
                       [4,1,5]]) == None)
```

This page is blank (for your intersectLines solution, if needed).

This page is blank (for your intersectLines solution, if needed).

4. Bonus [5 pts]

Indicate what these print. Place your answers (and nothing else) in the box next to each block of code.

```
import copy
def bonusCt1(L):
    while L:
        M = copy.deepcopy(L)
        L[0].append(sum(L.pop()))
        L.reverse()
    return M[0]
print(bonusCt1([list(range(i)) for i in list(range(2,8,2))]))
```

```
def bonusCt2(L, M):
    k = len(L)
    for r in range(k):
        for c in range(k):
            try:
                M[r][c] += L[-r][-c]
            except:
                pass
    return M
L = [[1,2],
     [3,4],
     [5,6]]
M = [[10, 20],
     [30, 40]]
print(bonusCt2(L, M))
```