Name: _____ Andrew Id: _____

15-112 Spring 2026 Quiz 4

Up to 25 minutes. No calculators, no notes, no books, no computers. Show your work!
Do not use dictionaries, sets, try/except, or recursion on this quiz.

1. (6 points) **Code Tracing: Lists**:

```
def CT(L):
    L.append(99)
    L = L + [10]
    sorted(L)
    print(L)
    L2 = L[:]
    print(L2 is L)
    print(L2 == L)
    L2 += [8]
    print(L2)
    L2[2:4] = [5, 6]
    L2.reverse()
    print(L2)
    L3 = L
    L3.pop()

L = [1, 2, 3]
CT(L)
print(L)
```

2. (6 points) **Fill-in The Blanks: Theater Seating**

Fill in the code for the function `drawSeating(app,seating)` to draw a theater seating grid. The function takes as input the comma-separated string `seating` that specifies for each row in the theater, which cells:

- Represent an available seat "O"
- Represent an occupied seat "X"
- Represent an aisle space (no seat) "A"

For example, `drawSeating(app, "OOOAOOO,OOXAOOX,AAAAAAA,XOOAOOO,OOOAOOO")` draws the theater seats in Figure 1. In this case:

- The first row, "OOOAOOO", has three available seats, one aisle space, and three available seats.
- The second row, "OOXAOOX", has two available seats, one occupied seat, one aisle space, two available seats, and one occupied seat.

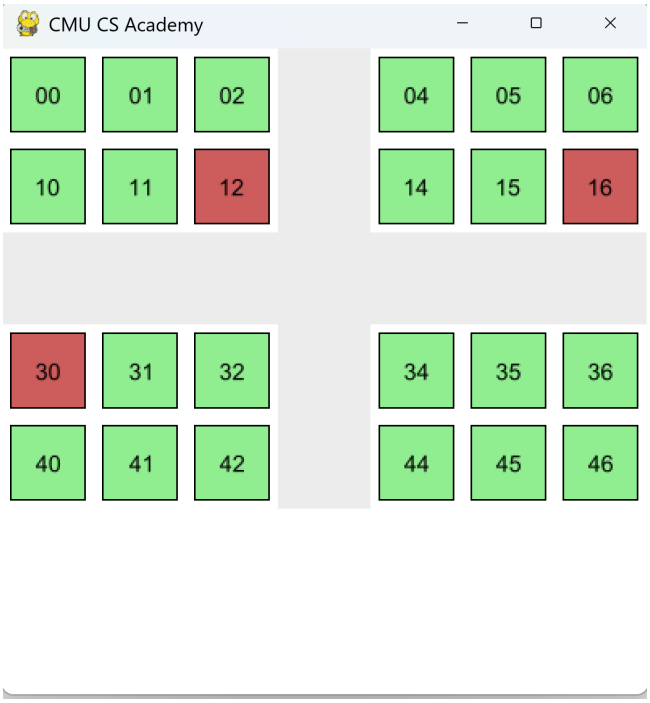

Figure 1: `drawSeating(app, "OOOAOOO,OOXAOOX,AAAAAAA,XOOAOOO,OOOAOOO")`

Note: Seats (O and X) should be drawn as smaller squares with white padding around them inside each grid cell. In contrast, aisle cells (A) should not have padding — the aisle background color should fill the entire grid cell; Figure 1.

You can assume that `app.width==app.height`. You can assume that `cmu_graphics` is already imported and there is a function `redrawAll(app)` that will call your function.

(Question continues on the next page)

```
def drawSeating(app, seating):
    padding = 5
    rowsList = []
    for row in seats_str.strip().split(","):
        rowsList.append(row)

    ### Number of columns and rows
    numCols = len(_______________________________)

    numRows = len(_______________________________)

    ### Calculate cellSize
    cellW = app.width // ____________________________
    cellH = (app.height * 0.9) // numRows
    cellSize = min(cellW, cellH) # Use the smaller of the two to avoid going off-screen
    for r in range(len(rowsList)):
        rowString = rowsList[r]
        for c in range(len(rowString)):
            char = rowString[c]
            ### Calculate top-left position of the cell

            x = ________________________________________________

            y = ________________________________________________
            ### Draw Aisle Background: Should fill the entire cell
            if char == 'A':
                drawRect(____________________, ____________________,

                    ____________________, ____________________, fill='silver', opacity=30)
            color = None

            if ___________________________:
                color = 'indianRed'

            elif ________________________:
                color = 'lightGreen'

            if color != None:  # Draw the Seat
                # The width/height of the drawn seat inside the cell,
                    # should take padding into account
                drawRect(x + padding,  y + padding,________________________,

                ________________________, fill=color, border='black')

                # Center the seat number label

                drawLabel(________________________,________________________,

                ________________________, size=max(8, cellSize/4))
```

3. (8 points) **Free Response**: Almost Palindrome

   A list is an "almost palindrome" if removing exactly one element would make it a palindrome.

   Write the function `checkAlmostPalindrome(L)` that takes a list `L` and returns:

   - The **index of the element that must be removed** to make it a palindrome; if the list is an almost palindrome.
   - **-1** if the list is not an almost palindrome (can't be made panlindrome with one removal)
   - **-1** if the list is already a palindrome (since no removal is needed).

   Your function must be non-destructive.

   Consider the following test cases:

```
assert checkAlmostPalindrome([1, 2, 3, 3, 2, 1]) == -1  # Already palindrome
assert checkAlmostPalindrome([1, 2, 3, 1]) == 1  # Remove 2
assert checkAlmostPalindrome([5, 1, 2, 3, 2, 1]) == 0  # Remove 5
# Can't make palindrome with one removal
assert checkAlmostPalindrome([1, 2, 3, 4, 5]) == -1
assert checkAlmostPalindrome([1, 1]) == -1  # Already palindrome
assert checkAlmostPalindrome([1, 2]) == -1  # Can't form palindrome
```

   Hint: A good strategy is to try removing one element at a time and check

Extra space for Problem 3.