# PART 2A (3 pts)

Use phrase with splicing and concatenation of two items to create the string 'tryher'.

phrase = 'thermometry'

result=phrase[-3:]+phrase[1:4]

ALTERNATE OPTIONS BASED ON RESULT TO FORM USING slice-[a:b]
'try': 8 or -3 in a and blank or 11 in b position.
'her': 1 or -10 in a position and 4 or -7 in b position

# PART 2B (3 pts)

# Use phrase with indexing and concatenation of three items to create the string 'mod'.

phrase = ['doberman']

result=phrase[0][-3]+phrase[0][1]+phrase[0][0]

ALTERNATE SOLUTIONS:
- first index position must be [0] for each index
- second index position:
  - 'm' can be -3 or 5,
  - 'o' can be 1 or -7,
  - 'd' can be 0 or -8

# PART 2C (3 pts)

Use lst with indexing and concatenation to create the string 'top'.

lst = ['computer']

result=lst[0][-3]+lst[0][1]+lst[0][3]

ALTERNATE SOLUTIONS:
- first position must be [0] for each index
- second position:
  - 't' can be -3 or 5,
  - 'o' can be 1 or -7,
  - 'p' can be 3 or -5

# PART 2D (3 pts)

# Use lst with indexing and the concatenation of two items to make the string 'fond'.

lst = [['first', 'second', 'third'], 'fourth']

NOTE: THIS PROBLEM DID NOT LIST PROPERLY TO USE SPLICING AS WELL. AS A RESULT, POINTS NOT DEDUCTED IF USED A SOLUTION THAT CONCATENATED 4 ITEMS AND NOT 2.

result=lst[-1][0:2]+lst[0][1][4:]
ALTERNATE SOLUTIONS (CONCATENATION OF TWO ITEMS):
#'fo': first position can be 1 or -1
#       second position (slice) can be [0:2], [:2], [-6:-4]

#'nd': first position must be 0 or -2
#       second position must be 1 or -1
#       third position (slice) must be [-2:], [4:]

#'f': lst[1][0]
#'ond': lst[0][1][3:] or lst[0][1][-3:]

ALTERNATE SOLUTIONS (CONCATENATION OF FOUR ITEMS):
#'f': lst[0][0][0] or lst[1][0] or lst[-1][0]..other options available as well
#'o': lst[0][1][3] or lst[1][1] or lst[-1][1]..other options
#'n': lst[0][1][-2] or lst[0][1][4]..other options
#'d': lst[0][1][-1] or lst[0][1][5] or lst[0][2][-1] or lst[0][-1][-1]..other options

# PART 2E (3 pts)
# Use lst with indexing to create ['one', 'two'].

lst = [['one', 'two'], 'three', ['four']]
result=lst[0]
result=lst[-2]

# PART 2F (3 pts)
# Use only lst with slicing to create the string 'Green'.

lst = [['Durham','Greensboro'], ['Charlotte'], 'Raleigh']
result=lst[0][-1][0:5]

ALTERNATE SOLUTIONS
#       first index position: 0 or -3
#       second index position: -1 or 1
#       slice: [:5] or [0:5] or [-10:-5]

# PART 2G (3 pts)
# Using slicing only, create a clone of lst
lst = ['pear', ['plum', 10], 'apple']

result = lst[:]

ALTERNATE SOLUTIONS
lst[0:]
lst[0:4]
lst[-3:]

# PART 2H (3 pts)
# Using the minimal slicing and concatenation, create the string "Hove".
phrase = 'Houston we have a problem.'

result = phrase[0:2] + phrase[13:15]

ALTERNATE SOLUTION:
"Ho": phrase[0:2] phrase[2:]

# PART 2I (3 pts)
# Using the minimal indexing and concatenation, create '823'.
lst = ['55', '24', '8', '3', '61']

result = lst[2] + lst[1][0] + lst[-2]

ALTERNATE SOLUTION:
"8": lst[2] or lst[-3] NOTE: OK if add second index(lst[2][0])
"2": first position: 1 or -4
    second position: 0 or -2
"3": lst[-2] or lst[3] NOTE: OK if add second index(lst[-2][0])

# PART 2J (3 pts)
# Using the minimal indexing and concatenation, create 'Fall2021'.
lst = ['Winter', 2020, 'Spring', '2021', 'fall', 2019, 'Fall']

result = lst[-1] + lst[3]

ALTERNATE SOLUTION:
"Fall": lst[-1] or lst[6]
"2021": lst[3] or lst[-4]
Problem 3A (10 points)

def compareString(text1, text2):
    if text1 >= text2:
        phrase = text1 + text2
    else:
        phrase = text2 + text1
    length = len(phrase)
    lst = [phrase, length]
    return lst

Problem 3B (10 points)
There are a number of ways they may have solved this. These are but 2 examples. Should you have questions, you can always create a version of this program in PyCharm and run their code. Though this is not required.

def purchase(toddler, child, adult, senior):
    price_child=child*5
    price_adult=adult*10
    price_senior=senior*5

    tix=toddler+child+adult+senior
    price=price_child+price_adult+price_senior
    print("Quantity:", tix, " Price:", price)

OR

def purchase(toddler, child, adult, senior):
    price=0
    tix=0

    #toddler info, no charge for tix
    tix += toddler

    #child info, $5 per tix
    tix += child
    price += (child*5)

    #adult info, $10 per tix
tix += adult
price +=(adult*10)

# senior info, $5 per tix
tix += senior
price +=(senior*5)

print("Quantity:", tix, " Price:", price)

Problem 4 (8 points)

# Part A (2 points)

What is the expected output of the program?

[['Michele', 'Teri'], 'Tori', ['Michele', 'Teri']]

# Part B (6 points)

Using only indexing and two new lines of code (assume they would be lines 9 and 10), modify list1 to create ['Tori', 'Tori', 'Tori']

list1[0] = 'Tori'
list1[-1] = 'Tori'

ALTERNATE SOLUTION

list1[0] = list1[1] OR list1[0] = list1[-2]

# Part 5 (6 points)
This program doesn't execute. It contains at least one error.

#PART A (3 points)

What line(s) of code contain the error?

Answer: Line 2, 11, and either line 20/21.

NOTE: This problem was designed to get students thinking about reading code. Line 2 Is a clear error. Line 11 is also an error, because it should be if state 1 == state 2.

They may have different interpretations of errors re: lines 20/21. Someone may decide (based on their review, that instead of changing lines 20/21, that a return statement is needed in the function (around line 15). The code should still work correctly throughout, if so.

#PART B (3 points)

What errors are present?
Answers: Students should catch at least 3 errors. One syntax and two semantics. See below.

Line 2: no ==
Line 11: should be state1 and state2 and not name1 and name2.
Line 20: assumes there is a return statement when calling compare.
Line 21 (if they identified line 20 as error): won’t output correct information (will output “None” as it is).

POSSIBILITY: Also (may not have line number): compare function could have a return statement as written (if not noting lines 20/21 as errors) OR line 20/21 need to be rewritten to NOT assign compare function to variables and print results.

#PART C (4 points)

Rewrite the line(s) of code with the correct version only as (for example) "X: corrected code", where X is the line number of the code to correct and corrected code is the rewritten line of code.

Answer:
Line 2: if len(state1)==len(state2):
Line 11: if state1==state2
Line 20: compare(name1, name3)
Line 21: delete this line

POSSIBILITY: Line 15: return XXX where XXX just needs to be a valid value in the program at this point. This will be acceptable here in lieu of line 20/21 corrections.