CS10 Python Programming Homework 4
20 points (List, tuples, sets)

(No dictionaries, classes allowed for this homework)

1. You must turn in your program listing and output for each program set. Each program set must have your student name, student ID, and program set number/description. Late homework will not be accepted for whatever reasons you may have.

*********_**for this homework, you are to submit your Program sets to Canvas**_ under Homework 4 link*********

   a. Name your files : HW4_PS1_ lastname_firstinitial.py for Program Set 1 and HW4_PS2_ lastname_firstinitial.py for PS2 and so on. PS means program set. If there are two program sets you will submit two files one for each program set. Example if your name is Joe Smith then the filename will be HW4_PS1_smith_j.py
   b. You must submit your homework by the deadline at the specified upload link on Canvas under homework 4. If the deadline is past, Program Sets will not be graded. Homework submitted via email attachment, comment in Canvas, Canvas message, or by any other method is not accepted and will be given a zero for no submission.
   c. if you do not follow the instructions on file naming provided in this section you will receive a zero for that program set that you did not correctly name the file.
   d. It is your responsibility to check if your homework is properly submitted to Canvas.

2. Please format your output properly, for example all dollar amounts should be printed with 2 decimal places when specified. Make sure that your output values are correct (check the calculations).

3. Use only the 'tools' in the topics we covered from lesson 1 to lesson 51 only. **No dictionary, classes. Global variables, break and continue are not allowed for this homework 4.**

4. Each student is expected to do their own work. **IF IDENTICAL PROGRAMS ARE SUBMITTED, EACH IDENTICAL PROGRAM WILL RECEIVE A SCORE OF ZERO.**

Grading:

Each program set must run correctly both syntactically, logically, and display the correct output exactly as specified. **If the program set does not run a zero will be given. If the program runs but does not display the correct output exactly as specified with proper formatting shown in the sample test run, a zero will be given**. If the program executes properly with proper syntax, logic, and displays the correct output with proper formatting as specified in the program set, you will receive the full points for that question. Then points will be deducted for not having proper:

a. Comments (1 pt. deducted for each occurrence)
   - Your name, description at the beginning of each program set. Short description of the what each section of your codes does.

b. Consistency/Readability (2 pts deducted for each occurrence)
   - Spacing(separate each section of codes with a blank line) but separate each function with 2 blank lines
   - Indentation
   - Style (proper naming of variables no a, b, c – use descriptive and mnemonics)
   - function **must include type hints or annotations** except for the main()
   - include **docstrings** for every function except the main()

c. Required elements
   - Use only 'tools' in the topics that have been covered in class. For example, in for this homework you are only allowed to use the topics covered in class up to sets(Lesson 1 to 51) only. **If you use dictionaries or files or classes and tools not in lesson 1 to lesson 51 your program set will result in a zero score for that program set.** Using **global variables and tools not in lesson 1 to lesson 51 like dictionaries, class, break, continue will result in zero score for that program set.**

d. Output (you **must provide the specified number of test runs or your program set will receive a zero score**)
   - to be displayed at the end of the program listing(codes) and commented
   - if no output(test runs) is provided for your uploaded file, a zero will be given for that program set.

Grading continued…

- must use test runs data when provided in the Program set question. Provide your own test runs if the program set does not ask for any.

Points will be deducted from grading items a. to d. above until your Program Set reaches zero points.

Program Set 1 (20 points)

Write functions for items a to j that carry out the following tasks for a list of integers.

a. Swap the first and last elements in the list.
b. Shift all elements by one to the right and move the last element into the first position. For example, 1 4 9 16 25 would be transformed into 25 1 4 9 16.
c. Replace all even elements with 0 (zeroes)
d. Replace each element except the first and last by the larger of its two neighbors.
e. Remove the middle element if the list length is odd, or the middle two elements if the length is even.
f. Move all even element to the front, otherwise preserving the order of the elements.
g. Return the second largest element in the list.
h. Return true if the list is currently sorted in increasing order.
i. Return true if the list contains two adjacent duplicate elements.
j. Return true if the list contains duplicate elements (which need not be adjacent).

The main() function to test each of the functions is included here for you. You cannot change the template codes in the main() function. Test runs 1, 2, 3 list values (ONE_TEN) are provided in which you have to use. You must use the same function names that is provide for you below. Using other function names will result in a zero for this program. You are not allowed to change any of the codes provided in the main() function. Adding, deleting, or omitting any of the codes provided in the main() function will result in a zero for the whole program that you submit. Call the functions that are highlighted in blue.

```
#Program to test functions a to j.
#
#Define constant variables.
ONE_TEN = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]              #this is test run 1
#ONE_TEN = [12, 20, 10, 14, 54, 16, 75, 38, 79, 103] #this is test run 2,
#ONE_TEN = [1, 25, 25, 4, 5, 4, 7, 60, 9, 10]         #this is test run 3,
                                                      #run test run 1 first, then
                                                      #comment out test run 1 and
                                                      #uncomment to run test run 2
                                                      #and so forth for test run 3

def main() :
    print("The original data for all functions is: ", ONE_TEN)

    #a. Demonstrate swapping the first and last element.
    data = list(ONE_TEN)
    swapFirstLast(data)          #call this function
    print("After swapping first and last: ", data)

    #b. Demonstrate shifting to the right.
    data = list(ONE_TEN)
    shiftRight(data)             #call this function
    print("After shifting right: ", data)

    #c. Demonstrate replacing even elements with zero.
    data = list(ONE_TEN)
    replaceEven(data)            #call this function
    print("After replacing even elements: ", data)

    #d. Demonstrate replacing values with the larger of their neighbors.
```

```
    data = list(ONE_TEN)
    replaceNeighbors(data)       #call this function
    print("After replacing with neighbors: ", data)

    #e. Demonstrate removing the middle element.
    data = list(ONE_TEN)
    removeMiddle(data)           #call this function
    print("After removing the middle element(s): ", data)

    #f. Demonstrate moving even elements to the front of the list.
    data = list(ONE_TEN)
    evenToFront(data)            #call this function
    print("After moving even elements: ", data)

    #g. Demonstrate finding the second largest value.
    print("The second largest value is: ", secondLargest(ONE_TEN))

    #h. Demonstrate testing if the list is in increasing order.
    print("The list is in increasing order: ", isIncreasing(ONE_TEN))

    #i. Demonstrate testing if the list contains adjacent duplicates.
    print("The list has adjacent duplicates: ", hasAdjacentDuplicate(ONE_TEN))

    #j. Demonstrate testing if the list contains duplicates.
    print("The list has duplicates: ", hasDuplicate(ONE_TEN))
```

#here are 2 sample functions for item a and b

```
def swapFirstLast(data:list)->????:       ##????you decide what is being returned
    '''Swap the first and last element in a list.'''
    #your codes


def shiftRight(data:list)->????:          ##????you decide what is being returned
    '''Shift the elements of list to the right.'''
    #your codes


##Complete the rest of the functions from items c to j

if __name__ == "__main__":
    main()
```

The output should look like this(blue font are to show you here the output results from your program)

Test Run 1

The original data for all functions is: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
After swapping first and last: [10, 2, 3, 4, 5, 6, 7, 8, 9, 1]
After shifting right: [10, 1, 2, 3, 4, 5, 6, 7, 8, 9]
After replacing even elements: [1, 0, 3, 0, 5, 0, 7, 0, 9, 0]
After replacing with neighbors: [1, 3, 4, 5, 6, 7, 8, 9, 10, 10]
After removing the middle element(s): [1, 2, 3, 4, 7, 8, 9, 10]
After moving even elements: [2, 4, 6, 8, 10, 1, 3, 5, 7, 9]
The second largest value is: 9
The list is in increasing order: True
The list has adjacent duplicates: False
The list has duplicates: False

Test Run 2

The original data for all functions is: [12, 20, 10, 14, 54, 16, 75, 38, 79, 103]
After swapping first and last: [103, 20, 10, 14, 54, 16, 75, 38, 79, 12]
After shifting right: [103, 12, 20, 10, 14, 54, 16, 75, 38, 79]
After replacing even elements: [0, 0, 0, 0, 0, 0, 75, 0, 79, 103]
After replacing with neighbors: [12, 12, 14, 54, 54, 75, 75, 79, 103, 103]
After removing the middle element(s): [12, 20, 10, 14, 75, 38, 79, 103]
After moving even elements: [12, 20, 10, 14, 54, 16, 38, 75, 79, 103]
The second largest value is: 79
The list is in increasing order: False
The list has adjacent duplicates: False
The list has duplicates: False

Test Run 3

The original data for all functions is: [1, 25, 25, 4, 5, 4, 7, 60, 9, 10]
After swapping first and last: [10, 25, 25, 4, 5, 4, 7, 60, 9, 1]
After shifting right: [10, 1, 25, 25, 4, 5, 4, 7, 60, 9]
After replacing even elements: [1, 25, 25, 0, 5, 0, 7, 0, 9, 0]
After replacing with neighbors: [1, 25, 25, 25, 25, 25, 60, 60, 60, 10]
After removing the middle element(s): [1, 25, 25, 4, 7, 60, 9, 10]
After moving even elements: [4, 4, 60, 10, 1, 25, 25, 5, 7, 9]
The second largest value is: 25
The list is in increasing order: False
The list has adjacent duplicates: True
The list has duplicates: True