Lab3. List Processing in Python

Question1. Write a function **find_average(student)** that takes student tuple as input and print student rollno, name, marks and average marks as output.

Test Cases:

1. stud1 = (1, “rex”, 60, 85, 70)
   find_average(stud1)

Modify the above function **find_average(student)** so that it processes a tuple of tuples.

2. stud2 = (2, “rex”, (80, 75, 90))
   find_average(stud2)

Question2. Write a weight management program that prompts the user to enter in 7 days of their body weight values as float numbers. Store them in list.
Then print first day weight, last day weight, 4^{th} day weight, highest weight, lowest weight and average weight.
Finally, print if average weight < lowest weight, then print “Your weight management is excellent”. Otherwise print “Your weight management is not good. Please take care of your diet”.

Question3. Write a function **lastN(lst, n)** that takes a list of integers and **n** and returns **n** largest numbers.

How many numbers you want to enter?: 6
Enter a number: 12
Enter a number: 32
Enter a number: 10
Enter a number: 9
Enter a number: 52
Enter a number: 45
How many largest numbers you want to find?: 3
Largest numbers are: 52, 45, 32

Question4. Given a list of strings, return a list with the strings in sorted order, except group all the strings that begin with 'x' first. Hint: this can be done by making 2 lists and sorting each of them before combining them.

Test Cases:

1. Input: ['mix', 'xyz', 'apple', 'xanadu', 'aardvark']
   Output: ['xanadu', 'xyz', 'aardvark', 'apple', 'mix']
2. Input: [‘ccc’,’bbb’,’aaa’,’xcc’,’xaa’]
   Output: [‘xaa’,’xcc’,’aaa’,’bbb’,’ccc’]
3. Input: [‘bbb’,’ccc’,’axx’,’xzz’,’xaa’]
   Output: [‘xaa’,’xzz’,’axx’,’bbb’,’ccc’]

Question5. Develop a function **sort_last()**. Given a list of non-empty tuples, return a list sorted in increasing order by the last element in each tuple. Hint: use a custom key= function to extract the last element form each tuple.

Test Cases:

1. Input: [(1, 7), (1, 3), (3, 4, 5), (2, 2)]
   Output: [(2, 2), (1, 3), (3, 4, 5), (1, 7)]
2. Input: [(1,3),(3,2),(2,1)]
   Output: [(2,1),(3,2),(1,3)]
3. Input: [(2,3),(1,2),(3,1)]
   Output: [(3,1),(1,2),(2,3)]

Question6. Other String Functions

a) Define a function **first()** that receives a tuple and returns its first element
b) Define a function **sort_first()** that receives a list of tuples and returns the sorted
c) Print lists in sorted order
d) Define a function **middle()** that receives a a tuple and returns its middle element
e) Define a functino **sort_middle**() that receives a list of tuples and returns it sorted using the key middle
f) Print the list [(1,2,3), (2,1,4), (10,7,15), (20,4,50), (30, 6, 40)] in sorted order. Output should be: [(2, 1, 4), (1, 2, 3), (20, 4, 50), (30, 6, 40), (10, 7, 15)]

Question7. Develop a function **remove_adjacent().** Given a list of numbers, return a list where all adjacent same elements have been reduced to a single element. You may create a new list or modify the passed in list.

Test Cases:

1. Input: [1, 2, 2, 3] and output: [1, 2, 3]
2. Input: [2, 2, 3, 3, 3] and output: [2, 3]
3. Input: [ ]. Output: [ ].
4. Input: [2,5,5,6,6,7]
   Output: [2,5,6,7]
5. Input: [6,7,7,8,9,9]
   Output: [6,7,8,9]

Question8. Write a function **verbing()**. Given a string, if its length is at least 3, add 'ing' to its end. Unless it already ends in 'ing', in which case add 'ly' instead. If the string length is less than 3, leave it unchanged. Return the resulting string. So ‘hail’ yields: hailing; ‘swimming’ yields: swimmingly; ‘do’ yields: do.

Question9. Develop a function **not_bad()**. Given a string, find the first appearance of the substring 'not' and 'bad'. If the 'bad' follows the 'not', replace the whole 'not'...'bad' substring with 'good'.
Return the resulting string. So 'This dinner is not that bad!' yields: This dinner is good!