Homework 1: Big O and Unit Testing

Theory

1. Once the lists are initialized, the concatenation can be carried out by using the '+' operator.
    In pseudocode:
        Let the name of the first list be <list1>; the second list be <list2>; the concatenated list be <newlist>
        Concatenation process:
            newlist = list1 + list2
    Using the big O analysis:
        Initialization of each of the lists (list1, list2):
            O(n) – linear, where n equals the size of the list (list1 or list2) being initialized
        Concatenation of the lists:
            O(M) – linear, where m equals the size of the concatenated list (newlist) adjusted by a, the resize constant for a list.
                M = a*(length of the concatenated list)
        Summarized analysis – O(M), linear

2. (a)
    Initializations –
        Count = 0
        MaxCount = 0
        SubIndex = 0
    Initial verification for if the input list of integers is of null length – if yes, skip to the end of the loop
        Define function (argument -> list of integers):
            Iterate through each element of the input but the 1st one
                Check if this element is bigger or smaller than the previous element
                    If it is bigger, note this down by adding one to count
                    Compare the current count to the max count
                        If it is bigger, we have found a new longest subsequence
                        Note the subsequence start index down in the subIndex
                If it is smaller, set the current count equal to zero

Returnlist = [0]*MaxCount
Replace the zeros of Returnlist with the elements of the subsequence using the starting index to the input list as (subIndex - MaxCount)

(b) The initializations of the variables can be classified as O(n) – linearly increasing runtime with respect to the size of the individual variables. The net effect can be defined as O(n), n being the size of the biggest variable in terms of space occupied.
Runtime for the function
If loop verification for the length of the list can be classified as O(1) – constant runtime
Iterating through the elements of the list can be classified as O(n) – linearly increasing runtime with respect to the size of the list of integers that is input to the function.
If loop verification for size inequalities of the list elements can be classified as O(1) – constant runtime
Arithmetic that is a part of this if true statement can be classified as O(1) - constant runtime.
For loop used to generate the function return list can be classified as O(n) – linearly increasing runtime with respect to the size of the return list