CS2302 Data Structures
Spring 2023
Lists Exercises

Write the following functions. See the commented out part of the program for expected results. Also, make sure your functions don't modify the list `L` received as parameter.

1. Write the function `all_equal(L)` that receives a list `L` and determines if all elements in `L` are equal. Your function must use a for loop.

2. Write the function `greater_than_x(L, x)` that receives a list `L` and an integer `x` and returns a list containing the elements in `L` that are greater than `x` in the order they appear in `L`. Your function must use a for loop and the append function.

3. Write the function `greater_than_x_lc(L, x)` that receives a list `L` and an integer `x` and returns a list containing the elements in `L` that are greater than `x` in the order they appear in `L`. Your function must use list comprehension and no loops or recursion.

4. Write the function `split_even_odd_index(L)` that receives a list `L` and returns two lists, one containing the elements of `L` that have an even index ([`L[0]`, `L[2]`, etc.) and one containing the elements of `L` that have an odd index. Your function must use a for loop (or more) and the append function.

5. Write the function `split_even_odd_index_s(L)` that receives a list `L` and returns two lists, one containing the elements of `L` that have an even index ([`L[0]`, `L[2]`, etc.) and one containing the elements of `L` that have an odd index. Your function must use list slicing and no loops or recursion.

6. Write the function `split_even_odd(L)` that receives a list `L` and returns two lists, one containing the elements of `L` that are even and one containing the elements of `L` that are odd. Your function must use a for loop (or more) and the append function.

7. Write the function `split_even_odd_lc(L)` that receives a list `L` and returns two lists, one containing the elements of `L` that are even and one containing the elements of `L` that are odd. Your function must use list comprehension and no loops or recursion.

8. Write the function `split_middle(L)` that receives a list `L` and returns two lists, one containing the first half of `L` and one containing the second half of `L`. Your function must use list slicing and no loops or recursion.

9. Write the function `split_pivot(L)` that receives a list `L` and returns two lists, one containing the elements of `L` that are less than `L[0]` and one containing the elements of `L` that are greater or equal to `L[0]`. If `L` is empty, your function must return two empty lists. Your function must use a for loop and the append function.

10. Write the function `split_pivot_lc(L)` that receives a list `L` and returns two lists, one containing the elements of `L` that are less than `L[0]` and one containing the elements of `L` that are greater or equal to `L[0]`. If `L` is empty, your function must return two empty lists. Your function must use list comprehension and no loops or recursion.