CSC 2720 Data Structure
Lab 2

Due: 01/21/2023 @ 11:59PM

Requirements:

(Failure to follow the requirements will result in a score of Zero.)

1. You may use whatever IDEs / editors you like, but you must submit your responses on iCollege as **.python files**.
2. Your submissions will work exactly as required.
3. Make sure you submit a file that compiles.
4. Your submission will show an output. Should you recive a Zero for no output shown do not bother to email with "but the logic is perfect".
5. Your program's output must exactly match the specs (design, style) given here for each problem to pass the test cases.
6. Design refers to how well your code is written (i.e., is it clear, efficient, and elegant), while Style refers to the readability of your code (correct indentation, good variable names). Add comments to have necessary explanations for your program.
7. Add a "heading" at the very beginning of your .java files as follow:
   Your Name
   CSc 2720 Lab #N
   Lab time: put your lab time here
   Due time: put the due date here

In user content generated web-services - let's say YouTube - the process of de-duplication is of serious importance. One straight forward reason is same video by different names is just an extra cost in data-storage. So getting rid of exact duplicate content makes financial sense. Please be reminded that exact same content with different video qualities may not be candidates for removal via de-duplication.

Today we will explore on ways to do a de-duplication of videos where video filenames are presented as integers. For the purposes of our task, we will set a very narrow criterion for de-duplication: just the filenames.

Below is how the filenames are represented LST = [50, 11, 33, 21, 40, 50, 40, 40, 21]

Below is the expected output after de-duplication LST = [11, 21, 33, 40, 50]

ATTN :

Notice the reduced size of the output list. Also, take a look again at the input list and be reminded that the input list is not sorted!

Further, please be reminded that you cannot use library functions to either sort and or perform the de-duplication operation. Doing so will result in a score of Zero!

Exercise 1

(40 points) Write a Python program called ***DeDuplication*** that the program takes an input list [50, 11, 33, 21, 40, 50, 40, 40, 21] and generate an output list [11, 21, 33, 40, 50], which removes the duplicated elements and sorts the list. Users might give any input list. Test your program with different testcases (provide 2 or 3 test scenarios). In this lab, you can use a "list counter" to help you. (Hint: In this way, you need to know what the maximum element is in the input list. Then initialize a "counting list" and count the frequency for each element. After that iterate the "counting list" and print the elements whose frequencies are not zero.)

Exercise 2

(60 points) Write a Python program called ***BinarySearch*** for the **de-duplicated** list (make sure it is sorted). The program prompts the user for an integer **n** to search, then print if the integer is found. If the integer is found, the program also prints how many "checks" it takes to find the integer. Otherwise, the program can just print "Fail to find the input number…" or something similar. Test your program with different testcase (provide 2 or 3 test scenarios).