Introduction:

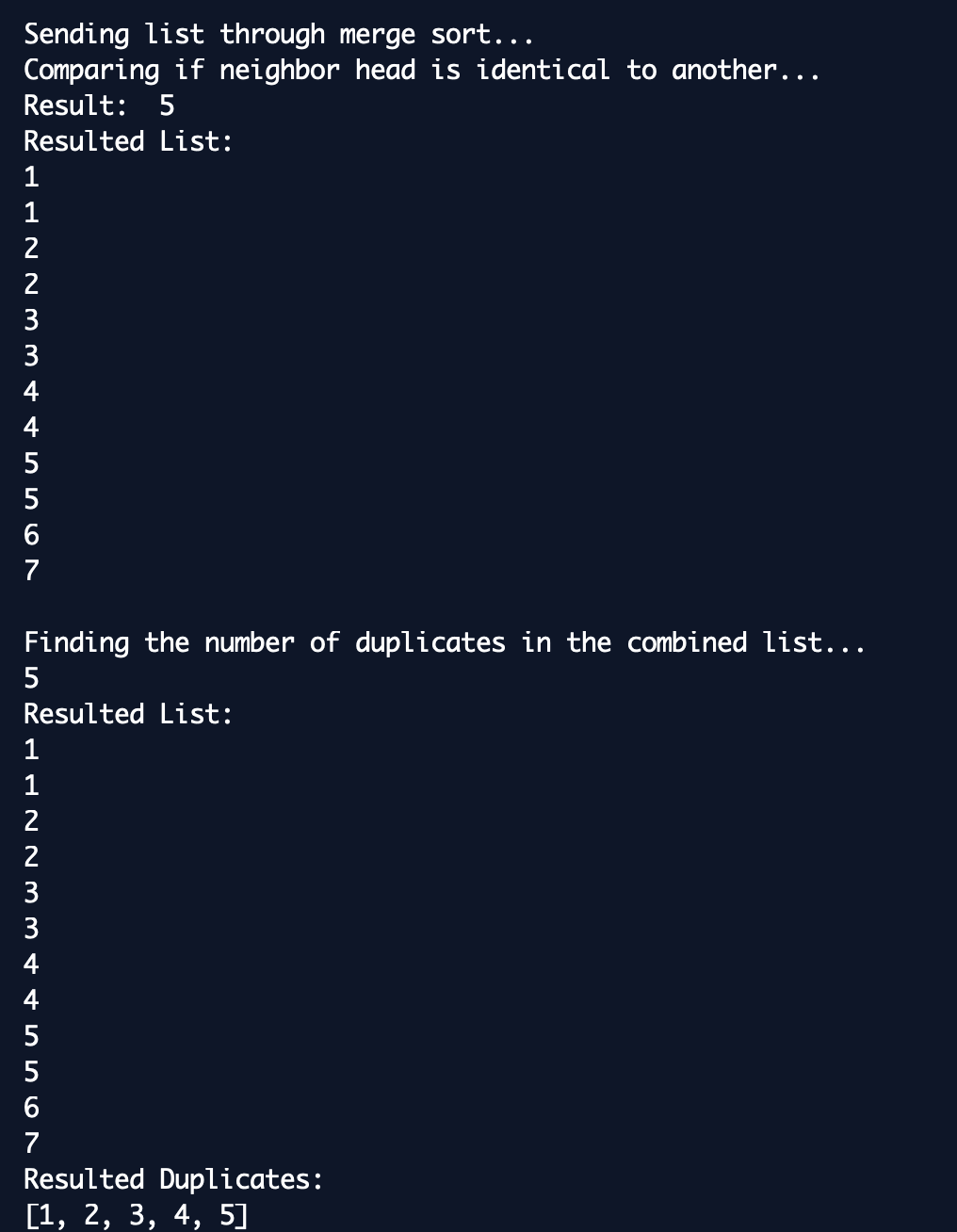
Link-List are a linear form of data structure without a fixed size meaning that inserting and deletion is less expensive as it would be with arrays. However, random access is not allowed, therefore, traversal always requires starting at the first node of the list. During this lab we will be comparing two text files containing hypothetical employee IDs from two different companies (activision and vivendi) in order for the companies to merge employee ID data to be one company. However, we must first combine them into one link list and check with four different methods to find duplicates of the IDs.

Proposed Solution and Implementation:

We are given the Link List Class and expected to create the following four methods accordingly to the data structure…. Compare Number: this method will be used to compare neighbor numbers and it will count the number of supplicates found in the given list. Bubble Sort: this method’s algorithm includes passing each number of a list to the right if it is greater than the following number until a sorted list is achieved. The link list has to be manipulated according to this algorithm which was quite simple to achieve. The link list head node is our starting point and it is checked whether it is greater the next value if not then the next node is checked with the following node. Merge Sort: this is a divide and conquer algorithm which splits the list by halves and sorts each one and merges them back together once sorted. In order to do this for a link list we must first create a method inside the link list class that counts the number of nodes inside the list to appropriately cut it in half. When the number of nodes are known then we sort each side and then merge! Search\_Duplicates: this method will find the duplicate values and insert them into a list that can be printed out for the user to know exactly which ones they are.

Experimental Results:

A screenshot of a cell phone

Description automatically generated

Conclusion: Link list finally make sense! Took me a whole three semesters but I am proud to say that the implementation in python was truly interesting. Even time complexity for each algorithm makes sense since before I thought it was just about counting for loops.