Project Overview for A2232JIsiLinkedList

The aim of this project is to develop a custom LinkedList class, named A2232JIsiLinkedList, that provides functionalities for sorting, searching, adding, removing, and replacing elements in a list. The class is generic and can hold any type that extends Comparable. It is designed to add elements in ascending order and maintain this order during subsequent manipulations. The project modifies Mark Allen Weiss' original LinkedList class to achieve these objectives.

Main Features:

Add Method with Sorting: The first main feature in the class is the ‘add(AnyType x)’ method. Unlike a typical LinkedList, where elements can be added arbitrarily, the elements here are always added in ascending order. This is accomplished by iterating through the list until the appropriate position is found for insertion. The time complexity of this operation is O(N) because each node needs to be checked in sequence.

Replace Method: I have designed a ‘replace(AnyType replace, AnyType x)’ method that swaps out a specific item in the list with a new one while maintaining the sorted order. The method first removes the target element and then adds the new one, thus re-establishing the sorted order of the list.

Mode and Count Calculations: In line with our need to perform statistical analysis, a ‘getMode()’ method has been implemented, which returns a Result object containing the mode and the count of the most frequent element. This utilizes a single pass algorithm with a time complexity of O(N).

Utility Methods: To enhance usability, I've included ‘showList()’ and ‘showList(int perLine)’ methods that print out the elements of the list in a readable manner. I've also included a ‘listSize()’ function to determine the size of the LinkedList, a common utility operation.

Authorship: For identification purposes, I included an ‘author()’ method that prints out the name of the author, in this case, James Luke C. Isidro.

Summary:

The project aims to extend a LinkedList class to add functionalities like sorted addition, item replacement, and statistical analysis. The ‘add()’ method ensures elements are added in a sorted fashion, the ‘replace()’ method allows for the replacement of elements while maintaining sort order, and the ‘getMode()’ method allows for a quick calculation of the mode and count of elements in the list. The class includes utility methods for displaying the list and identifying the author, enhancing its overall usability.