The class manages dynamic memory through a constructor, a copy constructor, a copy assignment operator, and a destructor. The constructor of the class should accept a word and a line number, initializing the corresponding data members. ~~; it should add the line number to its line number list only if it is not already there; however, it always increments the frequency count.~~ For the sake of simplicity in this assignment, the constructor should convert the upper case letters in the incoming word to lowercase before storing it.

The public interface of the class also includes the following operations:

➢ Determine whether a line number is in the line number list.
➢ Append a given number to the list of line numbers, only if it is not already there.
➢ Get the frequency count.
➢ Get a read-only pointer to the stored word.
➢ Get a read-only reference to the `NumList` object.
➢ Determine whether the stored word compares equal to, or comes before or after a given word. Use case insensitive alphabetical ordering of strings of characters when comparing two words.
➢ Print the word together with its frequency count and list of line numbers to a specified `ostream` object (like `cout`).

3.3 WordNode

This class represents the nodes in a singly linked list represented by the `WordList` class. Since `WordNode` is specifically designed to represent the nodes in a `WordList`, `WordNode` should be defined as a private `struct` data type nested within the `WordList` class.

A `WordNode` object stores two public data members:

➢ A `WordData` object
➢ A pointer called `next` pointing to another `WordNode` object

and provides the following public member functions:

➢ A constructor that accepts a word and a line number, setting the `next` data member to `nullptr`, and using the word and line number to initialize the `WordData` member.
➢ A constructor that accepts a word, a line number, and a pointer to another `WordNode` object, initializing the corresponding data members.

3.4 WordList

This class represents singly linked lists of `WordNode` objects. A `WordList` object stores and manages three data members:

➢ A *pointer* to the first node of the list
➢ A *pointer* to the last node of the list
➢ The size of the list