Lab Assignment - 2

1. Implement a *LinkedList* class which supports the following methods :
    a. Insert a node at the beginning
    b. Insert a node at the end
    c. Search for an element
    d. Get String representation
    e. Get length

Some notes :

- Pick relevant class attributes, and make them private. Provide getters and setters, wherever deemed appropriate.
- Write modular code. Do not dump every class and every method in the same file. Keep the *Node, LinkedList,* and *Client* classes separate.
- Choose your interfaces wisely! The client must know nothing about the implementation while using these functions provided. Avoid explicit use of Node class for building your list on the client side. The client should behave, as if it has no knowledge of the existence of the Node.
- Feel free to ask volunteers to explain logic to you in case you have forgotten, or have not done a Data Structures course yet.
- Make sure to get your design and implementation checked by a volunteer; It's relatively easy to get your code 'working' but much harder to design a clean interface.
- Naming conventions aids readability of your source code. Class names should be nouns, in mixed case, with each internal word capitalized. For example, BinaryTree, TapeRecorder, etc. Method names should be verbs, indicative of the desired operation, with the first letter in lowercase and the first letter of each internal word capitalized(camel-case). For example, getString(), increaseVolume(), etc. Variable names should typically be descriptive. Like method names, adopt the camel case. Despite the fact that Java allows you to use underscores and $'s as part of identifier names, avoid making your code look sparse with underscores or rich with dollars.