ECE 250 Project 1
Simple Calculator

1. Overview of Classes

I've implemented two classes to create the simple calculator function.

Class Node

- The Node class generates a node that stores two data types, a string name and a double value. The node generated also has a pointer to define the next node.

  Member Variables:
  1. value (double) - stores the value of the node
  2. name (string) - stores the name of the node
  3. next (Node*) - points to the next node in the Linked List

  No Member Functions

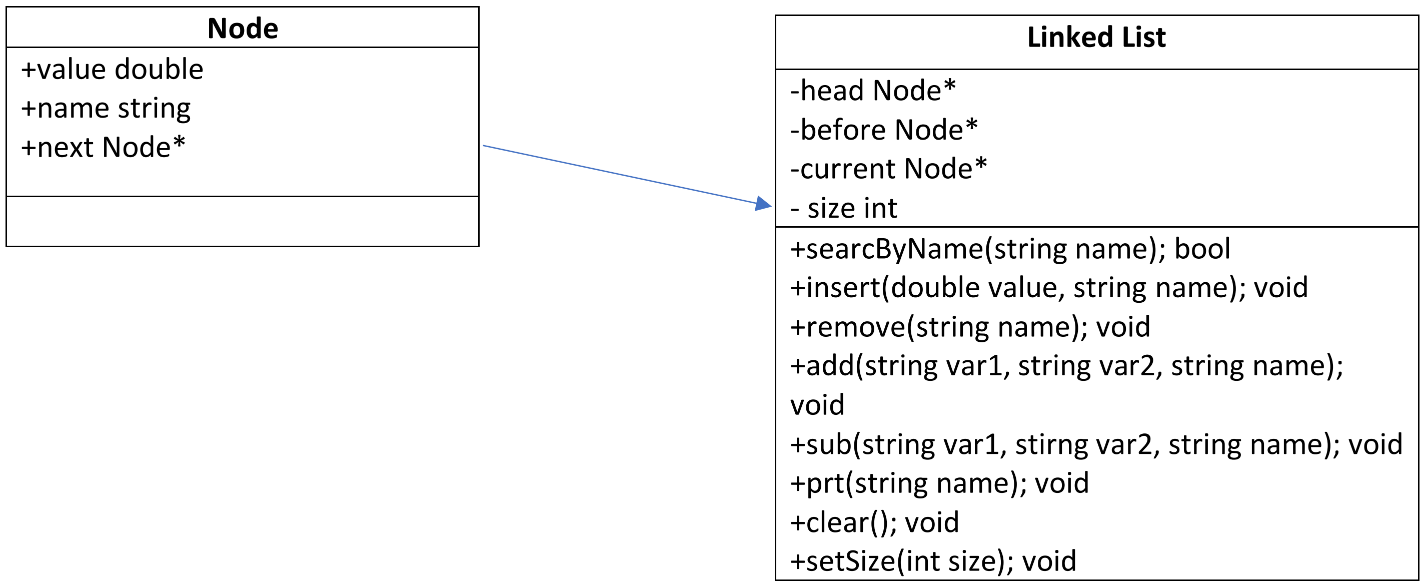
Class Linked List

- The Linked List class consists of multiple nodes that point to the next node. This class holds multiple functions that allows the user to search through the linked list to dynamically add and remove nodes. The user may also add and subtract values of nodes.

  Member Variables:
  1. head (Node*) - points to first node of the list. If the list is empty, it will be a nullptr
  2. before (Node*) - points to the node previous current node when traversing through the linked list
  3. current (Node*) - points to the current node when traversing through the linked list
  4. size (int) – sets the limit of nodes allowed in the linked list

  Member Functions:
  1. searchByName – takes in the name of a node and check if it exists within the linked list. Returns a bool depending on whether the name of the node was found in the linked list or not.
  2. insert – inserts a new node into the linked list if the nodes name is unique and if the list is not already full
  3. remove – takes in a node name, and if a node exists with that name, it is removed from the linked list
  4. add – takes in two node names and adds the values of the nodes, which is then stored in another node
  5. sub – takes in two node names and subtracts the values of the nodes, which is then stores in another node
  6. prt – takes in a node name, and prints the value of the node only if the node exists
  7. clear – deletes all the nodes in the linked list
  8. setSize – sets the max number of nodes allowed in the linked list

2. UML Class Design


Node
+value double
+name string
+next Node*
Linked List
-head Node*
-before Node*
-current Node*
- size int
+searcByName(string name); bool
+insert(double value, string name); void
+remove(string name); void
+add(string var1, string var2, string name); void
+sub(string var1, stirng var2, string name); void
+prt(string name); void
+clear(); void
+setSize(int size); void

3. Constructors/Destructors and Design Decisions

<u>Node Class:</u>
The constructor for this class passes 2 parameters, which sets the node's value and name. The variable next points to a nullptr when the node is constructed. During run time when nodes are added to the linked list, Node* next will point to the next node in the linked list.

<u>Linked List Class:</u>
The constructor for this class creates an empty linked list, where node* head is set to a nullptr, and the list can be populated at run time.
For the destructor, the clear function is used to delete all nodes stored in the linked list to avoid memory leaks.

No operators were overloaded in my code, as there was no need.

4. Runtime Implementation

Since all the requested commands need to make sure the node exists in the linked list, a run time of O(n) is implemented, with n being the number of nodes in the linked list. The run time O(n) is implemented through the searchByName method which loops through the entire linked list. For the CRT command that is supposed to run at O(1), it directly sets the size of the linked list to the value that was passed, allowing the code to execute instantly.