ECE250 Project 1: A Simple Calculator

1. Class Node
Represents a single node in a Linked List

1.1 Member variables of Node
All member variables are private, in accordance with encapsulation.

- `val` stores a double and represents the value associated with the node.
- `name` stores a string and represents the name of the variable associated with the node.
- `next` stores a reference to the next node in the linked list.

1.2 Member functions of Node
Since all member variables are private, their values can be accessed through public getter/setter functions.

- get_name(), get_value() and get_next_node() return the name, value and next node respectively for itself.
- set_name(), set_value() and set_next_node() allows the name, value and next node to be assigned for itself.

Also,

- Constructor of the Node class takes two parameters, String s and Double n. This initializes the Node with name s, and value n. Additionally, the next node is initialized as `nullptr`.

2. Class Calculator
Represents a simple calculator that stores variables in a linked list

2.1 Member variables of Calculator
All member variables are private.

- max_size stores the maximum number of variables that can be stored in the linked list. This is stored as an integer.
- current_size stores the current number of variables in the linked list. This is stored as an integer.
- head stores a pointer to an object of the class Node, and represents the head of the linked list.

2.2 Member functions of Calculator
All member functions are public.

- Constructor of the Calculator class takes an integer n to set the maximum size of the linked list. It initializes the linked list with a current size of `0` and sets head at nullptr.
- Destructor of the Calculator class traverses through the linked list and deletes each node.
- find_node takes in one parameter: String x that represents the variable name of the node that needs to be found in the linked list. It traverses through the linked list, hence, the asymptotic upper bound is O(n).
    - If the node is found, it returns a pointer to the Node. Otherwise, it returns nullptr.

- insert_node takes in two parameters: String x and Double val that represents the data associated with the node to be added to the linked list. It does not return a value.
    - The function calls find_node to search the existing linked list and see if a Node with the variable name already exists. If no previous node with the name is found, it is added to the beginning of the linked list.
    - If there are no nodes in the linked list, the Node is added immediately.
    - Hence, the asymptotic upper bound is O(n).
- remove_node takes in one parameter: String x that consists of the variable name of the node to be deleted. It does not return a value.
    - First, there's a check to see if the head of the linked list is the node to be removed. If note, the function searches the existing linked list to see if a Node with the variable name already exists.
    - Hence, the asymptotic upper bound is O(n).
- print_node_value takes in one parameter: String x that consists of the variable name of the node to be printed. It does not return a value. It calls the find_node function, so the asymptotic upper bound is O(n).
- arithmetic_op_nodes takes in three parameters: String x, String y, String z and String operation.
    - String x and String y represent the variable name of the Nodes that serve as the operands of the mathematical operation.
    - String z represents the variable name of the Node where the result is to be stored.
    - String operation indicates whether to add or subtract the Node values.
    - Traverse through the linked list and find all three nodes required for the calculation. Since all nodes are found at the first time of traversing the linked list, the asymptotic upper bound is O(n).