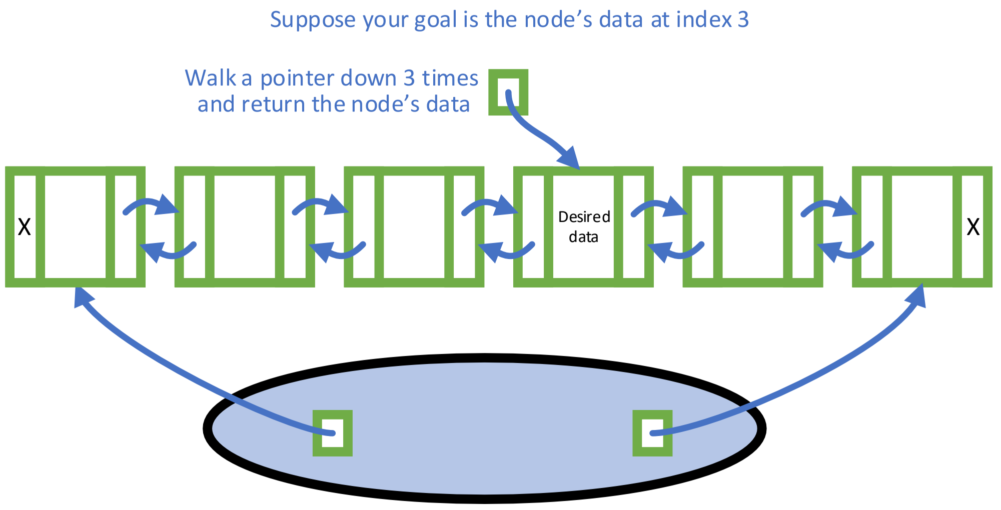
Homework - Linked List Generalized Methods

Brad Peterson – Weber State University

Goal: To understand how to work with basic methods of a doubly linked list.

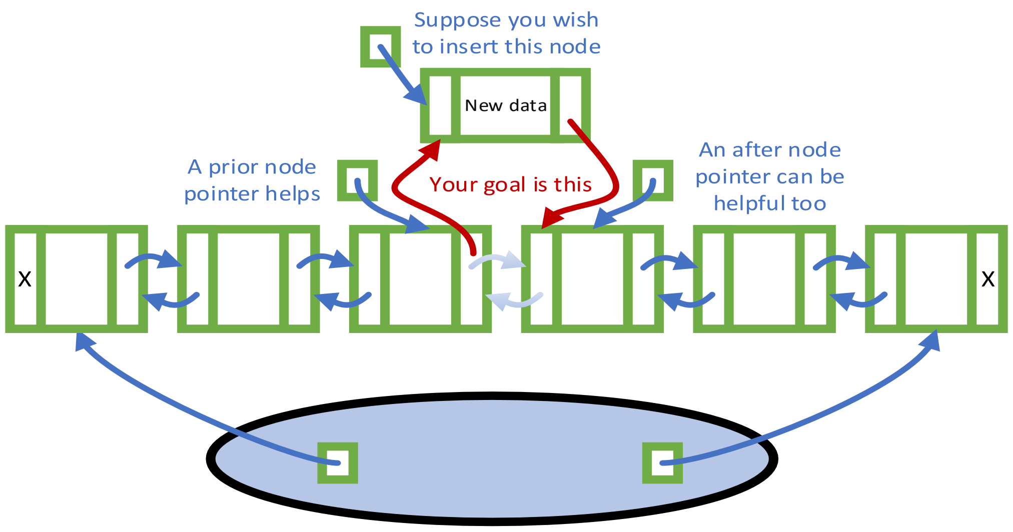
Assignment: Complete the .cpp file so that all tests succeed. Please read the instructions for these methods carefully. I will be grading based on your ability to meet these requirements.

- `T get (const unsigned int index) const.` This method accepts an integer. The method should then traverse to and return the kth node in the list. This is also zero based. It should throw an error (i.e. throw std::out_of_range) if the index is invalid.

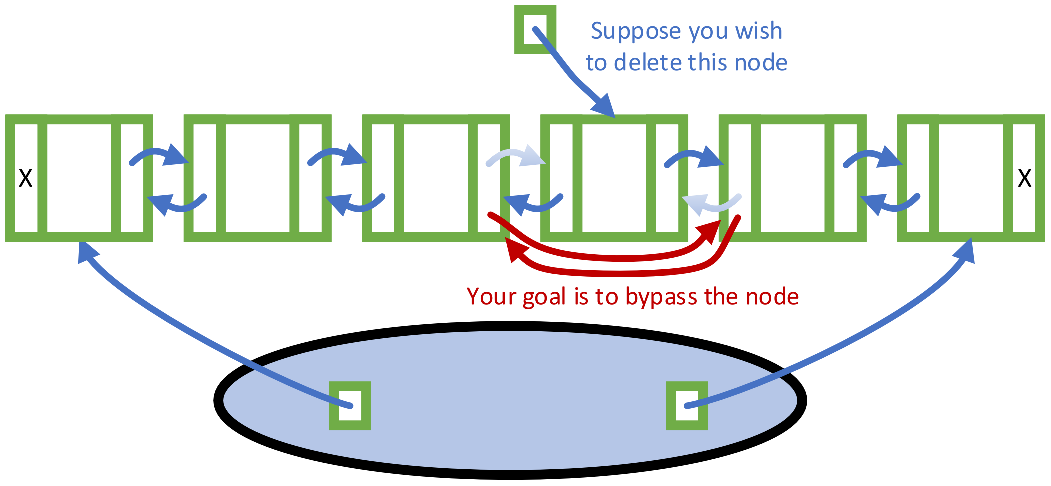

Suppose your goal is the node's data at index 3
Walk a pointer down 3 times and return the node's data
Desired data
X
X

- `T& operator[](const unsigned int index) const.` This is the overloaded [] operator. It is just a method, don't let the syntax scare you. It's just a method that accepts an integer. This method is very similar to `get`. It should return the node's value by reference (you don't have to do anything special for this, the T& returns it by reference for you). It should throw an error (i.e. throw std::out_of_range) if the index is invalid.

- `void insert (const unsigned int index, const T& value).` This method accepts an integer and a value. The method will insert a new node at that index. Anything already at that index is automatically shifted up by one index. It can also insert at position 0 (before all nodes), and at a position just after the last node. This index is zero based, meaning it starts counting from zero. It cannot accept indexes past the end of the list.


Suppose you wish to insert this node
New data
A prior node pointer helps
Your goal is this
An after node pointer can be helpful too
X
X

- `void remove(const unsigned int index)`. This method accepts an integer. The method should then traverse to and delete the kth node in the list. This is zero based. Make sure you handle all scenarios (empty list, one node, first node, last node, and the general scenario.)
-


Suppose you wish to delete this node
X
X
Your goal is to bypass the node

- `void removeAllInstances(const T& value)`. This method accepts a value. The method should then remove all instances of that value. **It cannot iterate down the list more than once, or in other words, it must traverse while it deletes. It cannot loop which calls another method to loop and find a value to delete**. (Some students have created their own **private** method which accepts a reference to the node pointer, then updates that pointer to point to the next node in line. Note that the pointer must be passed in by reference, so that the caller can have the updated pointer as well.)

All methods should use a linked list utilizing shared pointers.

For each of these, remember to draw out the algorithm on paper, and trace the process through an exact sequence of steps. Also, it's highly effective to organize methods into sections of scenarios, going from the most specific to the most general.