**Recitation 12**

**Topic:**

Linked lists, taking steps towards a linked list class

**Task:**

The problem is to define a class Sequence that has most of the features we would want for a full linked list class. Instead of the programmer keeping track of where he is currently in the list, the Sequence itself will keep track. So long as the Sequence is not empty, then there will be an implicit "current position". The user cannot directly access the current position, only using it through the methods below. The current position may by "invalid" if the sequence is empty or if the current position has been advanced beyond the last element.

**Copy Control:**

After handling (and testing) the basic features of the Sequence class specified below, add (and test) copy control.

**Basics:**

* **default constructor**
  + Initializes the sequence to be empty.
* **reset**
  + No arguments
  + Resets the current position to the head of the list.
* **add**
  + Argument: a data item to add to the sequence.
  + If the current position is valid, adds the data item right after the current position
  + Otherise adds the data item at the head of the list.
  + The current position is changed to refer to the new element.
* **remove**
  + No arguments
  + Does nothing if the the current position was invalid.
  + Otherwise
    - Removes the item at the current position.
    - The current position is set to the next position after the removed item.
* **data**
  + No arguments
  + Returns the data at the current position.
  + If there is no current postion, then the behavior is "undefined".
* **next**
  + No arguments
  + If the current position is valid, the function advances the current position to refer to the next item.
    - If there is no next item, then the current position becomes invalid.
  + Otherwise, does nothing
* **valid**
  + No arguments
  + Returns true if the current position is valid, false otherwise.
* **clear**
  + No arguments
  + sets the Sequence to have no elements.
* **display**
  + Arguments
    - a separation character to display between elements, defaults to a blank.
    - an output stream, defaults to cout
  + Displays all of the elements of the sequence, using the separator character between each pair. Terminates the line with a newline.

**Analysis**

* Any task that would require having two pointers into the same list would not work with this design.
* As spec'd, in order to add something to the beginning of a non-empty sequence, we would have to first advance the current position until it becomes invalid.
* This class can be implemented either with an underlying singly linked list or a double linked list. A singly linked list is generally less work to use to code the implementation except in the remove method, since we don't have a "prior" pointer.

Sample Test Program and Output

int main() {

Sequence s;

for (int i = 0; i < 6; ++i) s.add(i);

s.display();

cout << "==========\n";

s.reset();

for (int i = 0; i < 3; ++i) s.next();

s.add(42);

s.display();

cout << "==========\n";

s.reset();

for (int i = 0; i < 2; ++i) s.next();

s.remove();

s.display();

cout << "==========\n";

s.clear();

s.display();

cout << "==========\n";

}

0 1 2 3 4 5

==========

0 1 2 3 42 4 5

==========

0 1 3 42 4 5

==========

==========