Ethan Blank

204381410

Sequence (Project 2)

**Sequence:**

A sequence object consists of a private struct of type Node, that contains a data variable and two Node\* pointers that point to Nodes that come before and after the current Node. A sequence also contains two Node\* pointers, one to the first element of the list and one to the last element. A default Sequence is constructed by creating the two *first* and *last* elements and then linking them to each other.

**Psuedocode**

bool insert(int pos, const ItemType& value);

* check whether pos is within the bounds of the Sequence
  + if not return false
  + if so do the following:
    - Create a Node\* and point it to first, loop through the list until you arrive at either last or the position.
    - Create a new node with the given data.
    - Point the new node’s previous and next pointers to the correct Nodes.
    - Increase the size variable.
    - Return true

int remove(const ItemType& value);

* Set a Node\* pointer to the first element
* Set an int i to -1 to represent the index
* Create an erase\_counter, set it 0
* Loop through the Sequence, if the data in the Sequence is equal to value, utilize the erase function at the index “i" to erase it, increment erase\_counter
* Return erase\_counter

int subsequence(const Sequence& seq1, const Sequence& seq2);

* Loop through all the elements of the list and check to see if the values of the first sequence and the second sequence are the same
  + If they are the same, set the return\_value to the index of where the two sequences shared the first same value

void interleave(const Sequence& seq1, const Sequence& seq2, Sequence& result);

* Loop though the lists, obtain values for each position, then insert the data into a new Sequence, alternating the position of insertion and the data from the two lists.

**Test Cases**

Tests the insert function and the default constructor

void test()

{

Sequence s;

assert(s.empty() == 0) // tests the empty function by making sure

that it returns true when the Sequence is empty.

assert(s.insert(0, "lavash")); // tests insertion at the beginning

assert(s.insert(0, "tortilla")); // tests insertion at the beginning, and maintaining “lavash” after tortilla

assert(s.size() == 2); // tests the size function

ItemType x = "injera";

assert(s.get(0, x) && x == "tortilla"); // makes sure that tortilla and lavash were properly implemented

assert(s.get(1, x) && x == "lavash");

}

s.insert(“a”); // test the insert function. List should be abc

s.insert(“c”);

s.insert(“b”);

s.erase(1); // test the erase function. List should be ac

a.insert(“a”);

a.insert(“b”);

a.insert(“c”);

b.insert(“b”);

b.insert(“c”);

int num = subsequence(a, b); // should be 1

a.insert(1);

a.insert(2);

a.insert(3);

b.insert(100);

b.insert(200);

b.insert(300);

Sequence new;

interleave(a, b, new); // new should be 1 100 2 200 3 300

new.swap(b); // new should now be 100 200 300

new.set(0, 20) // new should now be 20 200 300

b.erase(1) // b should now be 1 2 200 3 300

a.remove(3) // a should now be 1 2

int finder = b.find(3) // finder should be 3