Name: Ethan Huang

ID: 406244368

**Notable Obstacles:**

The most notable obstacles I faced were simply just figuring out what the size\_t type was and preventing memory leaks in removeAllBiggerThan. I realized I had to create a temp node to assign the prior currentNode, then delete it.

**Test cases:**

assert(list.before(“notInList”, “A”) == false);

* This tests whether or not the before function works for cases where a parameter isn’t in the list

assert(list.position(“notInList”) == -1);

* This tests whether or not position works when not found

assert(emptyList.get(0, str) == false);

* Tests edge case that list is empty

emptyList.removeAllBiggerThan(tempB);

assert(emptyList.size() == 0);

* Tests removeAllBiggerThan when using a emptyList

for (int k = 0; k < l1.size(); k++)

{

string x;

l1.get(k, x);

cout << x << endl;

}

* This tests whether get works thoroughly through a list

List listWithEqualValues;

listWithEqualValues.addToFront("A");

listWithEqualValues.addToFront("A");

listWithEqualValues.addToFront("A");

listWithEqualValues.removeAllBiggerThan(tempA);

* This tests whether or not it works when the list is all equal values

For every function we tested that takes a string parameter, we could have used a string that isn’t in a list to test if it works for those cases.

My program works for all of the above cases, I can’t think of anything that would cause it to fail.