lab 09: Bubble Sort and Binary Search

Part 1: Extend your implementation from lab07 to include functions for Bubble Sort and Binary Search.

```

#ifndef ARRAY_H
#define ARRAY_H

template <class T>
class Array {
    private:
    /* You fill out the private contents. */

    public:
    /* Do a deep copy of the array into the list.
     * Note: This one uses a pointer!
     */
    Array(const T *array, const int size);
    /* Do a deep copy of the array into the list
     * Note: This one uses a reference to a List!
     */
    Array(const Array<T> &list);

    /* Return the current length of the array */
    int getLength() const;

    /* Returns the index in the array where value is found.
     * Return -1 if value is not present in the array.
     */
    int search(const T &value);

    /* Removes an item at position index by shifting later elements left.
     * Returns true iff 0 <= index < size.
     */
    bool remove(const int index);

    /* Retrieves the element at position pos */
    T& operator[](const int pos);

    /* Returns if the two lists contain the same elements in the
     * same order.
     */
    bool operator==(Array<T> &list) const;

    /* Runs a bubble sort algorithm on the array.
```

```
     * The array shall be ordered from least to greatest
     */
    void bubbleSort();

    /* Searches for an element with value value and returns the index of that
     * data.
     * NOTE: We assume the array is sorted!
     * Return -1 if the value is not found.
     */
    int binarySearch(const T &value);

    /* Free any memory used! */
    ~Array();
};

/* Since Array is templated, we include the .cpp.
 * Templated classes are not implemented until utilized (or explicitly declared).
 */
#include "array.cpp"

#endif
```

Write some test cases:
Create some test cases, using cxxtestgen, that you believe would cover all aspects of your code.

Part 2: Performance
Generate a graph to compare the performance of linear search vs binary search. Your graph should have array size on the x axis and time on the y axis. Make sure to label each graph line! Please turn in as a .pdf!

Auto Grader:
The auto grader is only grading part 1, I will have to assess part 2. In other words, if the auto grade issues a 100, that is only for part 1!

Memory Management:
Now that are using new, we must ensure that there is a corresponding delete to free the memory. Ensure there are no memory leaks in your code! Please run Valgrind on your tests to ensure no memory leaks!

STL:
You may not use anything from the STL.

How to turn in:
Turn in via GitHub. Ensure the file(s) are in your directory and then:

- $ git add <files>
- $ git commit

- $ git push

Due Date: February 13, 2019 2359

Teamwork: No teamwork, your work must be your own.