

Lab 07: Bubble Sort and Binary Search

Instructions: Extend your implementation from Lab04 to include functions for Bubble Sort and Binary Search.

```
1
2 #ifndef ARRAY_H
3 #define ARRAY_H
4
5 template <class T>
6 class Array {
7     private:
8         /* You fill out the private contents. */
9
10    public:
11        /* Do a deep copy of the array into the list.
12         * Note: This one uses a pointer!
13         */
14        Array(const T *array, const int size);
15        /* Do a deep copy of the array into the list
16         * Note: This one uses a reference to a List!
17         */
18        Array(const Array<T> &list);
19
20        /* Return the current length of the array */
21        int getLength() const;
22
23        /* Returns the index in the array where value is found.
24         * Return -1 if value is not present in the array.
25         */
26        int search(const T &value);
27
28        /* Removes an item at position index by shifting later elements left.
29         * Returns true iff 0 <= index < size.
30         */
31        bool remove(const int index);
32
33        /* Retrieves the element at position pos */
34        T& operator[](const int pos);
35
36        /* Returns if the two lists contain the same elements in the
37         * same order.
38         */
39        bool operator==(Array<T> &list) const;
40
41        /* Runs a bubble sort algorithm on the array.
```

```

42     * The array shall be ordered from least to greatest
43     */
44     void bubbleSort();
45
46     /* Searches for an element with value value and returns the index of that
47     * data.
48     * NOTE: We assume the array is sorted!
49     * Return -1 if the value is not found.
50     */
51     int binarySearch(const T &value);
52
53     /* Free any memory used! */
54     ~Array();
55 };
56
57 /* Since Array is templated, we include the .cpp.
58 * Templated classes are not implemented until utilized (or explicitly declared).
59 */
60 #include "array.cpp"
61
62 #endif

```

Write some test cases:

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

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!

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: September 26, 2016 2359

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