lab 07: Templates and Operator Overloading

Instructions: The purpose of this lab is to get you use to implemented templated classes with operator overloading. This powerful, and sometimes troublesome technique, will be used throughout the rest of the course.

You are going to implemented a templated version of your array class:

```
#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 indexOf(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;

```

```
40      /* Free any memory used! */
41      ~Array();
42  };
43
44  /* Since Array is templated, we include the .cpp.
45   * Templated classes are not implemented until utilized (or explicitly declared).
46   */
47  #include "array.cpp"
48
49  #endif
```

Write some test cases:
To facilitate this exercise, I have written some test cases for you. Please extend as needed.

Memory Management:
Use valgrind to ensure there are no memory leaks in your code!

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 06, 2019 2359

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