

lab 07: Virtual Classes and Deep Copying

Instructions: In this lab, utilize a List interface to implement our Array class. List.hpp has been provided:

```
1 #ifndef LIST_HPP
2 #define LIST_HPP
3
4 class List {
5 public:
6     /* Returns the index in the array where value is found.
7      * Return -1 if value is not present in the array.
8      */
9     virtual int indexOf(const int value) = 0;
10
11     /* Removes an item at position index by shifting later elements left.
12      * Returns true iff 0 <= index < size.
13      */
14     virtual bool remove(const int index) = 0;
15
16     /* Insert the integer val at position pos.
17      * Shift all values after pos up ("to the right") by one.
18      * This means the last element will be shifted out of the array
19      * (that is fine.)
20      * If pos is beyond the size of the array, increase the size of the array
21      * so val can be inserted.
22      */
23     virtual void insert(const int pos, const int val) = 0;
24
25     /* Retrieves the element at position pos
26      * Returns -1 if pos is invalid.*/
27     virtual int get(const int pos) const = 0;
28
29     /* Sets the element at position pos to the value val.
30      * Returns -1 if pos < 0.*/
31     virtual int set(const int pos, const int val) = 0;
32
33     /* Returns if the two lists contain the same elements in the
34      * same order.
35      */
36     virtual bool equals(const List &list) = 0;
37 };
38
39 #endif
```

Now provide an IntArray class that implements List and utilizes deep copy constructors. Consider the following code:

```

40  const int ary[] = {10, 50, 34, 20};
41  IntArray *ary1 = new IntArray(ary, 5);
42  IntArray *ary2 = new IntArray(*ary1);
43  std::cout << "(ary1 == ary2)?" << (ary1 == ary2) << "\n";
44  std::cout << "ary1->equals(*ary2)?" << ary1->equals(*ary2) << "\n";
45  ary2.set(2, 10);
46  std::cout << "ary1->equals(*ary2)?" << ary1->equals(*ary2) << "\n";
47  ary2.set(2, 34);
48  std::cout << "ary1->equals(*ary2)?" << ary1->equals(*ary2) << "\n";
49  };

```

Write some test cases:

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

Memory Management:

Now that we 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!

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

Webhook: The webhook is:

<http://coins.csuniv.edu:2234/github/build-csci-315-fall-2017.php>

Remember, after the first push, please wait 5-10 minutes for the auto-grader to get your repository. Then subsequent pushes should receive a grade.

Due Date: September 18, 2017 2359

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