COP 3530
Project 1: Part III

Data Structures & Algorithms
Fall, 2014

Part III: Summary

Due Monday, 9/29

Extend your list implementations to support the subscript operator, throw appropriate exceptions, and develop a comprehensive set of unit tests.

Generate informal documentation

For each class, for each method, define in words, its precise behavior. How should it behave when called with reasonable inputs? What should it do when the inputs aren't reasonable (e.g., attempting to access an element that doesn't exist*)? What should it do when it tries, but fails to allocate memory? [Hint: think exceptions!] Do this before doing anything else for part III.

*—Assuming your list items are indexed from 0 and the list contains 5 items:

- it *would* be an error to call `list.replace( element, 5 )` [there is no element in that position to replace],
- it would *not* be an error to call `list.insert( element, 5 )` [it would be placed immediately following the current last item, and thus is equivalent to a push_back()] , and
- it *would* be an error to call `list.insert( element,`

6) [there is no element at position 5].

List enhancements

Overload the subscript operator

By overloading the subscript operator, we can enable our lists to be indexed like an array. Update your `List<T>` and list classes to support these methods

- **`T& operator[](int i)`** — returns a reference to the indexed element
- **`T const& operator[](int i) const`** — returns an immutable reference to the indexed element

Add error checking

Add error checking to your list implementations. When an error is detected, throw an *appropriate* exception—i.e., the type of exception thrown should reflect what went wrong.

Unit tests

Using the CATCH unit testing framework, develop a comprehensive set of unit tests for your list classes. You should test both the public behavior and for behaviors that are implementation specific—e.g., if you never insert enough stuff into an SDAL to make it grow, or test things that should cause exceptions, &c. then you don't have a good set of unit tests.