CSCI 200 - Fall 2023
Foundational Programming Concepts & Design

Lab 6A - The Abstract List Test Suite

Home | HW Sets | Schedule | Files | Help | Links

This lab is due by Thursday, December 07, 2023, 11:59 PM.
As with all labs you may, and are encouraged, to pair program a solution to this lab. If you choose to pair program a solution, be sure that you individually understand how to generate the correct solution.

Jump To: **Rubric Submission**

At long last, we are now able to create our generic List structure! Throughout the semester we have worked through list operations (insert, remove, get, set) in an abstract manner. How each of those operations and/or algorithms are performed can vary based on how the list is stored in memory. For this lab, we will create the generic List abstraction and then the two concrete implementations.

Download the **Abstract List Starter Package**. This contains a `Makefile`, `main.cpp`, and `test_suite.cpp` to test your `List.hpp`, `Array.hpp`, and `LinkedList.hpp` files.

The Abstract List

The abstract list interface is already created for you:

```
template<typename T>
class IList {
public:
    virtual ~IList() {}

    virtual int size() const = 0;
```

RMP-H

```
    virtual T get(const int POS) const = 0;
    virtual void set(const int POS, const T VALUE) = 0;
    virtual void insert(const int POS, const T VALUE) = 0;
    virtual void remove(const T POS) = 0;
    virtual T min() const = 0;
    virtual T max() const = 0;
    virtual int find(const T VALUE) const = 0;
};
```

Your task is to complete the two concrete implementations for an Array and a LinkedList.

The Concrete Array

The Array class is stubbed out with comments for each operation, implementing the List interface using public inheritance. The Array class will need to implement each method using the array implementations.

The Concrete LinkedList

The LinkedList class is stubbed out with comments for each operation, implementing the List interface using public inheritance. The LinkedList class will need to implement each method using the LinkedList implementations.

Testing

Upon building and running the program, the program will test each concrete list operation against a series of test scenarios. When your implementations pass all the tests and the stress test runtimes match the expected performance, the implementations are complete.

Grading Rubric

Your submission will be graded according to the following rubric:

Points	Requirement Description
0.70	Fully meets specifications
0.15	Submitted correctly by Thursday, December 07, 2023, 11:59 PM

0.15	**Best Practices** and **Style Guide** followed
1.00	**Total Points**

Lab Submission

Always, **always**, **ALWAYS** update the header comments at the top of your `main.cpp` file. And if you ever get stuck, remember that there is LOTS of **help** available.

Zip together your `List.hpp`, `Array.hpp`, `LinkedList.hpp` files and name the zip file `L6A.zip`. Upload this zip file to Canvas under L6A.

This lab is due by Thursday, December 07, 2023, 11:59 PM.
As with all labs you may, and are encouraged, to pair program a solution to this lab. If you choose to pair program a solution, be sure that you individually understand how to generate the correct solution.

Last Updated: 11/02/23 16:00
Any questions, comments, corrections, or request for use please contact jpaone {at} mines {dot} edu.

[Jump to Top] [Site Map]