P2-1: Doubly-Linked List Implementation & Test

Building to a given specification (the Application Programmer Interface, or API) and verifying correctness of your implementation are important concerns in software engineering. APIs establish a sort of contract between the creators and users of a software library about its expected behavior and functionality. The interface also allows developers to separate the concerns of design and implementation of large projects. Early on, design can proceed without distraction from the details of implementation by instead producing a specification of required components and their interfaces. Later, the implementation of each module can proceed more independently from the design of the project as a whole.

Through the end of the course, we will explore this idea by creating and later using a C library based on a pre-specified API. In this project, you will create a doubly-linked list library that will later be used as a component of P2-2. Compliance to the specification is particularly important, so we'll take a look at automated testing as a tool for verifying implementation correctness.

Doubly-Linked List Library

The API for your doubly-linked list library can be found in `doubly_linked_list_API.pdf`. The functions in this module create and manipulate doubly linked lists of nodes (e.g., inserting/deleting nodes, accessing nodes, creating/destroying the list). *This document is the single source of truth for how you implementation should behave! Any code that later uses the library will assume that it behaves according to this API.*

The files `doubly_linked_list.h` and `doubly_linked_list.cpp` provide a shell implementation of this API. A few of these functions are already implemented in `doubly_linked_list.cpp`; you need to complete those that are not. (Note that all the code for this project should be written in C, even though the file extension is “cpp”.)

Automated Testing

Up to this point, your testing of projects has likely been mostly ad hoc and manual, such as trying some different inputs by hand. For this project, we introduce more powerful tools for writing automated tests. By generating a comprehensive test suite that can run automatically, you can be confident that your implementation meets the API specification. Automated tests are also useful during development, since they let you evaluate changes quickly and alert you immediately if anything breaks.

We'll be using a combination of two tools for this part of the project: Google Test Framework, a library for writing and automatically executing tests; and `make`, a build tool to ease compilation. You can find detailed documentation and instructions for our automated test setup in `testbench_documentation.pdf`.

A simple test suite has been provided in `dll_tests.cpp`. This test suite has a few cases that exercise inserting a node at the head of the list, but they are provided mainly as examples of how to use the framework. As you implement your library, you'll need to add many more tests to exercise all the edge cases for each function. This is the same framework we will use to grade your submissions, so it is to your advantage to test thoroughly.

Project Submission

In order for your solution to be properly received and graded, there are a few requirements.

For P2-1: Upload the following files to T-square before the scheduled due date, **5:00pm on Monday, 6 November 2017** (the grace period is documented on T-square):

- `doubly_linked_list.h`
- `doubly_linked_list.cpp`
- `dll_tests.cpp`

P2-2 description is coming soon…

You should design, implement, and test your own code. There are many ways to code this project. Any submitted project containing code (other than the provided framework code) not fully created and debugged by the student constitutes academic misconduct.

Project Grading: The project weighting will be determined as follows:

part	*description*	*due date*	*percent*
	Maze Runner Program		
P2-1	Doubly Linked List	Mon, 6 November 2017	35
P2-2	Baseline features	Mon, 20 November 2017	40
P2-2	Advanced Features		50
P2-2	Demo P2-2 to TA for checkoff	Mon-Fri, 27 Nov - 1 Dec 2017	
	Total		125/100 max