CSCI-1200 Data Structures — Fall 2024
Lab 7 — List Implementation

Checkpoint 1

estimate: 15-20 minutes

Let's practice working with our implementation of the `dslist` class that mimics the STL `list` class. Download these files:

http://www.cs.rpi.edu/academics/courses/fall24/csci1200/labs/07_list_implementation/dslist.h

http://www.cs.rpi.edu/academics/courses/fall24/csci1200/labs/07_list_implementation/lab7_testing.cpp

The implementation of the `dslist` class is incomplete and we have intentionally introduced a couple bugs. First, let's focus on the incomplete `destroy_list` private member function implementation. This function is used by the destructor and the `clear` member function. The provided test cases for Checkpoint 1 in `lab7_testing.cpp` work "fine", so what's the problem?

Before we fix the problem and implement `destroy_list`, let's use Dr. Memory and/or Valgrind to look at the details more carefully. If you don't have a memory debugger on your local machine, you can use the Submitty Memory Debugging gradeable. Study the memory debugger output carefully. The output should match your understanding of the problems caused by the missing `destroy_list` implementation. Ask a TA or mentor if you have any questions.

Now write and debug the `destroy_list` function and then re-run the memory debugger to show that the memory problems have been fixed.

To complete this checkpoint, show a TA the implementation and memory debugger output before and after writing `destroy_list`.

Checkpoint 2

estimate: 20-40 minutes

To get started with this next checkpoint, finish the the implementation of the `push_front`, `pop_front`, and `pop_back` functions. Uncomment the provided tests and write a few additional test cases of these functions to be sure they are working correctly.

For the next part of this checkpoint, we have intentionally added two small bugs impacting the copy constructor and assignment operator for `ds_list`. Can you spot these bugs? Even if you can, leave the bugs in the code for the moment. Write test cases of the copy constructor and assignment operator for `ds_list`. Have you revealed the bugs? Ask for help from a TA or mentor if you are stuck.

Now, you can fix the bugs and re-run your test cases and confirm that the copy constructor and assignment operator will work as expected.

To complete this checkpoint, show a TA the new code and test cases and show them the result of running the test cases before and after fixing the intentionally placed bugs in the copy constructor and assignment operator.

Checkpoint 3 will be available at the start of Wednesday's lab.
You will also have the opportunity to earn an extra credit checkpoint during lab.