CSCI-1200 Data Structures — Fall 2024
Lab 5 — `Vec` Implementation and Big O Notation

In this lab we will continue to explore our implementation from Lecture 6 of the `Vec` class, which mimics the STL `vector` class. Today we will customize the `Vec` class by adding additional member functions. Please download:

http://www.cs.rpi.edu/academics/courses/fall24/csci1200/labs/05_vec_and_big_o_notation/vec_lab5.h
http://www.cs.rpi.edu/academics/courses/fall24/csci1200/labs/05_vec_and_big_o_notation/test_vec.cpp

The first change is that the provided code includes a `print` member function in the `Vec` class to help you test and debug this lab. (Note, this is *not* part of the STL standard for the `vector` class). Run the provided code and study the output. Make sure you understand how the `print` function helps confirm the correct implementation of the copy constructor and assignment operator, and the `clear` and `push_back` member functions.

Checkpoint 1

estimate: 15-30 minutes

Your first task for the lab is to implement a new member function for `Vec` called `erase_at_index`. There is a sample usage in the `checkpoint_tests` function. Uncomment that to test your implementation. After calling your function, the `Vec` will have one fewer item in the container and the remaining items should stay in the same overall order. To do this we will have to slide every item after the specified index one spot to the left. NOTE: We won't decrease the size of the heap allocation for the `m_data` array.

We have provided additional tests in the `corner_cases` function to test the "corner cases" of this new function including, erasing the first item in a `Vec`, erasing the last item in a `Vec`, erasing the *only* item in a `Vec`. Uncomment these tests and make sure your implementation works correctly. For this lab, you don't need to worry about handling *bad input* (e.g., attempting to erase from a negative index, or an out-of-bounds index), but you should always be aware of the assumptions you are making when you write new code. Study these corner cases carefully to make sure you understand them. You will usually be required to write your own test cases to fully exercise and debug your programs.

To verify your code does not contain memory errors or memory leaks, use Valgrind and/or Dr. Memory on your local machine. Students with M1/M2/M3 Mac ARM chips should use the workaround provided on Submitty and upload their code Memory Debugging gradeable.

For a `Vec` that contains n elements, what is the Big O Notation of your implementation of `erase_at_index`? Does it matter which index is passed in? What is the *best case*, *worst case*, and *average case* running time for this function in terms of n?

To complete this checkpoint, show a TA your tested & debugged `erase_at_index` function, and be prepared to discuss the Big O Notation.

Checkpoint 2

estimate: 15-30 minutes

Next, write another new member function for `Vec` called `insert_at_index`. There are two sample usages in the `checkpoint_tests` function. Uncomment that to test your implementation. This function is similar to, but a little more work than, `erase_at_index` because we have to make sure that the `m_data` is large enough to accommodate the new element.

You should write additional tests in the `corner_cases` function to test the "corner cases" of `insert_at_index`: inserting something before the first item in a `Vec`, inserting something after the last item in a `Vec`, inserting an item into an empty `Vec`. Again use a memory debugger to confirm your implementation is fully debugged.

For a `Vec` that contains n elements, what is the Big O Notation of your implementation of `insert_at_index`? Does it matter which index is passed in? Does it matter if `m_size` and `m_alloc` are equal or not equal? What is the *best case*, *worst case*, and *average case* running time for this function in terms of n?

To complete this checkpoint, show a TA your tested & debugged `insert_at_index` function, and be prepared to discuss the Big O Notation.

Checkpoint 3 will be available at the start of Wednesday's lab.