CS 162 LAB #9 – Template Classes

In order to get credit for the lab, you need to be checked off by the end of lab. You can earn a maximum of 3 points for lab work completed outside of lab time, but you must finish the lab before the next lab and get checked off with your lab TAs during their office hours. For extenuating circumstances, contact your lab TAs and the instructor.

This lab is worth 10 points total, with 5 additional extra credits. Here's the breakdown:

- 5 points: Create template class and implement the big 3
- 5 points: Add operator[] and at(), with exceptions
- 5 points (extra credits): add capacity and size, and implement resize() and reserve()

In this lab, you'll start to work with Template Classes in C++.

(5 pts in total) Step 1: Template Classes

Create your own templated vector class and compare it with the `std::vector class`. Start with this `vector.h` file and `try_vector.cpp` file. You may download the files using wget command. Note that the current implementation doesn't have a capacity member!

vector.h:
wget https://classes.engr.oregonstate.edu/eecs/spring2023/cs162-010/labs/vector.h

try_vector.cpp:
wget https://classes.engr.oregonstate.edu/eecs/spring2023/cs162-010/labs/try_vector.cpp

You need to finish **implementing the Big Three, since the vector has dynamic memory**.

Copy Constructor:

```
vector(vector<T> &other){ … }
```

Uncomment line 27-28 in try_vector.cpp to test your CC.

Assignment Operator Overload:

```
vector& operator=(vector<T> &other){ … }
```

Uncomment line 31-37 in try_vector.cpp to test your AOO.

Make sure all of the above works and no memory leaks before moving forward!

(5 pts in total) Step 2: Add Functions and Exceptions

After you have convinced yourself and the TAs that your templated vector class constructors and big three are working, then you can move forward toward implementing the rest of the class. To begin, **implement the following functions:**

```
T operator[](int);  //Only perform address arithmetic
```

Uncomment line 41-46 in try_vector.cpp to test your [ ].

```
T at(int);  //Check to make sure not out of bounds
```

In addition, throw an exception from the `at()` function in the vector template class you created. This function should throw an `out_of_range` exception, when the user tries to access an element outside the bounds of the vector. You need to add the statement below to `at()`.

```
throw std::out_of_range(“out of my vector bounds”);
```

Note: do not copy and paste the above statement directly, the quotes are different!!!

First, run your program with trying to access out of bounds memory using the `at()` function to see what it does now that your function throws an exception, and you are not catching it.

Now, catch the exception so that it doesn’t have a run-time error!!! Remember, you can use the `what()` member function to see your message from the out of range exception.

You will also need to add `using std::exception` or `using std::out_of_range` for these types, since we are not bring in the whole std namespace to control which vector we are using!!!

Uncomment line 50 in try_vector.cpp, and add try, catch statements in main() to test your at().

***Optional: (5 pts extra credit) Step 3: Add capacity and size

Extended Learning: How would this change for having capacity and size members?

What will you have to change in your vector class when you add a capacity private member? The capacity is the actual number of elements allocated on the heap for the vector, and the size is that number that is being used.

How would the constructors and push_back() function change? Write out a plan for the extra constructors you might need for testing this and how the push_back() function changes with regard to having both capacity and size members.

Implement these extra constructors and change your push_back() function to operate correctly with capacity and size, now that they may differ.

Include a resize() and reserve() functionality.

- **resize(): https://cplusplus.com/reference/vector/vector/resize/**

  This function resizes the vector so that it contains **n** elements.

  If n is smaller than the current size, the content is reduced to its first n elements, removing those beyond (and destroying them).

  If n is greater than the current container size, the content is expanded by inserting at the end as many elements as needed to reach a size of n. If val is specified, the new elements are initialized as copies of val, otherwise, they are value-initialized.

If n is also greater than the current vector capacity, an automatic reallocation of the allocated storage space takes place.

- **reserve() : https://cplusplus.com/reference/vector/vector/reserve/**

  This function requests that the vector capacity be at least enough to contain n elements.

  If n is greater than the current vector capacity, the function causes the vector to reallocate its storage increasing its capacity to n (or greater).

  In all other cases, the function call does not cause a reallocation and the vector capacity is not affected.

  This function has no effect on the vector size and cannot alter its elements.

Show your completed work and answers to the TAs for credit. You will not get points if you do not get checked off!

Submit your work to TEACH for our records **(Note: you will not get points if you don't get checked off with a TA!!!)**

1. Create a **tar archive** that contains all files you've created in this lab:
2. Transfer the tar file from the ENGR server to your local laptop.
3. Go to TEACH.
4. In the menu on the right side, go to **Class Tools → Submit Assignment**.
5. Select **CS162 Lab9** from the list of assignments and click "**SUBMIT NOW**"
6. Select your files and click the Submit button.