Lab 8
Each lab will begin with a recap of last lab and a brief demonstration by the TAs for the core concepts examined in this lab. As such, this document will not serve to tell you everything the TAs will in the demo. It is highly encouraged that you ask questions and take notes. In order to get credit for the lab, you need to be checked off by the end of lab. For non-zero labs, 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. For extenuating circumstance, contact your lab TAs and Instructor.

(7 pts) Templated Class and Exceptions

Create your own templated vector class and compare it with the std::vector class. Copy this `vector.hpp` file and `try_vector.cpp` file. This current implementation doesn’t have a capacity!

vector.hpp: [http://classes engr oregonstate edu/eecs/spring2018/cs162-001/labs/vector.hpp](http://classes.engr.oregonstate.edu/eecs/spring2018/cs162-001/labs/vector.hpp)

try_vector.cpp: [http://classes.engr.oregonstate.edu/eecs/spring2018/cs162-001/labs/try_vector.cpp](http://classes.engr.oregonstate.edu/eecs/spring2018/cs162-001/labs/try_vector.cpp)

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

**Copy Constructor:**

```cpp
vector(vector<T> &other) { ... }
```

**Assignment Operator Overload:**

```cpp
void operator=(vector<T> &other) { ... }
```

Make sure all of this works before moving forward!

(3 pts) Adding 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:

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

```cpp
throw std::out_of_range("out of my vector bounds");
```

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!!!

**Remember**, you and your partner will not receive lab credit if you do not get checked off before leaving each lab. Once you have a zero on a lab, then it cannot be changed because we have no way of knowing if you were there or not!!

**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.**