*Exercise 4: algorithms(algorithms.cc)*

***Explained***

* This exercise ended up requiring more brainpower than initially realized.
* The first task was to instantiate a vector named v with 15 positions each filled with the number 0. After the earlier vector syntax exercise, this didn’t prove to be too difficult.
* Next, the “iota” function was introduced. This function takes in a value, then fills each position of the vector with the next increment of that initial value.
* I think the “fill” function was next. This acts very similar to what we did in the initial instantiation of the vector, except this fill function allows you to set your iterator range to be any range within the vector and not just beginning-end.
* Next, we had the reverse function. This acted just as it sounds (though not in the way that I had expected), reversing every value in the array (but doing so by traversing the vector with both beg and end iterators and swapping the values at each iterator along the way). Had to move the “fill” function to the bottom of the test\_algorithm() code as reversing an array of all 0’s would have been difficult to debug.
* The last two functions were sort and sort-compar-lambda, respectively.
* Sort does just like it sounded and simply sorts an array given it is filled with values that can be compared. This one got a little tricky though as, instead of just merely sorting the array, we were asked to sort from position v[1] to v[v.size()-2]. This was “tricky” since the sort function doesn’t accept array values or positions. Instead, this must be done by moving the iterator forward (or backward) and then passing those edited iterators to the sort function.
* Sort-compar-lambda takes the sort function a step further and allows you to enter a custom comparator function as the 3rd parameter. Took me a little before to realzing that this was an example of the functional programming that was shown to us last week.

