CS 2100: Data Structures, Spring 2016
Homework 5

Due *via email* to both me and the grader by midnight on March 3

For this program, you'll modify the Vector.h and testVector.cpp files that are posted on the schedule page; *all* of the problems are designed to be added to that class. Please don't forget to add appropriate comments to the functions and to the main, as well.

1. Finish the function `void erase(int index)`. This is actually the same as our current erase function, expect for one difference. You will rewrite the function erase from our implementation of vectors so that if the number of elements gets below capacity/4, you shrink the array size by half. If this happens, all the elements copied into a new array of the appropriate size.

2. Write the function `void remove(T val)` which takes as input a value of the correct type and removes the earliest occurrence of `val` in the vector. Note that this function does change the size.

3. Write the function `void swap(Vector& other)`, which exchanges the context of the current vector with the contents of another vector of the same type (called `other` in my declaration). After the call to this member function, the elements in this container are those which were in x before the call, and the elements of x are those which were in this. All iterators, references and pointers remain valid for the swapped objects. (Note: for full credit, this should be an $O(1)$ time function!)

4. Write the function `bool operator==(const Vector& other)`, which returns true if the contents of the vector are pairwise identical, and false otherwise. Note that this function ignores the capacity of the vectors, and only compares the elements that are actually in the vector.

5. Finally, write a main function to test all of your functions. Please comment and output appropriately, so that by looking at your code and running your main, we can see exactly where and how you are testing each problem.

6. Extra credit: Write the function `void resize ( int sz, T c )`, which resizes the vector to contain sz elements. If sz is smaller than the current vector size, the content is reduced to its first sz elements, the rest being dropped. If sz is greater than the current vector size, the content is expanded by inserting at the end as many copies of c as needed to reach a size of sz elements. This may cause a reallocation.

   Notice that this function changes the actual content of the vector by inserting or erasing elements from the vector; it does not only change its storage capacity.