Function Templates Lab/Assignment

This is a practice program that gives you experience with function templates, pointers and ranges. The goal is to implement several templated functions that work on *ranges*. You will implement functions that perform operations on ranges such as removing elements within the range, replacing elements within the range, searching for elements, copying one range to another range, etc. Many of these functions mimic how the generic algorithms in the STL work. Several of these were demonstrated in class. You will have a header file that looks like this:

```
#include <iostream> // Make sure this is not in the CS170 namespace!

namespace CS170
{
  /*
   *  Other template function declarations for count, remove, replace, etc.
   *  go here.
   */
  template <typename T> void swap(T &left, T &right);

  #include "Functions.cpp"
}
```

Your .cpp file will contain several functions. The sample driver shows many of them. You will need to figure out what others are required. Many of the functions are going to be very similar in their implementation. However, for this lab, don't be tempted to try and factor out the minimal common code into a separate function, because it will only complicate matters. Once you understand the concept of a range, you will see that the amount of code is not that great. (It never is.) The most complex function is the remove function, so you should work on that one last.

As you implement the functions, you should begin to see a pattern emerging in your code. This should help you understand the purpose of using pointers (a range) with these arrays instead of relying on the size. A range is much more flexible than an array with a size, specifically because it allows you to work on a part of the array (a range) rather than the entire array.

Other criteria

1. With the exception of the *remove* function, all of the functions are trivial, requiring about 4 or 5 lines of code.
2. Do not use the subscript operator anywhere in your code. You are given a range, not an array. Your code will not be accepted if you use the subscript in your program.
3. You are not to use **for** loops anywhere in your code. Use **while** loops. (Many students still don't understand the while loop and I want you to understand them.) Using a **for** loop will cause your program to be rejected.
4. Make sure you know how and when to use **const** in your code.
5. Templated functions can take many different types, even large user-defined types. Use references wherever possible.
6. Do not include any header files in your .cpp file. (You don't need any others.)
7. Notice that you are not creating any classes for this lab, just a bunch of small functions.
8. Your template parameter must be named **T**. If you have two template parameters, they must be named **T1,** and **T2**.
9. You must include the .cpp file at the end of the .h file exactly as shown above.
10. The *remove* function is the only non-trivial function and should be done last. You must understand the algorithm that is posted on the web site in order to implement this. Before attempting to write code for this function, you must write pseudocode. If you can't write the pseudocode (meaning that you don't understand what you are trying to do), then you can't write C++ code. Make sure that your function only makes one pass over the array. Drawing a diagram of what is supposed to happen will help you a lot.

Range Notes

When we are talking about a range of objects in a container (a container can be any kind of container, e.g. array, linked list, vector etc.), we are actually describing a *half-open* range. This means that the left end "points" to the first element in the range and the right end "points" to one *after* the last element to include. (*left* and *right* may not be pointers, but you can think of them that way.) This is also called a *left-inclusive* range and is noted like this:

```
[first, last)
```

So, if you wanted to print the middle 3 integers of this array, this is how you might do it:

```
int array[9] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
print_range(&a[3], &a[6]);  // some function that will print a range of ints
```