Practice with vector-based algorithms

1. *Filtering* is the idea of taking a data structure and creating a copy of it while retaining only elements that fit a certain criteria. For example, we may want to take a vector of integers and create a new vector that contains only the positive integers from the original vector. Sometimes filtering is done on the original structure (modifying it) instead of making a copy; this is called filtering *in place*.

   Write a function to filter a vector of integers, creating a copy that retains only the ones that are greater than a given value.

   ```
   vector<int> filter_greater_than(vector<int> vec, int value)
   ```

2. *Transforming* (sometimes called mapping) is the idea of taking a data structure and creating a copy of it while applying a function to all of the elements. This "function" does not have to be an actual named C++ function; it can be a piece of code that you write on the fly. For example, you may need a function that adds one to each element in a vector of integers. Transforming is often combined with filtering to create, for example, a function that takes a vector of integers, adds one to all of the positive integers, and either eliminates the negative ones or leaves them alone. Transforming, like filtering, can be done *in place* (modifying the original vector rather than making a copy first).

   Write a function that takes a vector of ints and returns a new vector (a copy) with all of the integers doubled; i.e., the vector {1, 2, 3} is transformed into the vector {2, 4, 6}.

   ```
   vector<int> double(vector<int> vec)
   ```

3. Finding the largest and/or smallest items in a data structure (commonly referred to as maximum and minimum, or just max and min) are fundamental ideas you will use a lot. Write two functions, called max and min, that search a vector of integers for the largest and smallest values.

   ```
   int max(vector<int> vec)
   int min(vector<int> vec)
   ```

Combining vectors and pass by value/pass by reference

4. Write a function that takes in a vector and adds 1 to every value in the vector. Write 2 versions of this function.
   a. Function returns a new vector with the updated values
   b. Function alters the vector passed into the function.

   Think about what the return type needs to be for each of these versions and if you should pass in the vector by reference, value or as a constant by reference.

   *Feel free to modify the function headers in 1-3 to be more efficient now that you've learned pass by value/pass by reference.