CSCI-1200 Data Structures — Fall 2017
Lab 5 — STL Vectors vs. Lists: Reversing Data, Insert, & Erase

This lab explores the use of the STL `list` class and the use of list iterators.

Checkpoint 1

Write and test a function named `reverse_vector` that reverses the contents of an STL `vector` of integers. For example, if the contents of the vector are in increasing order before the call to `reverse_vector`, then they will be in decreasing order afterwards. For this checkpoint, use **indexing/subscripting** on the vector, not iterators (or pointers). *You may not use a second vector or array.*

The trick is to step through the vector one location at a time, swapping values between the first half of the vector and the second half. As examples, the value at location 0 and the value at location `size()-1` must be swapped, and the value at location 1 and the value at location `size()-2` must be swapped.

Write a main function to test the function you have written. The main function should (a) create a vector of integers, (b) output the contents, (c) pass the vector to the reverse function, and then (d) output the resulting vector. To help with this, you should write an additional function that prints the size and the contents of a vector (so you don't need to keep writing for loops over and over). Your main function should test special cases of empty vectors and vectors of one or two values. Then you should test "typical" cases. Be sure to also test somewhat bigger vectors with both an even and an odd number of elements.

To complete this checkpoint, show a TA the completed and correct reverse function and the test main function, and then show the TA the compilation and correct output.

Checkpoint 2

Now convert your code from Checkpoint 1 to use lists instead of vectors. Start by replacing '`vector`' with '`list`' everywhere (rename your revers the function to be `reverse_list`). And you'll need to replace your subscripting with iterators.

You may want to use a straightforward concept we did not discuss in lecture: a reverse iterator. A reverse iterator is designed to step through a list from the back to the front. An example will make the main properties clear:

```
list<int> a;
unsigned int i;
for ( i=1; i<10; ++i ) a.push_back( i*i );

list<int>::reverse_iterator ri;
for( ri = a.rbegin(); ri != a.rend(); ++ri )
  cout << *ri << endl;
```

This code will print out the values $81, 64, 49, \ldots, 1$, in order, on separate lines. Observe the type for the reverse iterator, the use of the functions `rbegin` and `rend` to provide iterators that delimit the bounds on the reverse iterator, and the use of the `++` operator to take one step backwards through the list. It is very important to realize that `rbegin` and `end` are NOT the same thing! One of the challenges here will be determining when to stop (when you've reached the halfway point in the list). You may use an integer counter variable to help you do this.

To complete this checkpoint, show a TA your debugged `reverse_list` function and ask any questions you have about regular vs. reverse iterators for lists and vectors.