Introduction

This lab gives you practice in working with vectors, iterators, lists, & maps. It also gives you practice in how to write extra code to test functions that might be used for debugging. There are no files to download for Checkpoints 1 & 2, so you can disable your network until you reach that point.

Checkpoint 1

Below is a simple function that will reverse the contents of a vector of integers. For example, if the contents of the vector are in increasing order before the reverse function is called, then they will be in decreasing order afterwards.

```cpp
void reverse(vector<int> &v) {
    int i, j;
    for (i = 0, j = v.size()-1; i < j; i++, j--) {
        int tmp = v[i];
        v[i] = v[j];
        v[j] = tmp;
    }
}
```

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. For example, 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.

Type this code into your system and test it by writing a separate helper function to print out the size and contents of a vector, and “driver code” in the `main()` function that creates & reverses several vectors (be sure to try vectors of even & odd lengths, an empty vector, a vector of size 1, etc.). For practice, write the vector print function using iterators and do not use `indexing` (vector subscripts).

To complete this checkpoint: Show a TA your test cases and be prepared to discuss your testing strategy.

Checkpoint 2

Now, rewrite the reverse function to use iterators instead of indices to reverse the vector. You may need to use a straightforward concept we did not discuss in lecture: a reverse iterator. A reverse iterator is designed to step through a vector (or list) from the back to the front. The example below which prints out the contents of a vector in reverse order should make the main properties clear:

```cpp
vector<int> a;
unsigned int i;
for (i=1; i<10; ++i)
    a.push_back( i*i );
vector<int>::reverse_iterator ri;
for(ri = a.rbegin(); ri != a.rend(); ++ri)
    cout << *ri << endl;
```

Observe the `type` of 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! `rbegin()` points to the last element in the container and `end()` points to what is just past the last element in the container (not a valid entry).
Test your code (using your test suite from Checkpoint 1) to be sure it works. Then switch all of the vectors in your code to lists. If you have correctly avoided using the subscripting operator, this change should be simple. You may need to adjust some of your code if you used the < operator to compare vector iterators. Remember that for list iterators you may only use the == & != operators.

To complete this checkpoint: Show a TA your debugged program to reverse lists of integers.