Computer Science II — CSci 1200
Lab 4

Introduction

This lab gives you practice in working with vectors, iterators and lists. It also gives you practice in how to write extra code to test functions. Lab 5, next week, will focus more on debugging skills. There are no files to download for this lab, so you have no need of using the network.

Checkpoints

1. Write and test a function that reverses the contents of a vector of integers. For example, if the contents of the vector are in increasing order before the reverse statement, then they will be in decreasing order afterwards. For this checkpoint, use **indexing** on the vector, not iterators. You may not use a second vector.

   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.

   Start by creating a new project and a new file in the new project. Add the function to the file. Then, write a main function to test the reversing function. This main function should create vectors of integers, and for each it should output the contents, pass the vector to the reverse function, and then output the result. To help with this, you should probably write a function that prints the size and the contents of a vector (so you don't need to keep writing for loops over and over). As practice, write this function using iterators. Your main function should test special cases of empty vectors, and vectors of one or two values. Then you should test "typical" cases.

2. Write a new version of the reverse function that uses iterators instead of indices to reverse the vector. Add code to the main program to test this. (It will involve much copying, pasting and slightly modifying the code you wrote for Checkpoint 1 — boring but necessary stuff).

   You may need to use a straight-forward concept we did not discuss in class: a reverse iterator. A reverse iterator is designed to step through a vector (or list) from the back to the front. An example will make the main properties clear:

```
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 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. (They are close for a vector, but they are very different for a list.)

3. In the last checkpoint, write a function that reverses the contents of a **list** of doubles. As before, you may not use a second list. Remember that the list container class does have a `size` member function that gives the number of items stored in the list.

   To practice using a list, you should start by writing, compiling and testing code to create lists (the test lists) and to print the contents of a list (put this in a function). After you are satisfied that these work, proceed to the actual reverse function.

Final note: Writing code using iterators takes some practice. The examples provided here are good ones. Additional examples are (or will be) posted on the course web page under Lecture 7.