Computer Science II — CSci 1200
Lab 5

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 that might be used for debugging. Before starting this lab, please download the examples on the use of iterators and lists posted with Lecture 8 on the course web site. After this, since there are no other files to download for this lab, please turn off your internet connection.

Recall that using iterators (Lecture 8) is a straightforward extension of using pointers: the increment, decrement and dereferencing (`*`) operators are all defined, and comparisons between iterators are allowed and useful. Each standard library container has member functions to initialize iterators which can be used to step through the container and access and change its contents. Vector iterators have `<`, `<=`, `>`, `>=` operators. They may also be subtracted and have integer values added to them and subtracted from them. These properties are exactly the properties of pointers used with arrays, but do not carry over to iterators for other types of containers, including the lists that you will work on in this lab.

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.

   Write a main function to test the function you have written. The main function should (a) create the vectors of integers, (b) it should output the contents, (c) pass the vector to the reverse function, and then (d) output the resulting vector. To help with this, you should probably 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). 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.

   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.

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;
```

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!

To complete this checkpoint, show a TA your version of the code that uses iterators instead of subscripting to access the contents of the vector. One of the challenges here will be determining when to stop (when you've reached the halfway point in the vector). You may use an integer counter variable to help you do this.

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.

   To complete this checkpoint and the entire lab, show a TA the working version of the code that uses lists instead of vectors.