CSCI-1200 Data Structures — Fall 2024
Lab 6 — Iterators & Lists for Reversing Data
STL Vectors vs. STL Lists vs. Homemade Linked Lists

Checkpoint 1: Reverse via Element Swaps

estimate: 15-30 minutes

Download this starter code:
http://www.cs.rpi.edu/academics/courses/fall24/csci1200/labs/06_iterators_lists/checkpoint1.cpp

Study the function `reverse` that uses **indexing/subscripting**/`[]` to step through the vector one location at a time, swapping values between the first half of the vector and the second half. Compile & run this code to confirm you understand the expected behavior.

Your task is to switch this code from STL `vector` to STL `list`. Start by replacing '`vector`' with '`list`' everywhere. Then replace the use of subscript 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:

```
std::list<int> a;
for (int i=1; i<10; ++i) { a.push_back(i*i); }

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

Observe the type for the reverse iterator, the use of the functions `rbegin` (iterator pointing to last item in the list) and `rend` (iterator pointing one spot before the first item in the list) 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.

For this checkpoint you should not use `erase`, or `insert`, or the `push` or `pop` functions.

NOTE: You'll probably need to add the keyword `typename` in front of your templated iterator types to un-confuse the compiler. Don't worry about when this new keyword is required, you won't be tested on this.

```
typename std::list<T>::iterator itr = data.begin();
```

To complete this checkpoint, show a TA your debugged function to reverse an STL list by element swapping. Be sure to ask your TA/mentors any questions you have about regular vs. reverse iterators for lists and vectors.

Checkpoint 2

estimate: 30-40 minutes

Checkpoint 2, a team exercise for a group of 3 or 4 students,
will be available at the start of Wednesday's lab.

https://submitty.cs.rpi.edu/courses/f24/csci1200/course_materials

Checkpoint 3: Reversing a Homemade Linked List *estimate: 30-40 minutes*

This checkpoint is an individual checkpoint. (But you can ask your teammates questions if you get stuck.)

Pull out some paper. Following the conventions from Lecture 9, draw a picture of a "homemade" singly-linked list that stores the values 1, 2, 3, and 4. Make a variable on the stack named `my_list` of type `Node*` that points to the first node in the chain (the node storing the value 1). The 4 node objects should be separate blobs of memory dynamically-allocated on the heap.

Now, *modify this diagram* to reverse the list – you can do this by only changing pointers! You should use the existing node objects. Don't copy the entire diagram or make any new nodes. You should not change the values inside of any node – *don't swap values like we did for Checkpoint 1.*

Then, write pseudo-code to reverse this list, just changing the pointers as you diagrammed above. You may use helper variables (of type `Node*`) but no other data structures or variables. *Remember that when we directly manipulate homemade linked lists we don't use iterators.*

Finally, download this starter code:
http://www.cs.rpi.edu/academics/courses/fall24/csci1200/labs/06_iterators_lists/checkpoint3.cpp

And complete the `reverse` function using your diagram and pseudocode as a guide. Test and debug the code. Add a few additional test cases to the main function to ensure your code works with an empty list, and lists with one or two values. Also add a test or two of a node chain with something other than `int`s.

If you have time, write 2 versions of this function, one version should be iterative (using a `for` or `while` loop) and one version should be recursive.

To complete this checkpoint, show a TA or mentor your diagram and your debugged function(s) to reverse a homemade singly-linked list.