COEN 70L - Formal Specification and Advanced Data Structures

You will start with your sequence class that uses a linked list to store the items.
You will convert this class, as well as the node class, to be template classes.

Purposes:
Ensure that you can convert a container class to a template class. In fact, any time that you write a container class, it is a good idea to start by writing an ordinary container class. After the ordinary class is debugged, convert it to a template class.

Files that you must write:

1. `node2.h`: The header file for the node template class. Start with a copy of the given node2.h header file, and change the documentation at the top to indicate that the class is now a *template* class. Make sure to delete the part of the documentation that refers to the typedef (because you no longer have a typedef!). Now, change the node class, node_iterator class, and const_node_iterator class definitions to template classes. Finally, at the bottom of the header file, you will need the include statement: `#include "node2.template"`
2. `node2.template`: The implementation file for the node template class. Notice that the name of this file ends in "`.tempalte`" rather than "`.cpp`". This is to remind you that template implementation files are never compiled on their own. To implement this file, start with a copy of the given of the ordinary node class.
3. `sequence4.h`: The header file for the new Sequence template class. Actually, you don't have to write much of this file. Just start with a copy of the given header file for the sequence class that uses a linked list. Change the documentation at the top to indicate that the class is now a *template* class. Make sure that you delete the part of the documentation that refers to the typedef (because you no longer have a typedef!). Also, at the bottom of the header file, change the sequence class definition to a template class. Finally, at the bottom of the header file you will need the include statement: `#include "sequence4.template".`
4. `sequence4.template`: The implementation file for the new sequence template class. Notice that the name of this file ends in "`.template`" rather than "`.cpp`". This is to remind you that template implementation files are never compiled on their own. To implement this file, start with a copy of the given implementation of the ordinary sequence class.

Other files that you may find helpful:

1. `sequence_test.cpp`: This is the same interactive test program that you used with the earlier sequences.
2. `sequence_exam4.cpp`: A non-interactive test program that will be used to grade the correctness of your new sequence class.