ETEC2101 Lab5**: Linked Lists** **Points: 100 (*120 possible)***

1. Make a console application (or use your Lab4 as a starting point).
2. The critical stuff – do this first:
    a. (**5 points**) General structure (attributes, template, etc.)
    b. (**5 points**) Nested **node** class (make it protected!)
    c. (**10 points**) **Constructors**: a default one + Destructor (destructor is the interesting one!)
    d. (**5 points**) **[] operator**. .
    e. (**5 points**) **append** and **prepend**
    f. (**10 points**) **Stream operator** – Make the output similar to AL: "[4.1, 5.2, 6.3]"
    g. (**1 point**) **size** getter
    h. (**1 points**) **clear** method
    i. (**10 points**) Doxygen style documentation
3. Medium-level importance – do these second:
    a. (**20 points**) **LinkedListIterator** nested class and all required methods (plus a unique "index" method that tells us what "position" we're at). DO NOT duplicate any data from the linked list itself!
    b. (**5 points**) **begin** and **end** iterator-creator methods
    c. (**10 points**) **find** method. I'd like this to return an iterator (or the end iterator if we can't find the requested element)
4. Nice-but-not-critical – do these if you have time:
    a. (**5 points**) a [deep] **copy-constructor**, **initializer-list constructor**, and [shallow] **move-constructor**
    b. (**5 points**) **=operator**
    c. (**13 points**) **remove** method – Should take a LinkedListIterator and return a LinkedListIterator for the thing *after* the removed element (or end if there is none)
    d. (**5 points**) rbegin and rend reverse-iteration (might require some changes to the iterator)
5. (**10 points**) **Main** program to thoroughly test the above methods using google-tests (you will likely be able to copy-paste what we had for ArrayList since LinkedList's "interfaces" are the same)