Notes About Lab 4
Lab 4 has 3 parts:

A. Create a doubly-linked list: every node has a link to both the *next* node and the *previous* node.
B. Create an *iterator* for the list
C. Run a test program an algorithm both with and without the iterators to see why they are used. This part involves no coding.
Here is a picture of a typical doubly-linked list containing values 1, 3, and 3. Note that we have *sentinel* nodes – empty boxes – at each end.
Most list operations refer to a specific index. To get to the node at index \(n\), we start at the head and do \(n+1\) nexts:

\[
\begin{align*}
\text{Node } p &= \text{head}; \\
\text{for( int } i=0; i <= n. i++ ) \\
&\quad p = p.\text{next};
\end{align*}
\]

The lab directions suggest you make a method

\[
\text{Node getNth(int } n) \quad \text{// returns } p, \text{ not } p.\text{data}
\]
out of this, and use it to implement get(), set(), add() and remove()
If you read the lab directions carefully and draw pictures of your structures as you code, most of Lab 4 is straightforward. The place students tend to find confusing is in the interaction between calls to iterator.next() and iterator.previous().