Exercise – Linked Lists

You are tasked with creating a Double Linked List class, complete with Iterators.

Your *DoubleLinkedList* class will maintain a collection of class *Node*.

Each *Node* class should contain:

- Data
- previous – pointer that references the previous Node
- next – pointer that references the next Node

Your DoubleLinkedList class should also support the following methods:

- pushFront(value) – add a new value to the front of the list
- pushBack(value) – add a new value to the end of the list
- insert(Iterator, value) – add a new value one-past the specified iterator location
- begin() – return an iterator to the first element
- end() – return an iterator to a null element
- first() – return the first element by value, assert if no elements
- last() – return the last element by value, assert if no elements
- count() – return how many elements exist in the list
- erase(iterator) – remove an element by its iterator
- remove(value) – remove all elements with matching value
- popBack() – remove the last element
- popFront() – remove the first element
- empty() – return a Boolean, true if the list is empty, false otherwise
- clear() – remove all elements from the list

Challenges

- Make the *Node* class a **nested class** of *DoubleLinkedList*. This means it will only be accessible to the DoubleLinkedList.
- Make the *Node* and *DoubleLinkedList* classes generic (i.e using templates) to allow the list to store any type of data.
- Implement the following method:
  - remove(predicate) – remove all elements where the predicate returns true
    - You will need to research how predicates work.