Advantage of Sentinel Nodes

→ the slight extra space used greatly simplifies the logic of our operations
→ the header and trailer nodes never change – only the nodes between them do
→ we can treat all insertions and deletions in a unified manner because we are always guaranteed to have elements to left & right

Basic Implementation of Doubly-Linked List

```
class _Node:
  __slots__ = '_element', '_prev', '_next'
  def __init__(self, element, prev, next):
    self._element = element
    self._prev = prev
    self._next = next

class _DoublyLinkedBase:
  def __init__(self):
    self._header = self._Node(None, None, None)
    self._trailer = self._Node(None, None, None)
    self._header._next = self._trailer
    self._trailer._prev = self._header
    self._size = 0

  def _insert_between(self, e, predecessor, successor):
    newest = self._Node(e, predecessor, successor)
    predecessor._next = newest
    successor._prev = newest
    self._size += 1
    return newest
```

```
  def _delete_node(self, node):
    predecessor = node._prev
    successor = node._next
    predecessor._next = successor
    successor._prev = predecessor
    self._size -= 1
    element = node._element
    node._prev = node._next = node._element = None
    return element
```