**Linked Queue Push Operation**

Assume that we have the following lines of code:

```cpp
Queue<int> queue1; // Line 1
queue1.push(5);    // Line 2
queue1.push(8);    // Line 3
queue1.push(3);    // Line 4
```

The following sequence of diagrams shows how the Queue object and its associated dynamic storage changes as these lines are executed.

**Figure 1:** The new, empty Queue object `queue1` created in Line 1 of the code above. The `qFront` and `qBack` pointers are `nullptr`, while `qSize` is 0.

![Figure 1](image1)

**Figure 2a:** The call to `push()` in Line 2 causes a new list Node to be allocated using the temporary pointer `newNode`. The node's data field is initialized to the value passed to `push()`, while its next field is initialized to `nullptr`.

![Figure 2a](image2)

**Figure 2b:** Since the queue is currently empty, the pointer `qFront` is set to point at `newNode`. Then `qBack` is set to point at `newNode` and the `qSize` is incremented to 1.

![Figure 2b](image3)
**Figure 2c:** When the `push()` method ends, the local variable `newNode` ceases to exist.

![Diagram](image)

**Figure 3a:** The call to `push()` in Line 3 causes a new list `Node` to be allocated using the temporary pointer `newNode`. The node's data field is initialized to the value passed to `push()`, while its next field is initialized to `nullptr`.

![Diagram](image)

**Figure 3b:** Since the queue is not empty, the pointer `qBack->next` is set to point at `newNode`. Then `qBack` is set to point at `newNode` and the `qSize` is incremented to 2.

![Diagram](image)

**Figure 3c:** When the `push()` method ends, the local variable `newNode` ceases to exist.

![Diagram](image)
Figure 4a: The call to `push()` in Line 4 causes a new list `Node` to be allocated using the temporary pointer `newNode`. The node's `data` field is initialized to the value passed to `push()`, while its `next` field is initialized to `nullptr`.

Figure 4b: Since the queue is not empty, the pointer `qBack->next` is set to point at `newNode`. Then `qBack` is set to point at `newNode` and the `qSize` is incremented to 3.

Figure 4c: When the `push()` method ends, the local variable `newNode` ceases to exist.

**Linked Queue Pop Operation**

Assume that we then add the following lines of code after the code listed above:

```java
queue1.pop();       // Line 5
queue1.pop();       // Line 6
queue1.pop();       // Line 7
```

The following sequence of diagrams shows how the `Queue` object and its associated dynamic storage changes as these lines are executed.
Figure 5a: The call to `pop()` in Line 5 creates the temporary pointer `delNode` and sets it to the value of `qFront`.

Figure 5b: The pointer `qFront` is set to `qFront->next`. It now points to the 2\textsuperscript{nd} node in the list. `qFront` is not `nullptr`, so `qBack` is not changed.

Figure 5c: The node pointed to by `delNode` is deleted and `qSize` is decremented to 2.
**Figure 5d:** When the `pop()` method ends, the local variable `delNode` ceases to exist.

**Figure 6a:** The call to `pop()` in Line 6 creates the temporary pointer `delNode` and sets it to the value of `qFront`.

**Figure 6b:** The pointer `qFront` is set to `qFront->next`. It now points to the 2\textsuperscript{nd} node in the list. `qFront` is not `nullptr`, so `qBack` is not changed.
Figure 6c: The node pointed to by delNode is deleted and qSize is decremented to 1.

Figure 6d: When the pop() method ends, the local variable delNode ceases to exist.

Figure 7a: The call to pop() in Line 7 creates the temporary pointer delNode and sets it to the value of qFront.

Figure 7b: The pointer qFront is set to qFront->next. It is now nullptr. Since qFront is nullptr, qBack is also set to nullptr.
**Figure 7c:** The node pointed to by delNode is deleted and qSize is decremented to 0.

![Diagram of queue with delNode deleted and qSize set to 0.]

**Figure 7d:** When the `pop()` method ends, the local variable delNode ceases to exist. The queue is now empty.

![Diagram of queue with qSize set to 0 after pop().]