

Below is a quick “code-review” style checklist that you can use to validate the `LinkedList.hpp` you posted.

I’ve split it into *What works*, *Potential pitfalls / bugs*, and *Possible improvements*.

1 What’s correct

| Feature | Why it’s fine |

|-----|-----|

| `RAII` – `clear()` is called from the destructor, so no memory leaks. |

| `Rule-of-Five` – copy/move constructors + assignment operators are all defined and consistent. |

| `Iterator` – forward iterator that satisfies the requirements for range-based loops and STL algorithms. |

| `push_back`, `push_front`, `pop_front` – work correctly on empty/non-empty lists. |

| `insert_after` & `remove` – update `tail` when necessary. |

| `Size tracking` – `size` is updated in every mutating operation. |

| `Namespace isolation` – all symbols are inside `util`. |

| `Header-only` – no separate `.cpp` file needed. |

2 Things that could bite you

| Issue | Explanation | Suggested fix |

|-----|-----|-----|

| `clear()` uses `pop_front()` in a loop | Each `pop_front()` allocates and deallocates one node. In a very large list this is fine, but it’s an extra virtual-dispatch-like

overhead. | Replace with a simple loop that deletes nodes directly (see improvement below). |

| **insert_after** throws on `nullptr` | Good for safety, but the error message might be confusing if you pass `end()`. | Keep as is, but document that calling it with `end()` is undefined/throws. |

| **No `const_iterator`** | The public API only exposes a mutable iterator. For read-only traversal you'd normally provide a `const_iterator`. | Add a second iterator type or make the current one `const` -