**We should NOT implement linked list via smart pointer ?**

**Here is why?**

**Reference :** [**https://stackoverflow.com/questions/36673391/c-linked-list-using-smart-pointers**](https://stackoverflow.com/questions/36673391/c-linked-list-using-smart-pointers)

You do not "need" to use a smart pointer for a linked list, because that statement doesn't make sense. You do **not** use smart pointers for low-level data structures. You use smart pointers for high-level program logic.

As far as low-level data structures are concerned, you use a standard container class from the C++ standard library, like std::list [\*], which solves all your memory-management problems anyway, without using any smart pointers internally.

If you really really need your own highly specialised/optimised custom container class because the entire C++ standard library is unfit for your requirements and you need a **replacement** for std::list, std::vector, std::unordered\_map and other optimised, tested, documented and safe containers – which I very much doubt! –, then you have to manage memory manually anyway, because the point of such a specialised class will almost certainly be the need for techniques like memory pools, copy-on-write or even garbage collection, all of which conflict with a typical smart pointer's rather simplistic deletion logic.

In the words of [Herb Sutter](https://herbsutter.com/elements-of-modern-c-style/):

Never use owning raw pointers and delete, **except in rare cases when implementing your own low-level data structure** (and even then keep that well encapsulated inside a class boundary).

Something along those lines is also expressed in [Herb Sutter's and Bjarne Stroustrup's C++ Core Guidelines](https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md#r3-a-raw-pointer-a-t-is-non-owning):

This problem cannot be solved (at scale) by transforming all owning pointers to unique\_ptrs and shared\_ptrs, partly because **we need/use owning "raw pointers" as well as simple pointers in the implementation of our fundamental resource handles**. For example, common vector implementations have one owning pointer and two non-owning pointers.

Writing a linked-list class in C++ with raw pointers can be a useful academic exercise. Writing a linked-list class in C++ with smart pointers is a pointless academic exercise. Using any of these two self-made things in production code is almost automatically wrong.