Advanced Programming in C++

Exercise – Ordered_List iterators

In this exercise you shall add iterators to a given list class Ordered_List, implemented as a simple, single-linked list.

The iterators

Define two structs named Ordered_List_iterator and Ordered_List_const_iterator, respectively. These structs shall be templates, parametrized on the element type for Ordered_List, i.e. T. The iterator classes shall have the following nested types

```
value_type
reference
pointer
difference_type
iterator_category
```

and the following functionality

- default constructor; which shall initialize an iterator to a past-last value
- a constructor taking a list node pointer and initialize the iterator with that pointer
- copy constructor, move constructor, copy assignment operator, and move assignment operator
- destructor
- **operator*** to dereference an iterator – shall return a reference to the pointed-to object
- **operator->** shall return a pointer to the pointed-to object
- **operator**++ (prefix and postfix) shall move the iterator forward one element
- **operator**== shall return **true** if the compared iterators points to the same list element
- **operator**!= shall return **true** if the compared iterators does not point to the same list element

Ordered_List modifications

Ordered_List shall have the following member types, beside those given,

```
iterator
const_iterator
```

and the following member functions to acquire iterators:

- begin() – shall return an iterator for a non-const Ordered_List, a const_iterator for a const Ordered_List
- cbegin() – shall always return a const_iterator
- end() – shall return a past-end iterator for a non-const List, a past-end const_iterator for a const Ordered_List
- cend() – shall always return a past-end const_iterator

See also instructions in the given files.