1 测试程序的设计思路

在设计测试程序时，我首先确保每个功能点都得到了验证，包括基本的插入和删除操作、迭代器操作（前置递增和后置递增）、范围删除（‘erase‘）、以及链表的清空操作。

我测试了如下功能：

- 我创建了一个 ‘List¡int¿‘ 对象，也测试了初始化列表构造函数。
- 测试了使用 ‘push_back()‘ 和 ‘push_front()‘ 函数插入多个元素。
- 测试了 ‘front()‘ 和 ‘back()‘ 函数，以验证它们是否正确返回链表的第一个和最后一个元素。
- 测试了 pop_back()和pop_front()删除头尾元素
- 使用前置递增 ‘++iterator‘ 和后置递增 ‘iterator++‘ 分别遍历链表，确保迭代器的行为正确。
- 测试了erase单个元素和erase(iterator from, iterator to) 删除指定范围内的元素，检查链表在删除后的状态。
- 测试了insert函数。
- 对 ‘clear()‘ 函数进行测试，清空链表，并验证链表的大小和状态。
- 测试了拷贝构造和移动构造，验证链表在深拷贝和资源转移后的行为是否正确。
- 测试了赋值运算符和移动赋值运算符
- 测试了begin()和end()
- 测试了empty()和size()

我的测试充分覆盖了链表的所有主要操作，包括插入、删除、遍历、赋值、拷贝和移动语义等。

2 测试的结果

测试的终端输出结果如下：

```
测试: 默认构造函数成功
测试push_back和push_front: 插入 5, 10, 15, 20, 25 到链表
5 10 15 20 25
测试: front() 返回 5
测试: back() 返回 25
测试pop_back()和pop_front(): 删除头尾元素，此时链表应剩余 10 15 20
10 15 20
测试前置递增 ++iterator
当前元素为: 15
测试后置递增 iterator++
当前元素为: 10
测试初始化列表构造函数
1 2 3 4 5
测试拷贝构造函数
10 15 20
```

测试移动构造函数
10 15 20
测试insert: 在链表头部插入 100, 尾部插入 200
100 10 15 20 200
测试erase: 删除头部元素
10 15 20 200
测试erase(from, to)
10 200
测试赋值运算符
10 200
测试移动赋值运算符
10 200
测试clear() 函数清空链表
测试empty()和size()成功
测试空链表中begin()和end()成功

从测试结果中可以看到：

- ‘push_back()‘ 和 ‘push_front()‘ 成功将元素插入到链表中，输出结果与预期一致。
- ‘front()‘ 函数返回了链表的第一个元素，‘back()‘ 返回了最后一个元素，测试成功。
- ‘pop_back()‘ 和 ‘pop_front()‘ 成功删除了链表的头部和尾部元素，链表中的剩余元素符合预期。
- 前置递增 ‘++iterator‘ 和后置递增 ‘iterator++‘ 操作成功，能够正确遍历链表并访问元素。
- 初始化列表构造函数工作正常，链表元素按照给定顺序插入。
- 拷贝构造和移动构造测试通过，链表的深拷贝和资源转移行为正常。
- ‘insert()‘ 函数成功在链表的头部和尾部插入元素，‘erase()‘ 函数能够删除指定的范围。
- ‘clear()‘ 函数能够正确清空链表，且调用 ‘empty()‘ 和 ‘size()‘ 结果符合预期。
- 最终验证了空链表下的迭代器行为，‘begin()‘ 和 ‘end()‘ 的结果正确。

此外，我还使用了 ‘valgrind‘ 对程序进行内存泄漏测试，结果显示程序在整个运行过程中没有发生内存泄漏，说明链表的所有动态分配的内存都得到了正确的管理和释放。

3 （可选）bug报告

测试过程中没有发现功能上的问题或 bug，所有功能都符合预期。