作業：合併兩個已排序的Linked List

敘述：

給定兩個已排序的連結列表 **list1** 和 **list2**，要求寫一個函數 **Linked List**，將它們合併成一個新的連結列表，使得合併後的列表仍然按照升序排序。下面以 **mergeSortedLists(list1, list2)**為函式呼叫完成。可以不以此命名。

1. 空列表
2. 一方為空
3. 不同長度的列表
4. 不同型別的列表
5. 大型數據集
6. 單一節點的列表
7. 重複值的列表
8. 大量重複值的列表
9. 節點值相同的情況
10. 大型數據集的反向排序

測資**1 -** 空列表：

```
Node<int>* list1 = nullptr;

Node<int>* list2 = nullptr;

Node<int>* mergedList = mergeSortedLists(list1, list2);

// 預期 mergedList 為 nullptr
```

測資**2 -** 一方為空：

```
Node<int>* list1 = new Node<int>(1);

list1->next = new Node<int>(3);

list1->next->next = new Node<int>(5);

Node<int>* list2 = nullptr;
```

```
Node<int>* mergedList = mergeSortedLists(list1, list2);
// 預期 mergedList 為 1 -> 3 -> 5
```

測資3 - 不同長度的列表：

```
Node<int>* list1 = new Node<int>(1);
list1->next = new Node<int>(3);
list1->next->next = new Node<int>(5);
Node<int>* list2 = new Node<int>(2);
list2->next = new Node<int>(4);
Node<int>* mergedList = mergeSortedLists(list1, list2);
// 預期 mergedList 為 1 -> 2 -> 3 -> 4 -> 5
```

測資4 - 不同型別的列表：

```
Node<char>* list1 = new Node<char>('a');
list1->next = new Node<char>('c');
list1->next->next = new Node<char>('e');
Node<char>* list2 = new Node<char>('b');
list2->next = new Node<char>('d');
Node<char>* mergedList = mergeSortedLists(list1, list2);
// 預期 mergedList 為 a -> b -> c -> d -> e
```

測資5 - 大型數據集：

```
// 創建兩個大型的已排序連結列表
Node<int>* list1 = nullptr;
Node<int>* list2 = nullptr;
for (int i = 0; i < 1000000; ++i) {
    insert(list1, i * 2);
    insert(list2, i * 2 + 1);
}
Node<int>* mergedList = mergeSortedLists(list1, list2);
// 預期 mergedList 包含了 0 到 1999999 的所有奇數和偶數
```

測資**6 -** 單一節點的列表：

```
Node<int>* list1 = new Node<int>(1);
Node<int>* list2 = new Node<int>(2);
Node<int>* mergedList = mergeSortedLists(list1, list2);
// 預期 mergedList 為 1 -> 2
```

測資**7 -** 重複值的列表：

```
Node<int>* list1 = new Node<int>(1);
list1->next = new Node<int>(3);
list1->next->next = new Node<int>(5);
Node<int>* list2 = new Node<int>(3);
list2->next = new Node<int>(4);
```

```
Node<int>* mergedList = mergeSortedLists(list1, list2);
// 預期 mergedList 為 1 -> 3 -> 3 -> 4 -> 5
```

測資**8 -** 大量重複值的列表：

```
Node<int>* list1 = new Node<int>(1);
list1->next = new Node<int>(1);
list1->next->next = new Node<int>(1);
Node<int>* list2 = new Node<int>(1);
list2->next = new Node<int>(1);
list2->next->next = new Node<int>(1);
Node<int>* mergedList = mergeSortedLists(list1, list2);
// 預期 mergedList 為 1 -> 1 -> 1 -> 1 -> 1 -> 1
```

測資**9 -** 節點值相同的情況：

```
Node<int>* list1 = new Node<int>(1);
list1->next = new Node<int>(2);
list1->next->next = new Node<int>(3);
Node<int>* list2 = new Node<int>(1);
list2->next = new Node<int>(2);
list2->next->next = new Node<int>(3);
Node<int>* mergedList = mergeSortedLists(list1, list2);
```

```
// 預期 mergedList 為 1 -> 1 -> 2 -> 2 -> 3 -> 3
```

測資**10 -** 大型數據集的反向排序:

```
// 創建兩個大型的已排序連結列表（反向排序）
Node<int>* list1 = nullptr;
Node<int>* list2 = nullptr;
for (int i = 1000000; i > 0; --i) {
    insert(list1, i * 2);
    insert(list2, i * 2 - 1);
}
Node<int>* mergedList = mergeSortedLists(list1, list2);
// 預期 mergedList 包含了 0 到 1999999 的所有奇數和偶數
```