题目

编写一个单向链表类 List，其中的每个节点 Node 符合以下结构：

```
struct Node {
    int value;
    Node* next;
    Node(int v) : value(v), next(NULL) {}
};
```

并为这个链表类实现以下的操作：

1. List::List(int* arr, int array_size);
   使用一个数组来初始化链表的构造函数。

2. int List::length() const;
   返回链表当前的长度。

3. bool List::insert(int pos, int value);
   向链表中插入一个值为 value 的节点，使得插入后该节点的位置为 pos（节点的位置定义为从链表头开始往后数的位移，链表头为 0，下一个为 1，等等）。pos 的范围在[0, length)之间才能插入成功（返回 true），否则插入失败（返回 false）。

4. int List::find(int pos, int value); 在链表中查询从 pos 位置开始(包含 pos)，第一个值为 value 的节点的位置，若这样的节点不存在则返回-1。

5. int List::remove(int value); 删去链表中所有值为 value 的节点，返回删除掉的节点的个数。若这样的节点不存在则返回-1。

6. bool List::split(int pos, List& new_list); 将链表在 pos 位置之后(不包含 pos)的链接处切断，分裂出另一个链表，并将这个分裂出来的链表对象 写入到 new_list 参数中。pos 的范围在[0, length)之间才能分裂成功（返回 true），否则分裂失败（返回 false）。
   需要判断 new_list 是否长度为 0。 长度不为 0，返回 false；长度为 0，继续后续执行。

7. void List::combine(List& append_list); 将 append_list 合并到当前链表的末尾(即当前链表的末尾链接上 append_list 的链表头)，两个链表对象被合并为一个链表对象。
   需要把 append_list 置为空链表。

8. void List::print(); 从头到尾打印出当前链表中的所有节点的值。
   打印格式：一行，每个值之后加上一个空格，行尾需打印换行符。使用标准输出。
   即使该链表没有元素，也需要打印一个换行符。

提示

你的 List 类可以含有以下两个成员变量：
Node* head; // Head of the list.
int size; // The current size(length) of the list.

注意

请保证你用 new 操作符申请的所有内存最终都被 delete 释放掉（且每处内存只释 放一次），避免内存泄漏。
因此你还需要为 List 类编写析构函数，完成动态申请内存的释放工作。

关于提交

在本次作业中，请将 List 类的定义和实现均写入一个头文件中，命名为"学号_姓名.h"， 并只上传提交这一个文件。题目中提供的 Node 的定义也一并放入此文件中。

关于测试：

1. 可自行向你的项目中添加 main.cpp，在 main.cpp 中 include 你编写的头文件，并 在 main 函数中自行编写一些调用 List 类的测试代码来测试你所编写 List 类的正确性。但测试使用的 main.cpp 无需提交。
2. 作业批改将使用自动评测程序，测试方法与 1.中所述完全一致，但使用不同的测试代码。请保证你的代码中类名、函数名、函数原型、输出格式以及提交的文件名等完全符合以上的 描述，否则将导致失分。

关于评分

正确性（90 分）：评测共有 8+1（delete） 个测试点，每个测试点 10 分。
代码风格（10 分）：你的程序应具有良好的代码风格。 总分 100 分。

样例

如图是一个 main.cpp 的测试样例和输出结果

```cpp
int main(){
    int arr[6] = {1,2,3,4,5,6};

    List list(arr,6);
    cout << list.length() << endl;
    list.print();
    cout << "----------------------" << endl;
    cout << list.insert(4,8) << endl;
    cout << list.insert(2,8) << endl;
    cout << list.length() << endl;
    list.print();
    cout << "----------------------" << endl;
    cout << list.find(2,8) << endl;
    cout << list.find(3,8) << endl;
    cout << list.find(6,8) << endl;
    cout << "----------------------" << endl;
    cout << list.remove(8) << endl;
    cout << list.length() << endl;
    list.print();
    cout << "----------------------" << endl;
    List list2;
    cout << list.split(1,list2) << endl;
    cout << list.length() << endl;
    list.print();
    cout << list2.length() << endl;
    list2.print();
    cout << "----------------------" << endl;
    list.combine(list2);
    cout << list2.length() << endl;
    list2.print();
    cout << list.length() << endl;
    list.print();
}
```

```
6
1 2 3 4 5 6
----------------------
1
1
8
1 2 8 3 4 8 5 6
----------------------
2
5
-1
----------------------
2
6
1 2 3 4 5 6
----------------------
1
2
1 2
4
3 4 5 6
----------------------
0

6
1 2 3 4 5 6
```