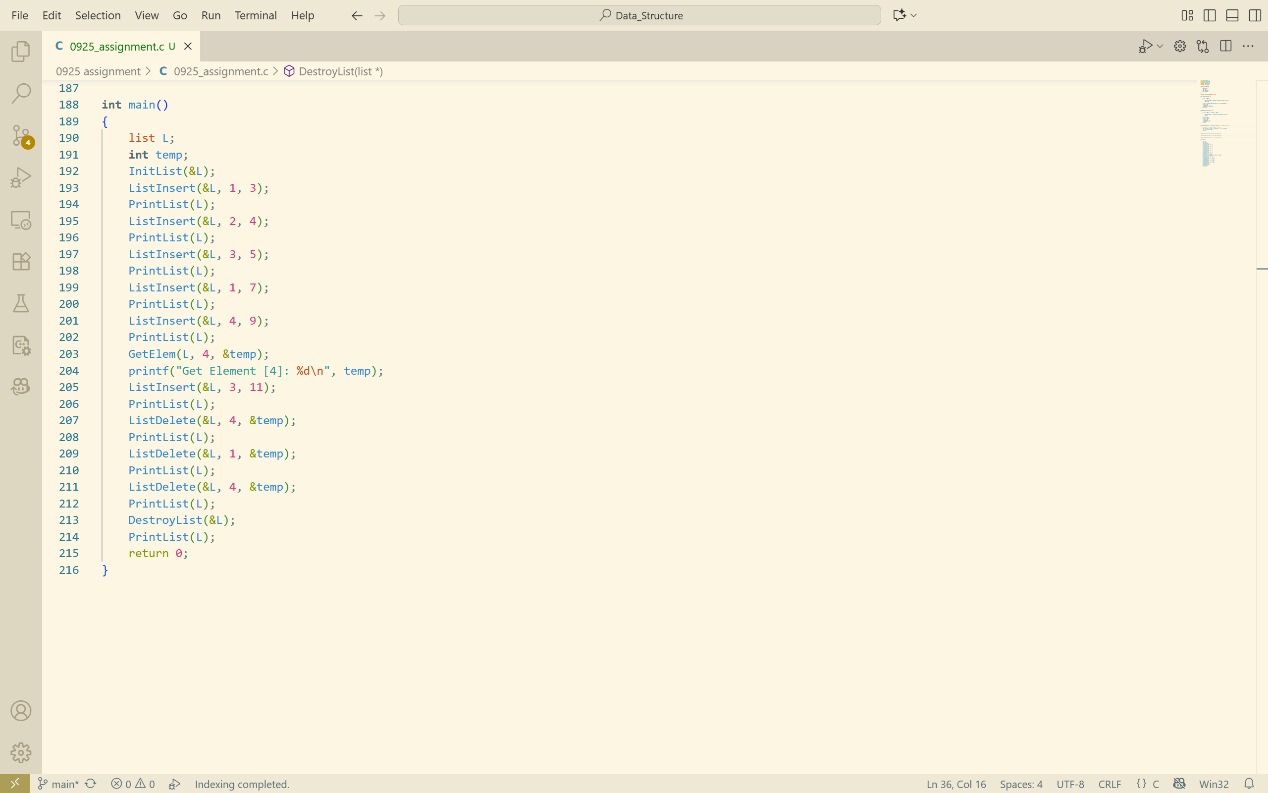
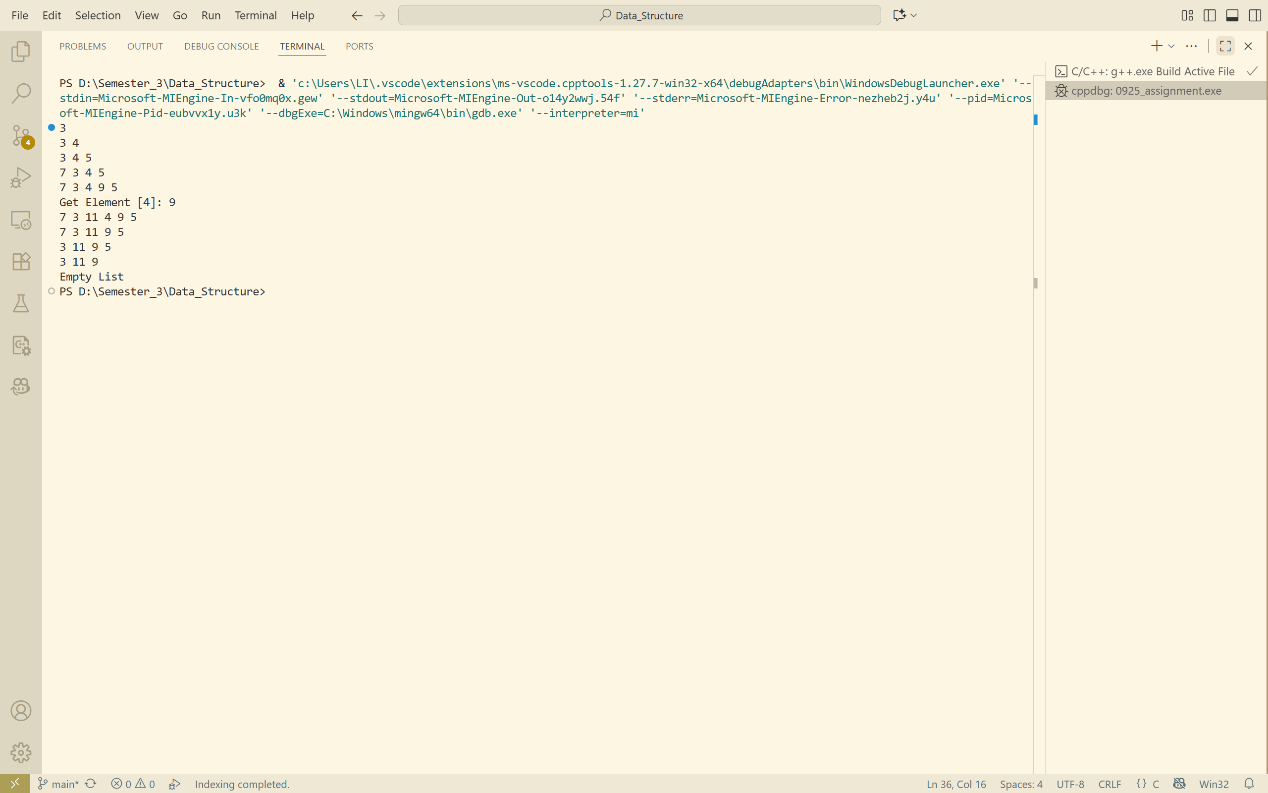
数据结构第3次作业（9月25日）

1. 源程序

源程序位于本目录下，其文件名为0925\_assignment.c

1. 运行截图





这两幅图片分别为main函数截图和运行结果截图。我在初始化list之后在其1、2、3位置分别插入了3、4、5，因此第3行输出为3 4 5。接下来，我在第1位插入了7，因此第4行输出为7 3 4 5。下一步，我在第4位插入了9，因此输出为7 3 4 9 5。然后，我使用GetElem函数读取第4位的数值，其结果显示为9，是正确的。接下来，我在第3位插入11，所以输出为7 3 11 4 9 5。下一步，我使用ListDelete函数删除了第4位的值，因此输出为7 3 11 9 5。接着，我删除了第1位的值，输出为3 11 9 5。我又尝试删除了第4位（末尾）的值，输出为3 11 9。最后，我使用DestroyList销毁了这个线性表，输出显示表为空。

1. 讨论

在本次实验中，插入、删除的最坏时间复杂度还是，虽然它在处理表头、表尾的表现很优秀，但我为此编写了200行左右的代码，因此我觉得十分不值得。具体原因如下：

1. 编写时间过长。我耗费了大概3天时间才写完这个程序，这并不是因为我的技术不熟练，而是给定的数据结构过于复杂，需要大量的时间来思索“要使用什么数据结构”、“怎么样使用这个数据结构”。
2. 编写过程易出错。本次实验中使用了许多指针，这些指针在我进行了多次调试之后才勉强不会出现Segmentation Fault等错误。虽然我在代码中很详细地考虑到了各种异常情况，但如果我们不选用这样的数据结构，这些错误本可以被避免。
3. 很少有这样的需求驱使我们这样编写程序。计算机学科是一门从实际出发的学科，因此，我们做的事情就是发现问题并以最简单的方式解决这个问题。在现实生活中，很少有人使用顺序储存来实现逻辑上的首尾相连结构，因为这增大了程序员的工作量，还会引发潜在的风险。
4. 进一步优化的设想

正如我在讨论部分所写，这种数据结构有诸多问题，因此我们需要一种更优化的结构来实现。如果我们需要实现线性表，单链表是一种不错的选择，在我们已经定位到要插入、删除的元素后，操作的时间复杂度为。如果要实现逻辑上首尾相接，可以使用环形链表。

当然，如果我们使用链表，插入删除的复杂度确实减少了，但无论要查找哪个元素，我们都必须从头遍历。尽管如此，这依旧是一种十分高效的结构，但它已经不能被称为“顺序储存”了，而是链式储存。