|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《数据结构》实验报告** |
|  |
| 实验一  线性结构及其应用  学 院: 计算机科学与技术   |  |  | | --- | --- | | 姓 名: |  | | 学 号: |  | | 专 业: | 计算机科学与技术 | | 日 期: | 2020-04-05 | |

# 一、问题分析

题目要求收集两个班的学生成绩信息并按降序排列，其中每个学生的数据包括班级，学号和成绩信息，需要用一个结构体作为整体存储。并且数据是可以不断添加进来的，可以用两个线性表分别来存储两个班级的学生信息。但是如果用数组等顺序存储的方式来存储数据，每次存储一位学生的信息都要进行大量移动操作，并且学生数量未知，用顺序存储可能造成资源的浪费。这时利用链表的插入和删除操作时间复杂度仅为O（1）的良好特性，并且可以根据实际情况动态扩展内存，可以选择线性链表为存储结构。合并两个班级的学生信息，可以用链表的归并算法，因为两个链表都是有序的，归并的时间复杂度为O（n）。

# 二、详细设计

## 2.1 设计思想

1、选择线性链表做为存储结构。

每一个节点包含学生的三个信息信息和一个指向下一个节点的指针，节点与节点之间通过降序连接起来。每次要插入一个元素，先动态申请内存，遍历链表找到第一个不比它成绩值大的元素，在它前面插入该元素（要考虑插入位置为链表头和尾的特殊情况）。这样就可以按降序建立一个班级的链表，不用每次增加元素都要进行排序，十分方便。两个班级的链表分别用两个头指针管理，可以构造一个指针数组来存储这两个头指针，这样在不只是2个班级的情况下只需增加数组的大小即可。

2、链表的遍历

遍历时先判断是否为空表，如果不为空则用一个新指针指向第一个节点，通过节点与节点之间的联系便可以遍历链表。遍历链表时可以进行匹配学号操作，实现按照学号查询成绩和删除学生信息。也可以打印全部学生信息。

3、链表的合并和反转

运用归并算法进行两个有序链表的合并。分别用两个指针指向两个链表的头结点，大的就添加为一个新链表的节点，循环比较。这样就可以得到一个合并了两个班级信息的新链表。反转则利用两个指针，第一个指向合并后的新链表节点，第二个指向一个逆序建立链表的反转节点，先复制原合并链表指针指向节点的数据，让他指向反转指针指向的节点，再让反转指针指向这个新节点。循环后便可以O(n)的时间复杂度构建一个反转链表。

## 2.2 存储结构及操作

(1) 存储结构（一般为自定义的数据类型，比如单链表，栈等。）

链表节点结构体StudentLinkedListNode

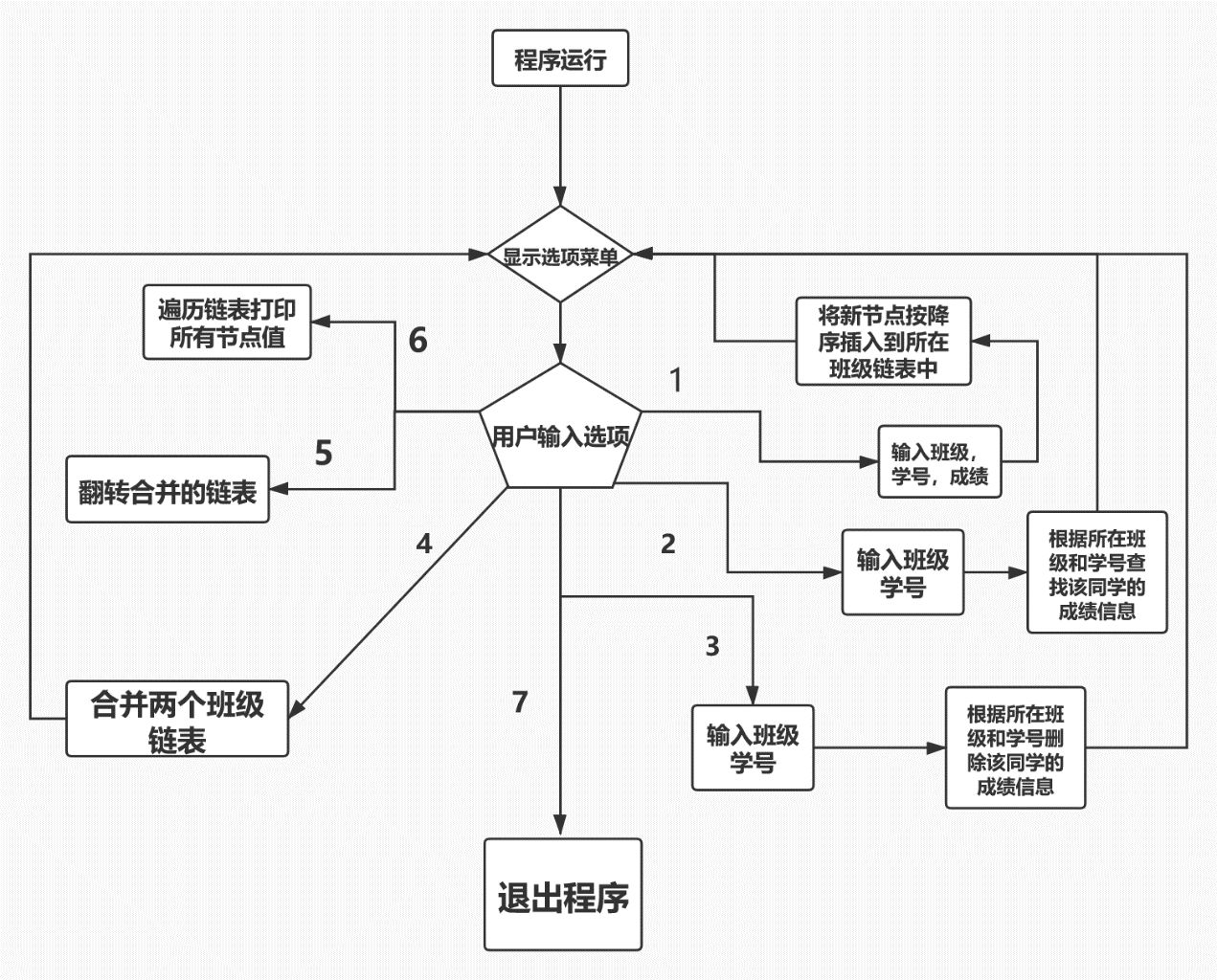
内容包括ClassID（班级）,StuID（学号）,Grade(成绩，以上均为int类型)，指向结构体StudentLinkedListNode的指针next。

(2) 涉及的操作（一般为自定义函数，可不写过程，但要注明该函数的含义。）

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名 | 参数 | 功能 | 返回值 |
| printLinkedListNode | 链表节点指针 | 打印一个节点学生的各项数据 | 无 |
| outputStudentLinkedList | 链表头指针 | 打印这个头指针指向链表的所有节点值（三项信息） | 无 |
| studentLinkedListCreate | 学生班级，学号和成绩 | 根据参数新建一个节点 | 指向新节点的指针 |
| studentLinkedListCopy | 链表节点指针 | 根据节点指针的内容值复制出一个新节点 | 指向新节点的指针 |
| studentLinkedListAdd | 链表头指针，要插入的链表节点指针 | 按照降序将该节点插入到合适位置 | 链表头指针 |
| searchByID | 链表头指针，要查找的学号 | 遍历链表查找是否存在有该学号对应的学生信息，有就输出该学生信息 | 无 |
| deleteByID | 链表头指针，要删除的学号 | 遍历链表查找是否存在有该学号对应的学生信息，有就删除该节点 | 链表头指针 |
| mergeLinkedList | 链表头指针数组 | 合并两个班级的成绩情况 | 合并后的链表头指针 |
| reverseLinkedList | 链表头指针 | 反转这个链表 | 反转后的链表头指针 |
| isExistStudentId | 链表头指针，要查重的学号 | 遍历链表确认是否有重复的学号 | 布尔值，存在返回真，不存在返回假 |

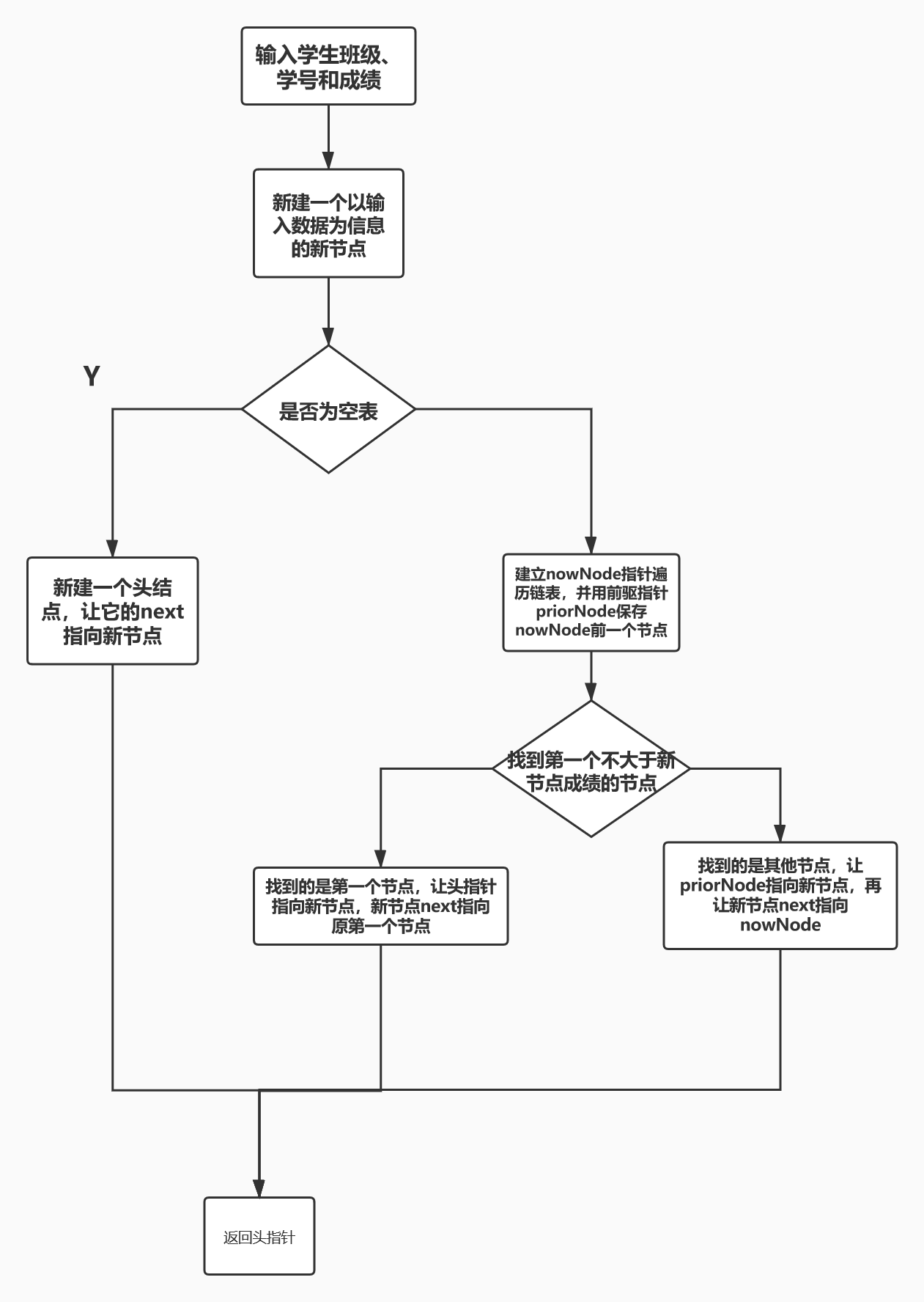
## 2.3 程序整体流程

整体流程：



核心算法流程：

（1）插入节点studentLinkedListAdd：

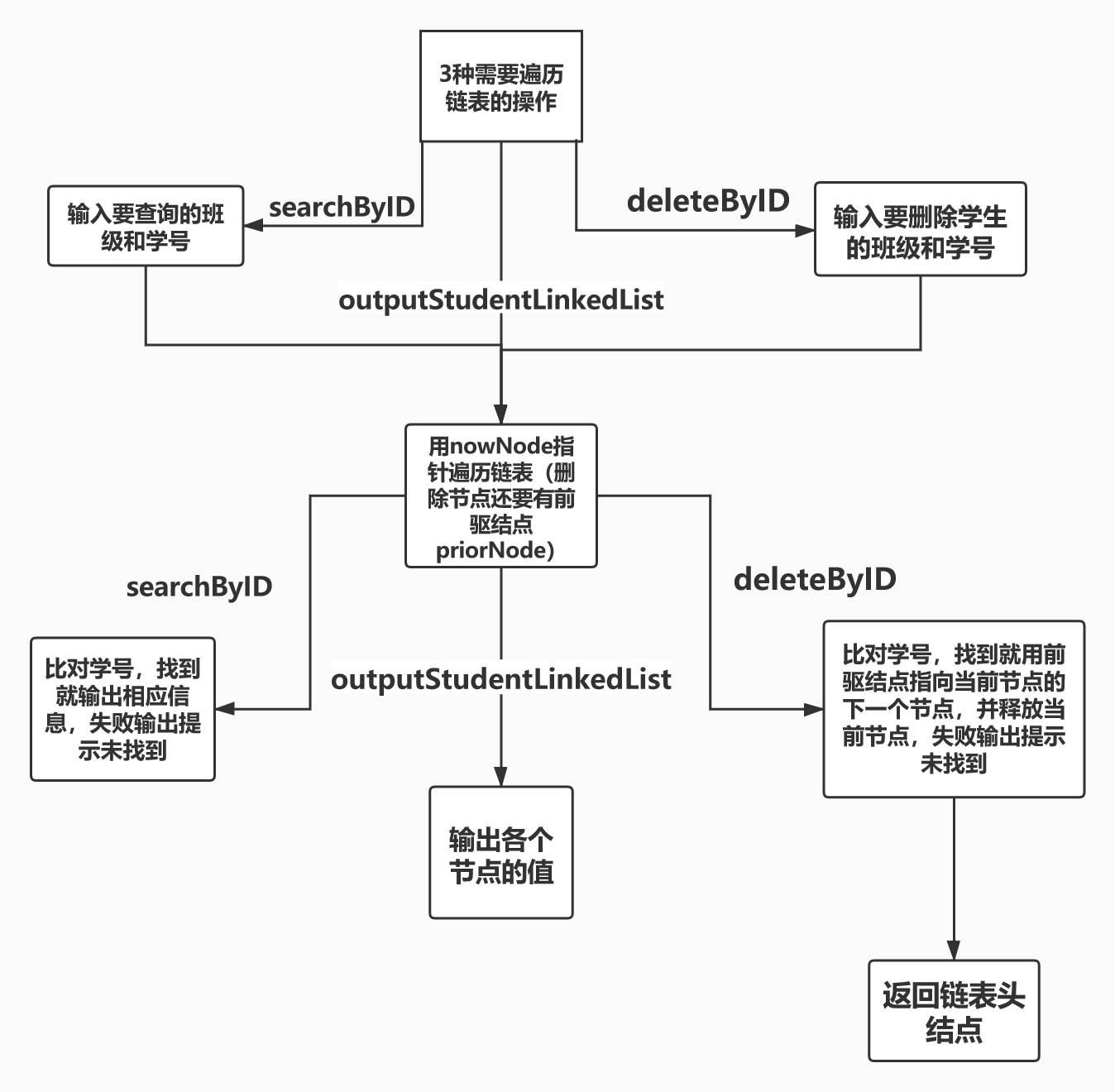


（2）遍历链表（查找，删除，查找）

searchByID

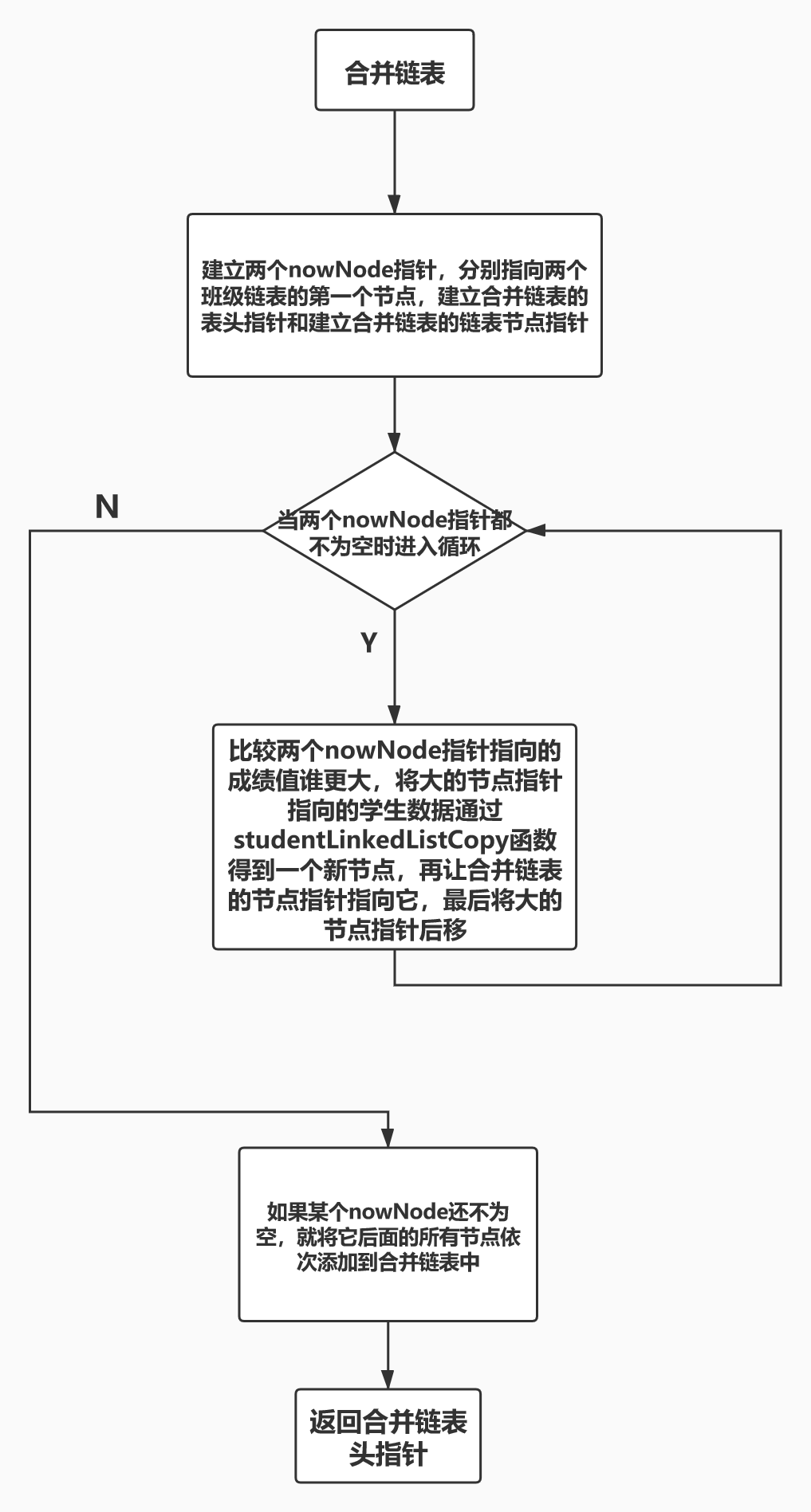
deleteByID

outputStudentLinkedList：



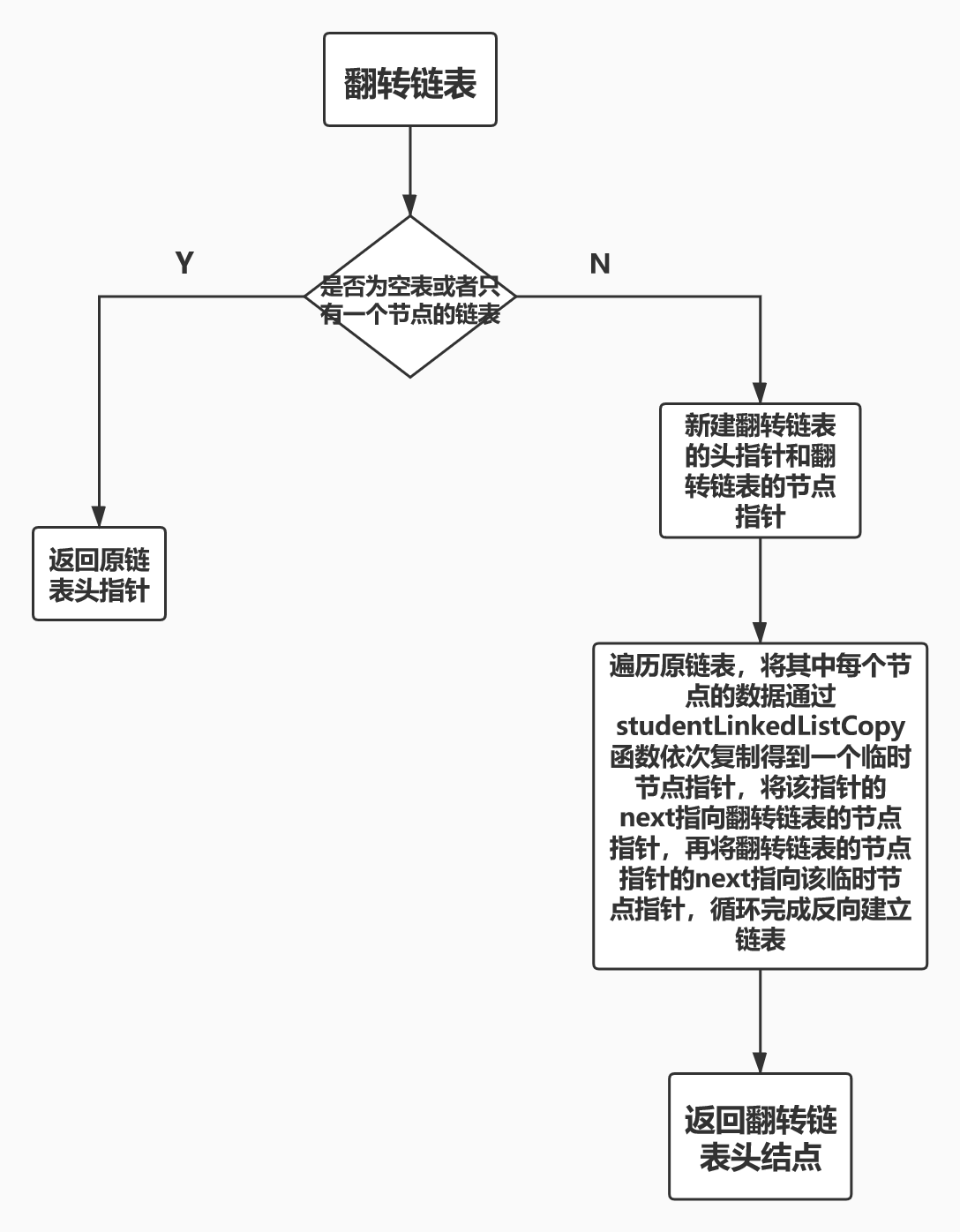
（3）合并链表

mergeLinkedList



（4）翻转链表

reverseLinkedList



# 三、用户手册

如：(1)输入数据的方式；(2)实现各种功能的操作方式等。

进入程序，首先会出现一个选择操作菜单。

*1.insert 2.search by StuID 3.delete by StuID 4.merge 5.reverse 6.output 7.exit*

输入上述之一的数字后回车即可进入相应的操作功能，退出程序请按7。

输入1将插入数据，程序会提示

*How many rows of data do you need to input?*

输入你本次要插入的学生信息总行数，如果输入行数小于1，将会提示输入的行数不能小于1，要求重新输入直到输入的行数大于0，会显示

*input the %d row data format as:class\_id,student\_id,grade*

请按顺序输入学生班级，学号和成绩，到遇到输入班级不是0或1、学号小于0和成绩小于0的异常输入，将会提示您重新输入，直到正确输入完要求输入的行数的数据后再次显示选择菜单。

特色：输入已存在于链表的学号（同一班级）时，会提示

*学号为XXX的学生信息已存在，请重新输入：*

验证数据：

输入

1,190110424,99

1,190110424,98

输出

*学号为190110424的学生信息已存在，请重新输入：*

输入2，进入按学号查找学生信息的功能，程序会提示

*input the data format as: ClassID, StuID*

按照指示依次输入班级和学号，如果输入的班级不是1或2，或者输入学号小于0，将直接输出未找到，否则，将进入所在班级表中进行查找，找到则输出该学生的信息，否则输出未找到。

输入3，进入按学号删除学生信息的功能，程序会提示

*input the data format as: ClassID, StuID*

按照指示依次输入班级和学号，如果输入的班级不是0或1，或者输入学号小于0，将直接输出未找到，否则，将进入所在班级表中进行查找，找到则删除该学生的信息，否则输出未找到。

输入4，将进入合并两个班级成绩的功能，无需输入，程序会按降序输出两个班级合并后学生成绩情况 ，其中既包含0班同学也包含1班同学，同时降序性不变。

输入5，将进入翻转合并两个班级的链表的功能，无需输入，如果之前未执行操作4的合并操作，程序将提示为空表，否则程序会按升序输出两个班级合并后学生成绩情况 ，即翻转了降序排列的合并链表。

输入6，将进入输出全部学生成绩的功能，程序将分别输出两个班各自的学习成绩情况，无需输入。

输入7，退出程序。

输入其他数字，都会提示

*指令只能为1-7之间的某个整数值*

整体特色：

在所有可以输入指令的地方输入非法字符，将提示：

*输入的信息中含非法字符，请检查输入格式再请重新输入：*

循环直到输入数据符合要求才会继续运行程序。

输入

，，。。

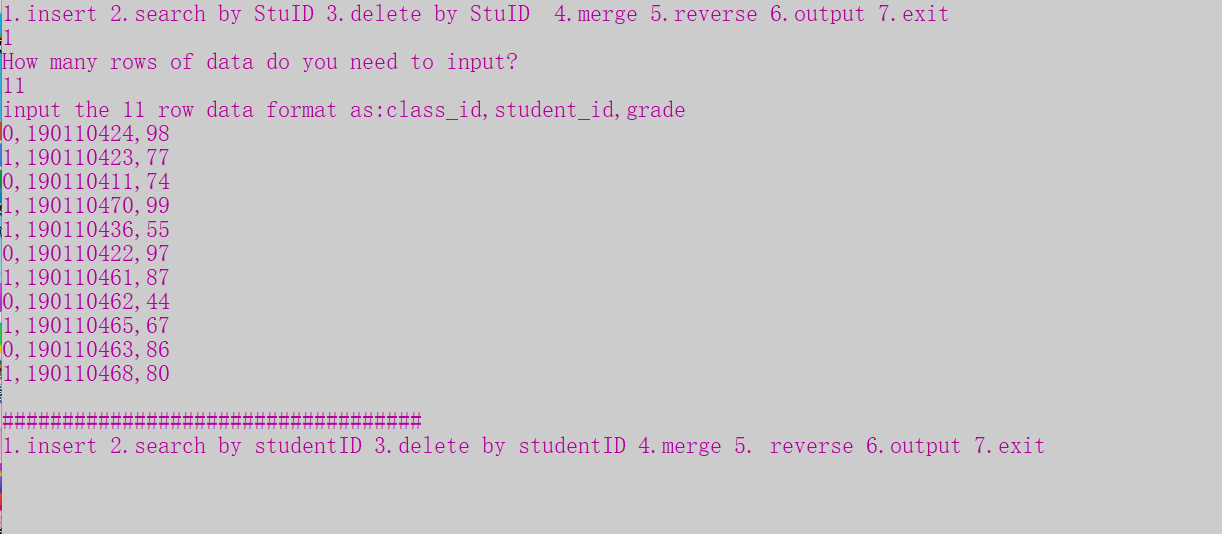
输出

*输入的信息中含非法字符，请检查输入格式再请重新输入：*

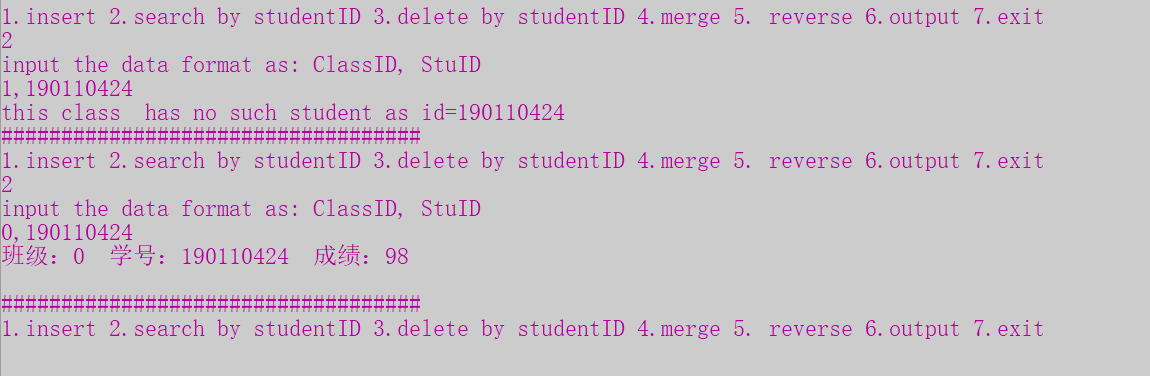
# 四、结果

程序正确运行的结果截图。

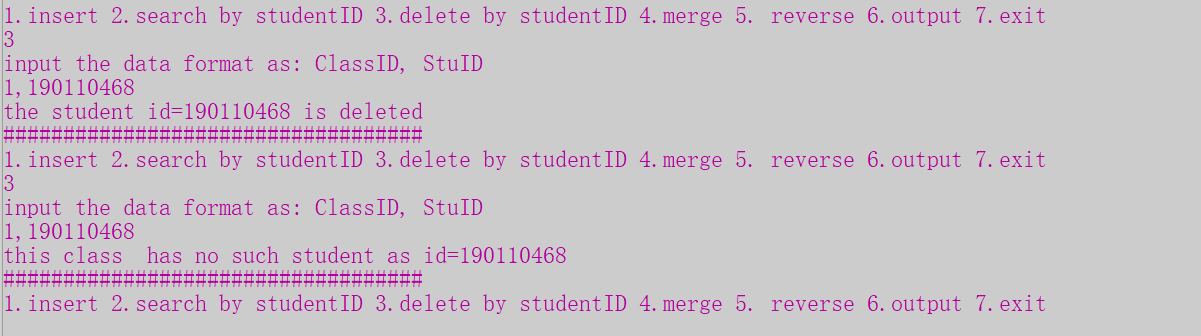
输入1，执行插入数据功能



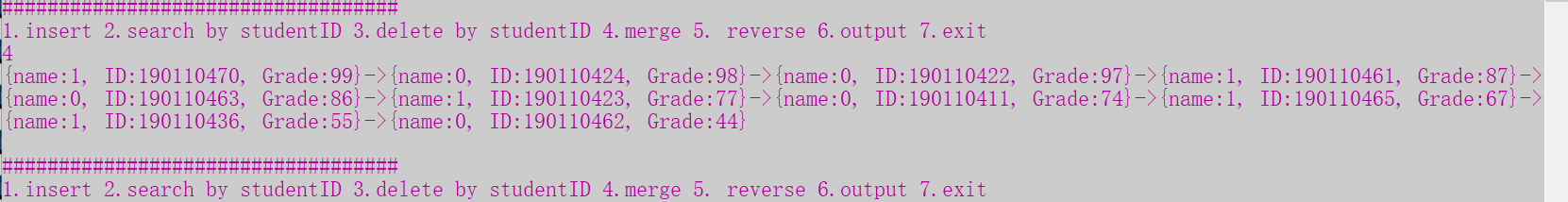
输入2，按班级和学号查找学生信息



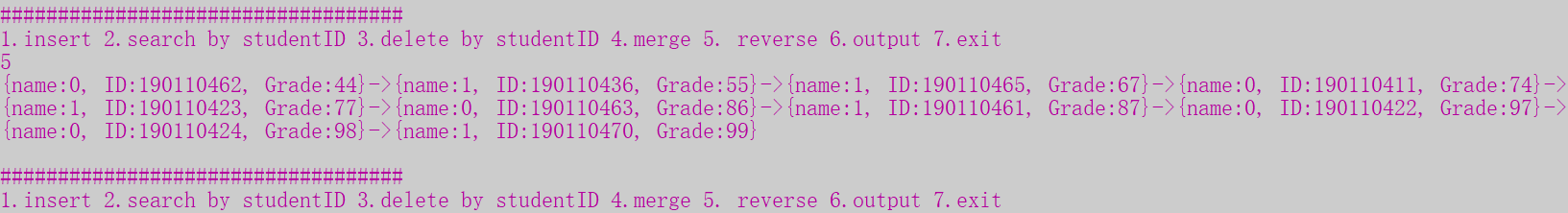
输入3，按照班级和学号删除学生信息功能



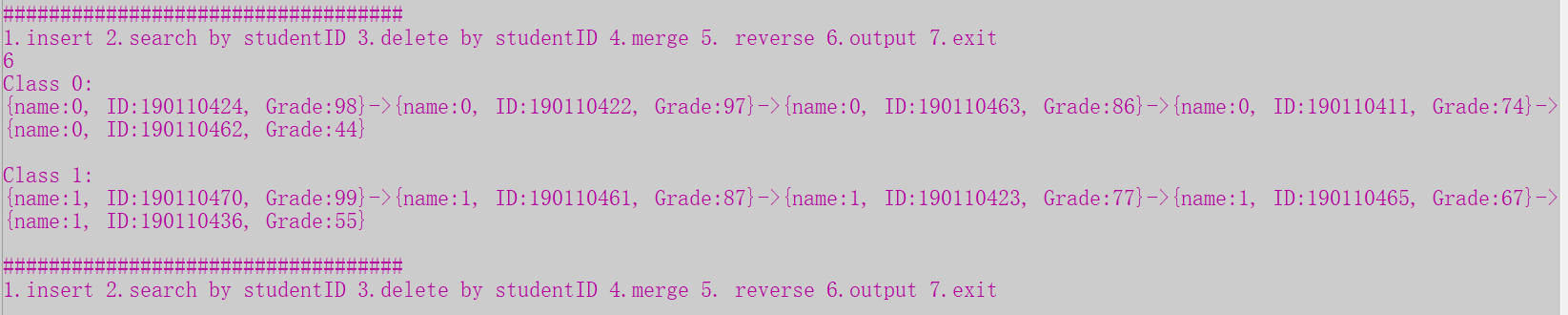
输入4，合并两个班级链表功能



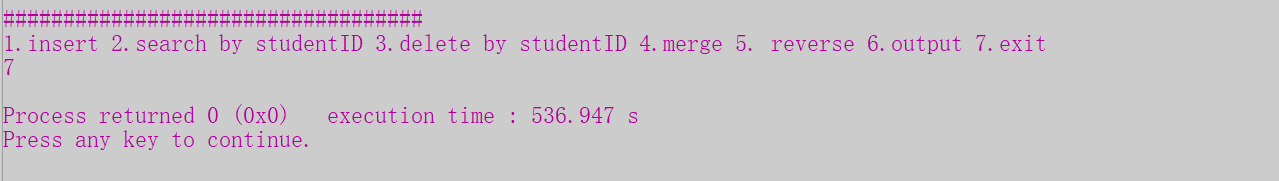
输入5，翻转合并后链表的功能



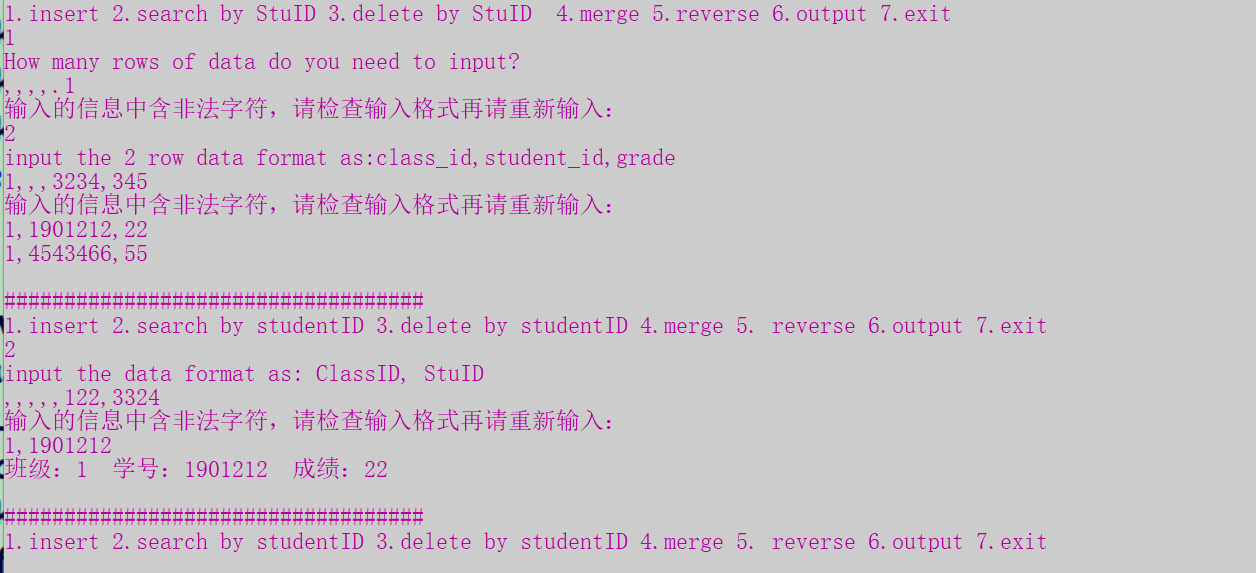
输入6，按班级输出所有学生成绩



输入7退出程序



非法输入



# 五、总结

该实验涉及到的数据结构和算法，以及遇到的问题和收获。

这次实验是基于可以动态扩展的线性链表完成的。在按降序插入学生的成绩时，线性链表的延展性强，灵活性高的特点得以充分体现。如若采用静态存储的方式，将提高程序的时间复杂度，且资源的利用率比较低。利用线性链表，可以根据实际情况随时插入数据，并不干扰其他数据的位置。在建立了链表的基础上，删除，查找和输出操作都变得十分简单。同时利用归并算法，巧妙的合并了两个有序的链表，时间和空间的效率都很高。最后采用原地翻转的方式将链表翻转成新的链表，从降序变成升序，管理学生的成绩就变得很方便了。

但是，程序还是存在许多可以改进的地方。比如许多操作需要建立在其他操作已经完成的基础上才能输出，比如如果没有执行合并链表操作就不能执行翻转链表的操作。甚至没有执行过一次插入操作也不能进行查找和删除。虽然并不会导致程序崩溃（直接输出为空表或者没找到），但是会让用户体验变差。可以让用户在执行一个空操作时（链表为空），先转到让链表不为空的操作，先输入数据再执行其他操作，这样提示就可以让用户更加便捷的完成业务。

在设计程序的过程中，收获颇丰。首先是对异常输入的拦截，对于不是整型的非法输入，都进行了有效拦截，防止导致程序崩溃。同时在开发中也学会了如何在一个大框架内填充逻辑，运用所学的知识，将数据结构化成真正实用的工具，极大方便了针对任务的开发任务。在写完程序后，大部分时间都用在了实验报告的完善上，学会了怎样总结实验，利用流程图清楚表达实验过程，填写用户手册，真正完成了一个小项目。这让我对未来的学习充满了希望。