实验一 线性结构及其应用

1001 体测成绩

一. 实验目的

掌握线性表的链式存储结构和基本操作，重点巩固和体会链表的遍历等基本应用。

二．实验内容

众所周知，大学生体测的成绩一般呈现逐年下降的趋势。贴心的辅导员为了帮大家记录身体素质最好的时刻，决定统计一下大一新生的体测成绩。

需求：

1. 存储链表：按头插法将两个班的成绩存储到链表中。注：输入的成绩是降序的，因此链表存储的成绩是升序的。
2. 反转链表：将建立的两个升序链表按降序进行反转
3. 链表交点：

辅导员亲自录入时，由于班级分数最低的若干同学恰好同分，她不小心将链表的尾部交叉到了一起，只保留了一部分同学。辅导员很着急，因为她想重点关注这些体测成绩吊车尾的学生，劝导他们多锻炼身体。现在辅导员请你帮她编程解决这个问题。

你的任务是编程**找出两个交叉链表的第一个公共交点，你只能拿到两个链表的头结点**。注：**创建交叉链表的代码已给出**。

交叉链表的产生过程如图：

三．输入输出格式

输入分为三个部分：

1. 两个班级的人数n1、n2
2. 两个班级的成绩（两个班分别降序）
3. 两个链表交叉之前的结点个数m1、m2

输入示例：

|  |
| --- |
| 5 6  Student1 100  Student2 90  Student3 80  Student4 70  Student5 60  Student6 95  Student7 88  Student8 82  Student9 80  Student10 70  Student11 60  3 4 |

输出也分为三个部分：

1. 存储链表的结果（升序）
2. 反转链表的结果（降序）
3. 链表交点的结点

输出示例：

|  |
| --- |
| Store Linkedlist  {ID:Student5, Grade:60}->{ID:Student4, Grade:70}->{ID:Student3, Grade:80}->{ID:Student2, Grade:90}->{ID:Student1, Grade:100}  {ID:Student11, Grade:60}->{ID:Student10, Grade:70}->{ID:Student9, Grade:75}->{ID:Student8, Grade:82}->{ID:Student7, Grade:88}->{ID:Student6, Grade:95}  Reverse Linkedlist  {ID:Student1, Grade:100}->{ID:Student2, Grade:90}->{ID:Student3, Grade:80}->{ID:Student4, Grade:70}->{ID:Student5, Grade:60}  {ID:Student6, Grade:95}->{ID:Student7, Grade:88}->{ID:Student8, Grade:82}->{ID:Student9, Grade:75}->{ID:Student10, Grade:70}->{ID:Student11, Grade:60}  Cross node  {ID:Student4, Grade:70} |

四．实验要求

1. 使用链表实现上述需求。
2. 编程语言：C（C++和其他编程语言目前平台不支持）
3. 建表时要求成绩升序存储。
4. 一个班级的成绩存在一个链表中。
5. main函数及部分函数已给出，请不要改动，你只需完成其他函数。
6. 寻找链表交点时，请不要使用学号和分数进行判断，且尽可能使用较少的比较次数。
7. 请到实验平台<http://10.249.176.82:9000/>完成题目1001。

1002 栈的基本操作

一．实验目的

实现栈的基本操作。

二．实验内容

用数组实现栈的基本操作，包括：

1. Push 将元素压入栈顶
2. Pop 获取并移除栈顶元素
3. GetTop 获取栈顶元素
4. StackEmpty 判断栈是否为空

三．输入输出格式

单次输入一行为一个独立的指令，一行中是一个或几个整型参数，用空格隔开。

第一个参数是指令，0-3对应上述四种操作，-1表示退出程序。

1. Push 将元素压入栈顶

输入：第二个参数为操作次数，后续参数为数据

|  |
| --- |
| 0 7 0 1 2 3 4 5 6 /\*入栈7次，数据为0、1、2、3、4、5、6\*/ |

输出：栈的全部元素（自栈顶至栈底，用空格隔开）

|  |
| --- |
| Stack: 6 5 4 3 2 1 0 |

1. Pop 获取并移除栈顶元素

输入：第二个参数为操作次数

|  |
| --- |
| 1 2 /\*将栈顶元素弹出栈并获取\*/ |

输出：弹出的栈顶元素和栈的剩下元素（栈顶至栈底，用空格隔开）

|  |
| --- |
| Pop: 6 /\*依次弹出栈顶元素6、5\*/  Stack: 5 4 3 2 1 0  Pop: 5  Stack: 4 3 2 1 0 |

弹出失败

|  |
| --- |
| Pop failed /\*失败了几次就会打印几次该信息\*/ |

1. GetTop 获取栈顶元素

输入：

|  |
| --- |
| 2 /\*获取栈顶元素，而不弹出\*/ |

输出：栈顶元素和栈的剩下元素（栈顶至栈底，用空格隔开）

|  |
| --- |
| GetTop: 4  Stack: 4 3 2 1 0 |

获取失败

|  |
| --- |
| GetTop failed |

1. StackEmpty 判断栈是否为空

输入：

|  |
| --- |
| 3 /\*判断栈是否为空\*/ |

输出：空

|  |
| --- |
| The Stack is Empty /\*空\*/ |

输出：非空

|  |
| --- |
| The Stack is not Empty /\*非空\*/  Stack: 4 3 2 1 0 |

四．实验要求

1. 基于数组实现上述需求
2. 编程语言：C（C++和其他编程语言目前平台不支持）
3. main函数及部分函数已给出，请不要改动，你只需完成其他函数。你也可以不使用模板代码，但请注意代码规范以及输入输出格式。
4. 请到实验平台<http://10.249.176.82:9000/>完成题目1002。

1003 队列的基本操作

一．实验目的

实现队列的基本操作。

二．实验内容

用栈实现先进先出队列的基本操作，包括：

1. EnQueue 将元素插入到队列的尾
2. DeQueue 获取并移除队列头元素
3. GetHead 获取队列开头的元素
4. QueueEmpty 判断队列是否为空

并实现辅助接口 QueueToArray 将队列中元素按照从头到尾的顺序写到数组中。并且思考：**怎么使用两个栈能使连续的EnQueue / DeQueue效率最高？**

三．输入输出格式

单次输入一行为一个独立的指令，一行中是一个或几个整型参数，用空格隔开。

第一个参数是指令，4-7对应上述四种操作，-1表示退出程序。

1. EnQueue 将元素插入到队列的尾

输入：第二个参数为操作次数，后续参数为数据

|  |
| --- |
| 4 7 0 1 2 3 4 5 6 /\*入队7次，数据为0、1、2、3、4、5、6\*/ |

输出：队列的全部元素（队头到队尾，用空格隔开）

|  |
| --- |
| Queue: 0 1 2 3 4 5 6 |

1. DeQueue 获取并移除队列头元素

输入：第二个参数为操作次数

|  |
| --- |
| 5 3 /\*弹出队头元素并获取\*/ |

输出：弹出的队头元素和队列的剩下元素（队头到队尾，用空格隔开）

|  |
| --- |
| DeQueue: 0  Queue: 1 2 3 4 5 6  DeQueue: 1  Queue: 2 3 4 5 6  DeQueue: 2  Queue: 3 4 5 6 |

弹出失败

|  |
| --- |
| DeQueue failed /\*失败了几次就会打印几次该信息\*/ |

1. GetHead 获取队列开头的元素

输入：

|  |
| --- |
| 6 /\*获取队头元素\*/ |

输出：队头元素和队列的剩下元素（队头到队尾，用空格隔开）

|  |
| --- |
| GetHead: 3  Queue: 3 4 5 6 |

获取失败

|  |
| --- |
| GetHead failed |

1. QueueEmpty 判断队列是否为空

输入：

|  |
| --- |
| 7 /\*判断队列是否为空\*/ |

输出：空

|  |
| --- |
| The Queue is Empty /\*队列空\*/ |

输出：非空

|  |
| --- |
| The Queue is not Empty /\*队列非空\*/  Queue: 3 4 5 6 |

四．实验要求

（1） 可基于题目1002数组实现的栈来实现本题的队列。也可通过其他方法实现的栈来实现本题的队列。

（2）编程语言：C（C++和其他编程语言目前平台不支持）

（3） main函数及部分函数已给出，请不要改动，你只需完成其他函数。你也可以不使用模板代码，但请注意代码规范以及输入输出格式。

（4） 请到实验平台<http://10.249.176.82:9000/>完成题目1003。

源代码评分标准

1. 实验一总分100分
2. 体测成绩（40分）
3. 栈的基本操作（30分）
4. 队列的基本操作（30分）
5. 对于实验的每个编程小题，通过**所有测试用例**才可拿到**满分**，没有通过则**零分**。对于未通过的用例实验平台会给予相应的提示。
6. 建议在本地IDE（如**codeblocks**）编写代码，编译测试通过后再将代码复制到实验平台进行测试。
7. **禁止**忽略逻辑直接printf()输出结果，所有代码实验平台后台都有存储，一旦发现，本次实验0分。
8. 本次实验无时间、空间复杂度限制，返回此类信息仅供参考。