项目2说明文档

数据结构课程设计

——两个有序链表序列的交集

作 者 姓 名： 谭欢秘

学 号： 1954185

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 分析 1](#_Toc54123798)

[1.1 背景分析 1](#_Toc54123799)

[1.2 功能分析 1](#_Toc54123800)

[2 设计 1](#_Toc54123801)

[2.1 数据结构设计 1](#_Toc54123802)

[2.2 类结构设计 1](#_Toc54123803)

[2.3 类成员与操作设计 1](#_Toc54123804)

[2.4 主函数设计 2](#_Toc54123805)

[3 实现 3](#_Toc54123806)

[3.1输入有序链表序列的实现 3](#_Toc54123807)

[3.1.1 input()函数流程图 3](#_Toc54123808)

[3.1.2 input()函数核心代码 3](#_Toc54123809)

[3.1.3 input()函数演示 4](#_Toc54123810)

[3.2 打印有序链表序列的实现 5](#_Toc54123811)

[3.2.1 display()打印函数流程图 5](#_Toc54123812)

[3.2.2 display()打印函数核心代码 5](#_Toc54123813)

[3.2.3 display()打印函数演示 6](#_Toc54123814)

[3.3重载”\*”取交集的实现 7](#_Toc54123815)

[3.3.1 operator\*()函数流程图 7](#_Toc54123816)

[3.3.2 operator\*()函数核心代码 7](#_Toc54123817)

[3.3.3 operator\*()函数截屏示例 8](#_Toc54123818)

[3.4 重载”=”复制链表的实现 9](#_Toc54123819)

[3.4.1 operator=()函数流程图 9](#_Toc54123820)

[3.4.2 operator=()函数核心代码 9](#_Toc54123821)

[3.4.3 operator=()函数截屏示例 9](#_Toc54123822)

[3.5 主函数的实现 10](#_Toc54123823)

[3.5.1 主函数流程图 10](#_Toc54123824)

[3.6.2 总体系统核心代码 10](#_Toc54123825)

[3.6.3 总体系统截屏示例 10](#_Toc54123826)

[4 测试 11](#_Toc54123827)

[4.1 功能测试 11](#_Toc54123828)

[4.1.1 插入功能测试 11](#_Toc54123829)

[4.1.2 删除功能测试 11](#_Toc54123830)

[4.1.3 查找功能测试 12](#_Toc54123831)

[4.1.4 修改功能测试 12](#_Toc54123832)

[4.1.5 统计功能测试 13](#_Toc54123833)

[4.2 边界测试 13](#_Toc54123834)

[4.2.1 初始化无输入数据 13](#_Toc54123835)

[4.2.2 删除头结点 14](#_Toc54123836)

[4.2.3 删除后链表为空 14](#_Toc54123837)

[4.3 出错测试 15](#_Toc54123838)

[4.3.1 考生人数错误 15](#_Toc54123839)

[4.3.2 操作码错误 15](#_Toc54123840)

[4.3.3 插入位置不存在 15](#_Toc54123841)

[4.3.4 删除考号不存在 16](#_Toc54123842)

[4.3.5 查找考号不存在 16](#_Toc54123843)

[4.3.6 修改考号不存在 17](#_Toc54123844)

# 1 分析

## 背景分析

本题目已知输入的两个链表序列S1、S2均为非降序排列，在求解交集S3时，则在求解问题时应充分利用“非降序”的性质来减少程序的时间、空间开销。

## 1.2 功能分析

首先是设计链表序列的输入输出功能。输入时，统一以“-1”作为序列的结尾，需要注意的是不将“-1”存入链表中。输出时以空格隔开每个元素，最后一个元素最后没有空格。然后是最重要的求交集功能，求出交集。最后，在求交集、输出交集的过程中，可能用到复制链表的功能，还应设计一个复制函数。

综上所述，本项目至少应该具有输入、输出、求交集以及复制链表序列的功能。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求大量的增加、删除、修改操作，而链表进行增加、删除等操作十分简便，因此考虑使用链表数据结构。同时，为了实现简易，在第一个结点之前附加一个头结点，这样就使得增加或者删除头结点与处理其他结点方法相同，使得程序简洁。

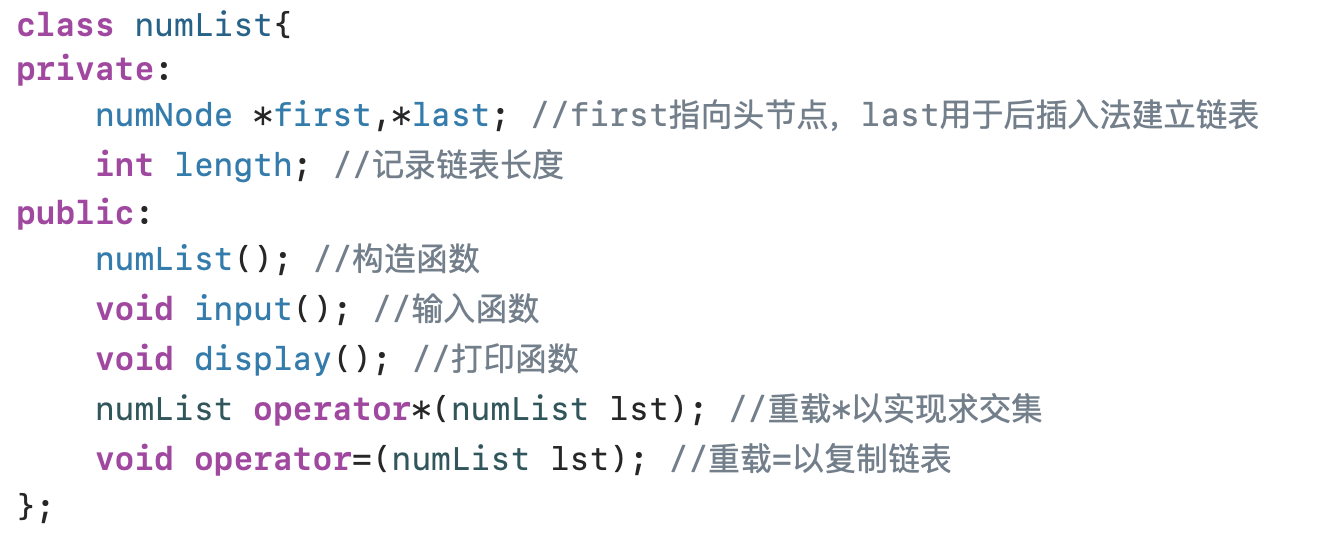
## 2.2 类结构设计

经典的链表一般包括两个抽象数据类型（ADT）——链表结点类（Node）与链表类（List），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。为方便处理与直观表示，本题采用描述链表结点类（numNode），这样使得链表结点类（numList）可以访问链表结点。

## 2.3 类成员与操作设计

**链表结点类（numNode）**



**链表类（StuList）**

## 2.4 主函数设计

声明三个numList类的对象s1、s2、s3，它们都是有头节点的链表。首先分别调用s1.input()、s2.input()函数实现s1、s2的初始化，完成对链表s1、s2的创建和输入数据工作。然后用“s1\*s2”求交集（返回numList类型）后，将返回值赋给s3。最后输出交集s3序列。

# 3 实现

## 3.1输入有序链表序列的实现

### 3.1.1 input()函数流程图

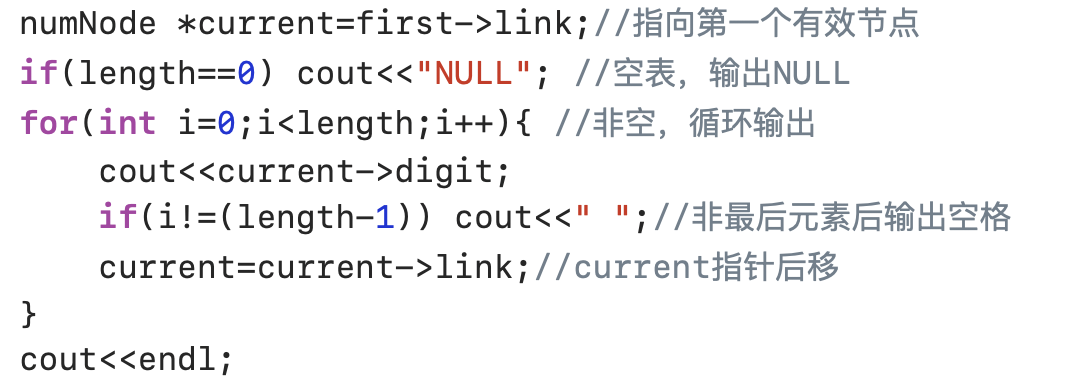
### 3.1.2 input()函数核心代码



## 3.2 打印有序链表序列的实现

### 3.2.1 display()打印函数流程图

### 3.2.2 display()打印函数核心代码



## 3.3重载”\*”取交集的实现

### 3.3.1 operator\*()函数流程图

### 3.3.2 operator\*()函数核心代码

## 3.4 重载”=”复制链表的实现

### 3.4.1 operator=()函数流程图

### 3.4.2 operator=()函数核心代码

## 3.5 主函数的实现

### 3.5.1 主函数流程图

### 3.6.2 总体系统核心代码

# 4 测试

## 4.1 功能测试

### 4.1.1 输入功能测试

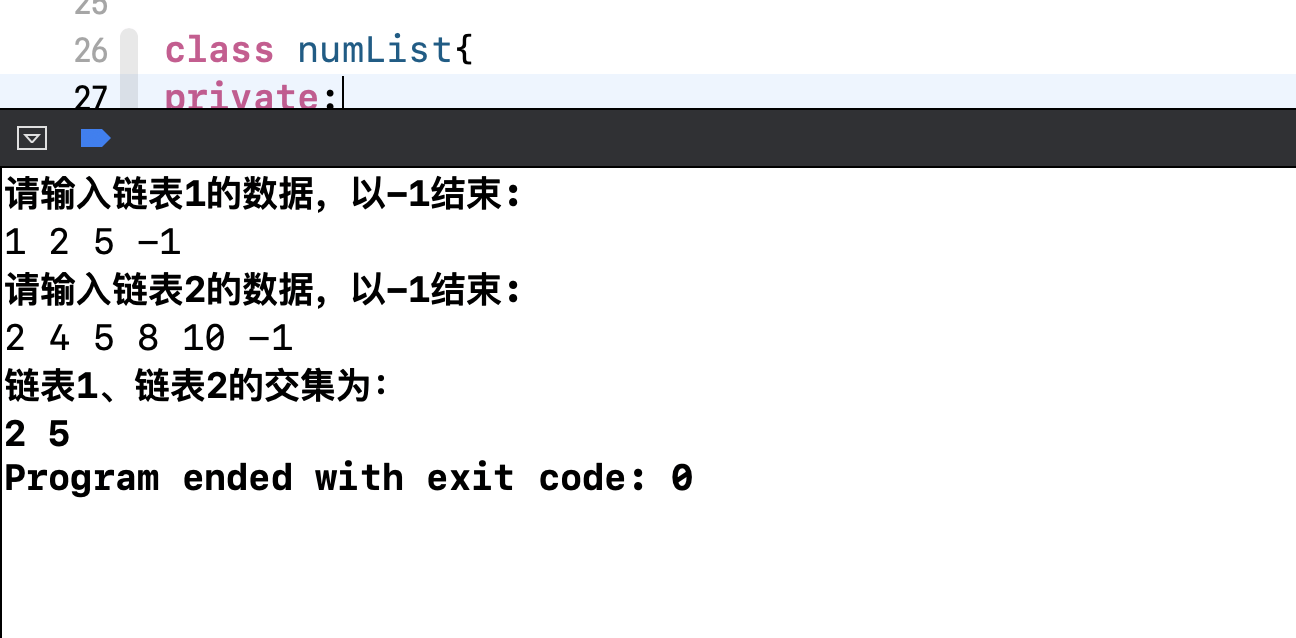
**测试用例**：1 2 5 -1

2 4 5 8 10 -1

**预期结果**：

输入两个集合，分别为{1,2,5}和{2,4,5,8,10}

**实验结果**



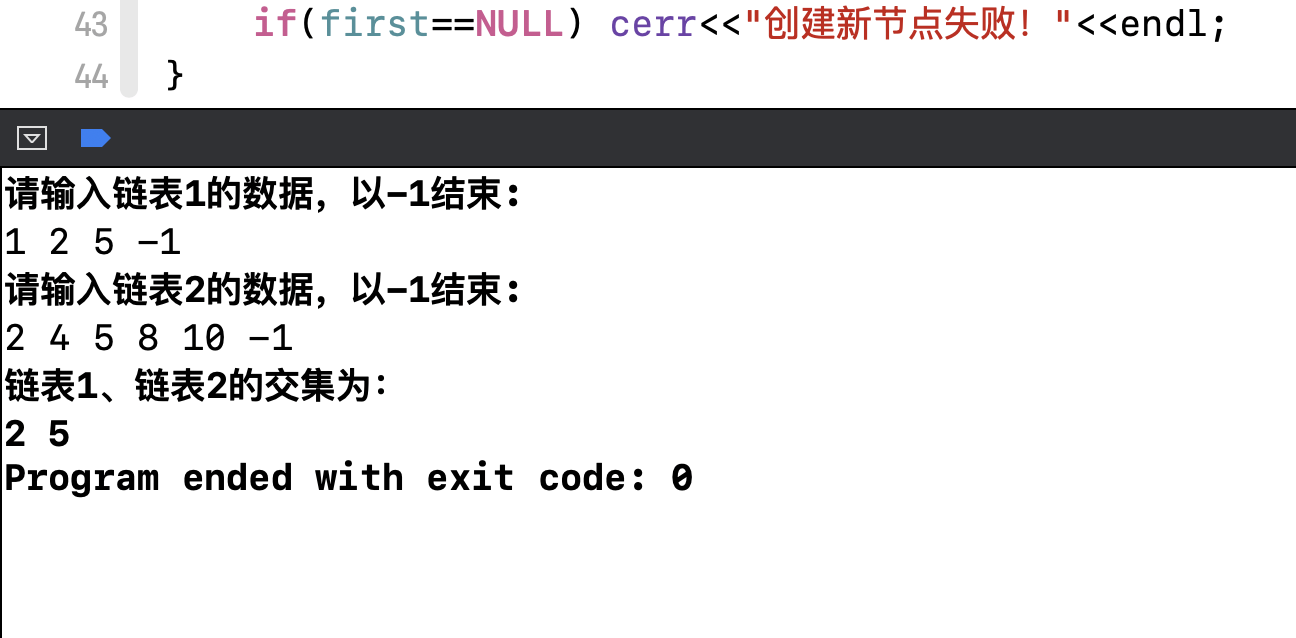
### 4.1.2 取交集、打印功能测试

**测试用例：**1 2 5 -1

2 4 5 8 10 -1

**预期结果：**2 5

**实验结果：**



## 4.2 边界测试

### 4.2.1 初始化输入非法

**测试用例：**初始输入非法，输入序列中有非正整数

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**

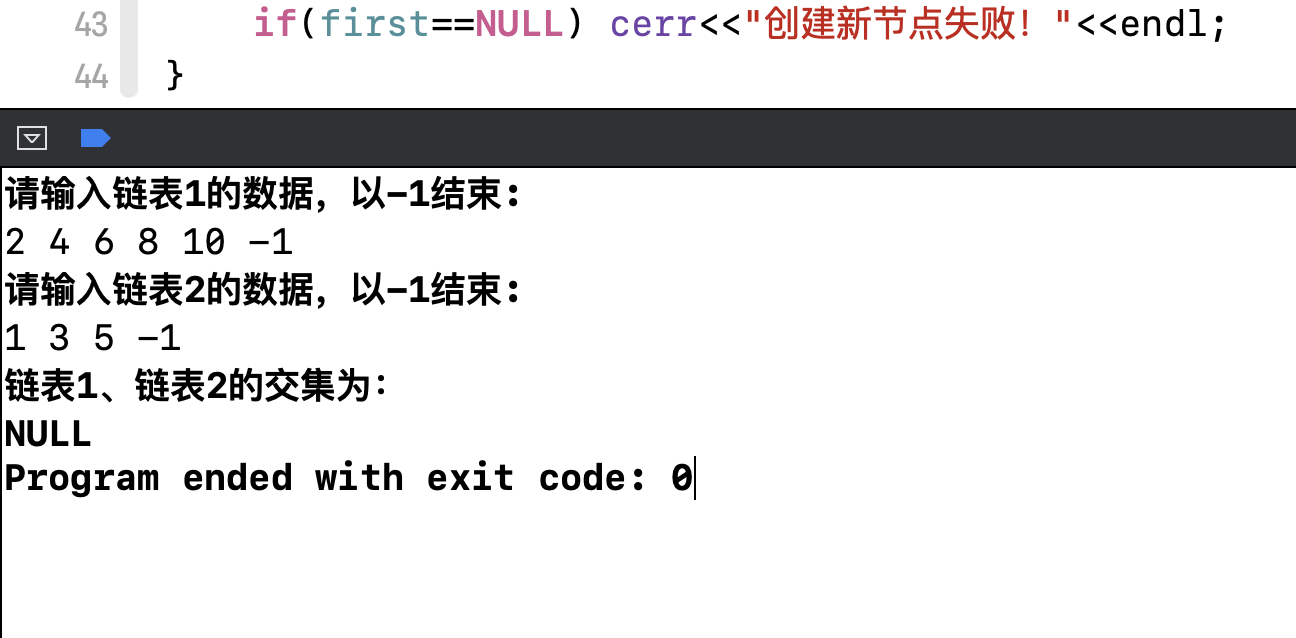


### 4.2.2 两个链表交集为空

**测试用例：**1 3 5 -1

2 4 6 8 10 -1

**预期结果：**程序正常运行，输出交集为NULL，不崩溃。

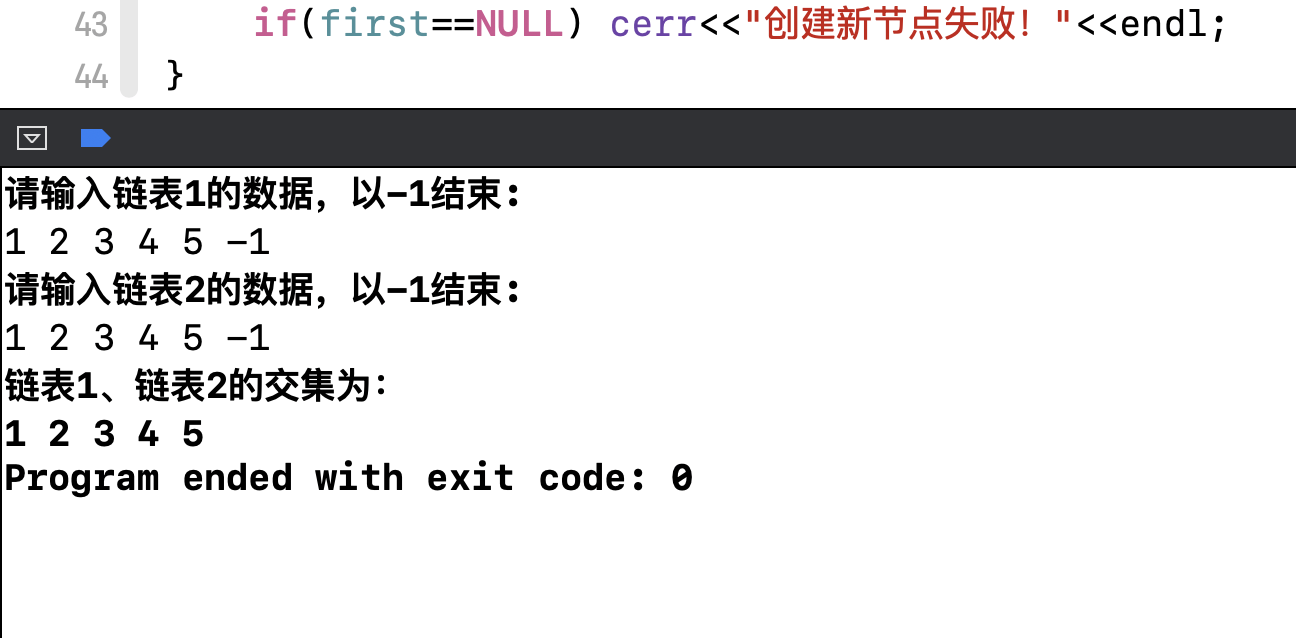
**实验结果：**

### 4.2.3 链表完全相交

**测试用例：**1 2 3 4 5 -1

1 2 3 4 5 -1

**预期结果：**程序正常运行，不崩溃。

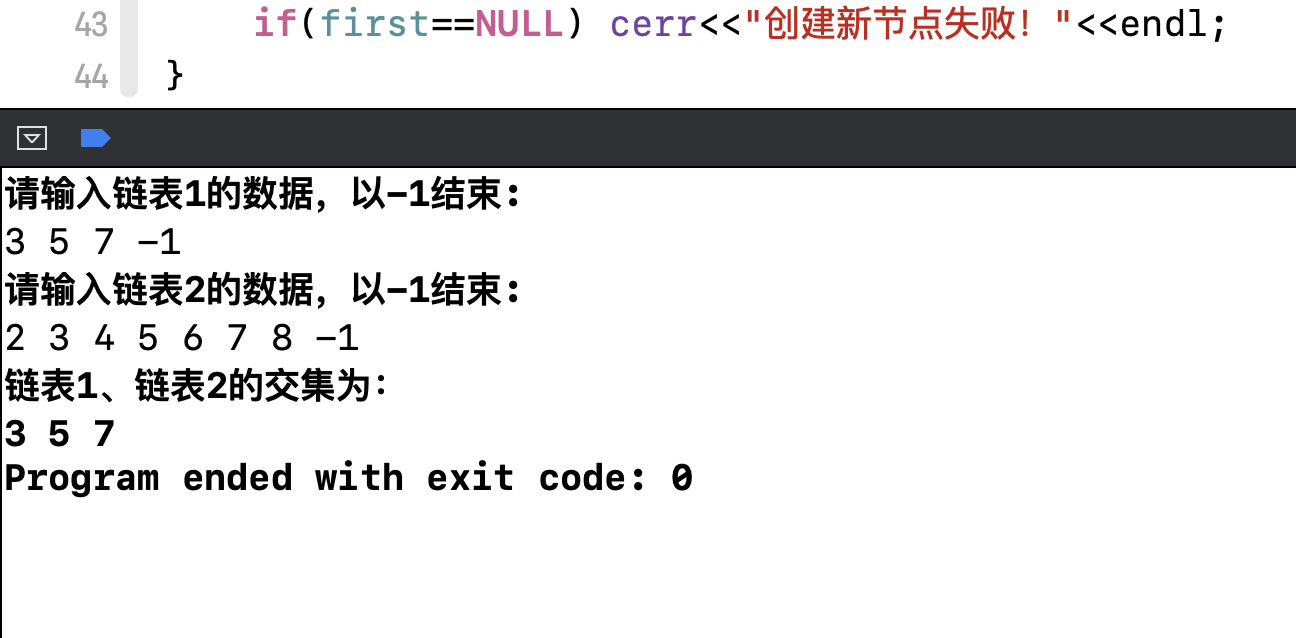
**实验结果：**

### 4.2.4 其中一个链表是另一链表的子集

**测试用例：**3 5 7 -1

2 3 4 5 6 7 8 -1

**预期结果：**程序正常运行、输出交集，不崩溃。

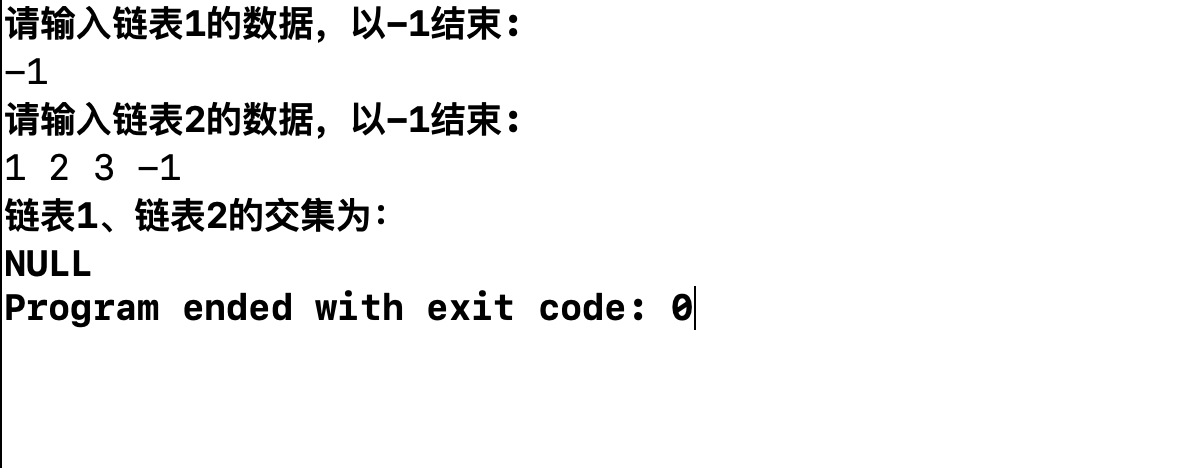
**实验结果：**

### 4.2.5 有链表为空

**测试用例：**-1

1 2 3 -1

**预期结果：**程序正常运行，输出交集为NULL，不崩溃。

**实验结果：**