# 四川师范大学 实验报告

学期： 2024 至 2025 第一学期 实验成绩：

课程名字：《程序设计基础——数据结构(C语言版)》 专业： 信息与计算科学

班级： 2023 级 9 班 实验编号： 02

实验项目： 实验二 指导老师： 冯山

姓名： 刘智恒 学号： 2023060522

**一、实验题目**

线性表及其应用

**二、实验目的及要求**

1.熟悉线性表的ADT及其存储结构和基本操作算法实现特点。

2.针对给定问题，运用线性表存储结构和基本操作进行问题求解算法设计和实现。

3.熟练掌握线性表的顺序存储结构和动态存储结构的差异。

**三、实验内容：(类C算法的程序实现，任选其二)**

1.设计并实现一元多项式的加、减、乘、除运算的算法。(加法必做，其它选做)

2.设计并实现有序表的并的算法。

3.设计并实现集合的并、交、补、差运算算法。

4.设计并实现约瑟夫问题的求解算法。

**四、实验准备**

1.计算机设备;

2.程序调试环境的准备，本实验采用**Microsoft Visual Studio**环境;

3.实验内容的算法分析与代码设计与分析准备；

4.实验源程序**Exp\_2(1)，Exp\_2(2)**准备。

5.实验测试用例：

实验内容1：

（1）**加法运算：**

**第一个一元多项式：n = 4（一共4项）；**

**系数 指数：1 5 / -3 2 / 7.5 3 / 0 2；**

**第二个一元多项式：n = 3（一共3项）；**

**系数 指数：1 3 / 1.5 2 / 5 0；**

（2）**减法运算：**

**第一个一元多项式：n = 5（一共5项）；**

**系数 指数：2 4 / -2 2 / 3 2 / 3 3 / 4.5 1；**

**第二个一元多项式：n = 4（一共4项）；**

**系数 指数：-3.6 3 / 2.5 2 / 2 1 / 1.6 3；**

（3）**乘法运算：**

**第一个一元多项式：n = 4（一共4项）；**

**系数 指数：-1 4 / -2 3 / -3 2 / -4 1；**

**第二个一元多项式：n = 5（一共5项）；**

**系数 指数：1 1 / 2 2 / 3 3 / 4 4 / 5 5；**

实验内容3：

**集合A：a = 5（5个元素）：{ 3 2 1 2 4 }**

**集合B：b = 7（7个元素）：{ 2 3 1 2 5 4 3 }**

（1）**全集U：u = 6（6个元素）：{ 2 3 5 1 4 6 }**

（2）**全集U：u = 4（4个元素）：{ 2 3 4 1 }**

**集合A：a = 7（7个元素）：{ 3 2 1 2 4 6 5 }**

**集合B：b = 4（4个元素）：{ 2 3 1 2 }**

（3）**全集U：u = 5（5个元素）：{ 2 3 5 1 4 }**

（4）**全集U：u = 10（10个元素）：{ 2 3 5 1 5 6 8 4 7 9 }**

**五、实验过程**

**（一）问题分析**

**1.实验内容1：**

（1）确定数据结构：一元多项式的其中一项包含**系数（coef）**和**指数（exp）**两个基本属性，本问题选择**单向链表**结构，类型定义如下：

**typedef struct PolyNode{**

**double coef; //系数**

**int exp; //指数**

**struct PolyNode \*next; //指向下一结点（下一项）的指针**

**}PolyNode, \*Polynomial; //Polynomial生成头节点，PolyNode生成中间结点**

（2）确定相关参数的**输入次序**。此处确定先输入一元多项式的项数**n**，然后输入每一项的系数**coef**和指数**exp**。其中系数**coef**和指数**exp**可以通过函数**创建一个新的多项式结点**来进行输入，然后将每个具有给定系数**coef**和指数**exp**的结点再通过另一个函数**插入到多项式链表**中。

（3）确定相关运算。本问题主要实现**两个一元多项式的加法、减法、乘法**运算。其中，加法使用**类有序表合并**算法，减法此处只考虑**第一个多项式减去第二个多项式**。

**注：本问题将三种运算进行综合，通过一个简易的界面菜单对三种不同运算进行选择。**

（4）结点插入分析：由于后续的加法运算以及减法运算会使用到**类有序表合并**算法，故此处结点插入**根据每一项的指数大小进行降序插入。**

（5）相关运算分析：

**【加法运算】**根据**有序表合并**的思想，类似地，可对两个即将进行相加的多项式**poly1**和**poly2设定两个指针p1和p2，分别指向各自第一项（即最高次项），再设定一个多项式poly3用于插入相加结果**。然后**以指数大小为依据**，逐项进行比较：若各自指针所指的**指数相同**，则将对应系数相加，取其中一个指数为最终指数插入poly3中，同时**两个指针移动**；若**指数不同**，则插入指数更大的项并**移动其指针**。若其中一个多项式的所有项比较完毕，而另一个多项式还存在剩余项，则插入剩余项到**poly3**中。

**两个一元多项式的加法数学公式如下：**

**【减法运算】**与加法运算的算法类似：只需在进行指数大小比较时，若各自指针所指的**指数相同**，则将系数相加改为相减即可。本问题选择的是**poly1-poly2（即只需将coef1+coef2改为coef1-coef2）**。

**两个一元多项式的加法数学公式如下：**

**【乘法运算】**根据两个一元多项式相乘的性质可知，其运算过程为**“逐项相乘再求和”**，故同样可**设定两个指针p1和p2分别指向poly1和poly2的第一项**，然后**设定两个循环过程**，**外循环遍历poly1的每一项，内循环遍历poly2的每一项**；再将对应相乘的两项的**系数相乘**，**指数相加**，同时伴随指针移动；最终将每一个相乘结果插入到**poly3**中。

**两个一元多项式的乘法数学公式如下：**

其中，**m**和**n**分别是**P1(x)**和**P2(x)**的**最高次幂**。

**2.实验内容3：**

（1）确定数据结构：由于一个集合包含其**元素（element）**和**大小（size）**两个基本属性,故本问题选择**动态顺序表**结构。类型定义如下：

**typedef struct Set{**

**ElemType data; //存储元素**

**int size; //集合大小**

**}Set;**

（2）基本分析：

若要实现集合之间的并集、交集、补集运算，则需要先给定已知元素的集合。本问题只**考虑两个集合**：集合**A**与集合**B**之间的并集、交集、补集运算。

【**并集**】指将两个集合中的**所有元素**合并在一起，此处考虑**元素重复**问题。将并集记为**C**，符号表示如下：

【交集】指将两个集合中的**共有元素**合并在一起，此处考虑**元素重复**问题。将交集记为**D**，符号表示如下：

【补集】具体指集合**A**或集合**B**相对于全集**U**的补集，即全集**U**中不包含集合**A**或集合**B**中的元素的集合。本问题中考虑**给定全集U**，不考虑集合**A**（或集合**B**）相对于集合**B**（或集合**A**）的补集。并且**考虑全集U的大小是否合理**：

**situation 1**：若全集**U**规模**大于**集合**A**（或集合**B**）**并且小于**集合**B**（或集合**A**），则集合**A**（或集合**B**）有补集。

**situation 2**：若全集**U**规模**大于**集合**A**和集合**B**，则集合**A**和集合**B都有各自的补集**。

**situation 3**：若全集**U**规模**小于**集合**A**和集合**B**，则集合**A**和集合**B均无补集**。

将集合**A**的补集记为**E**，集合**B**的补集记为**F**，符号表示如下：

（3）确定关键函数：通过分析并集、交集、补集的特性可知，可以创建一个**通用函数Contain**对集合中的元素进行**存在判定**。**Contain**包含两个参数，**集合seti**和**元素element**，即通过**判断集合seti中是否存在该元素element**得到相应的返回值。

**（二）算法描述**

**1.实验内容1：**

（1）**Status InitPoly(Polynomial\* poly);**

//本算法用于初始化一个空的多项式链表：将Polynomial\*类型的头指针poly置空，表示初始化后的多项式在结构上没有任何项

（2）**PolyNode\* createNode(double coef, int exp);**

//本算法用于创建结点:其中包含待给定double类型的系数coef，以及int类型的指数exp两个参数，并返回一个指向PolyNode结构体的指针

（3）**void InsertNode(Polynomial\* poly, double coef, int exp);**

//本算法用于将一个新的多项式项插入到多项式链表中，并按照项的指数大小进行降序排列插入：其中包含新项的double类型系数coef和int类型指数exp，以及Polynimial类型的指向多项式链表头指针的指针

（4）**void AddPoly(Polynomial \*poly1, Polynomial \*poly2, Polynomial \*poly3);**

**//**本算法用于两个多项式poly1与poly2的相加运算：其中poly1与poly2是两个已知的一元多项式，最后将相加结果返回一个Polynomial\*类型的多项式poly3中

（5）**void SubPoly(Polynomial \*poly1, Polynomial \*poly2, Polynomial \*poly3);**

**//**本算法用于两个多项式poly1与poly2的相减运算（poly1-poly2）:其中poly1与poly2是两个已知的一元多项式，最后将相减结果返回一个Polynomial\*类型的多项式poly3中

（6）**void MulPoly(Polynomial \*poly1, Polynomial \*poly2, Polynomial \*poly3);**

**//**本算法用于两个多项式poly1和poly2的相乘运算:其中poly1与poly2是两个已知的一元多项式,最后将相乘结果返回一个Polynomial\*类型的多项式poly3中

（7）**void freePoly(Polynomial poly);**

//本算法用于释放多项式链表的内存：接受一个Polynomial类型的头节点poly，逐个结点依次释放内存

（8）**void printPoly(Polynomial poly);**

//本算法用于输出显示相应的多项式：接受一个Polynomial类型的头节点poly，同时考虑特殊情形下的简化表示

**2.实验内容3：**

（1）**Status InitSet(Set \*seti, ElemType SIZE)；**

**//**本算法用于为集合分配内存空间，并设置集合的大小：seti指向要初始化的集合，SIZE为集合的大小

（2）**Status FreeSet(Set \*seti)；**

**//**本算法用于释放集合所占的内存：seti指向要释放的集合，同时检查集合数据是否为NULL，如果不为NULL则释放内存，同时将集合大小设置为0

（3）**void PrintSet(Set seti)；**

**//**本算法用于输出打印集合元素：seti作为要打印的集合，打印前对集合中的元素进行升序排序，然后打印集合中不重复的元素

（4）**Status Contain(Set \*seti, ElemType element)；**

**//**本算法用于判断集合中是否包含特定元素：seti指向待检查的集合，element为要检查的元素。遍历集合的每个元素，如果找到与element相等的元素则返回OK

（5）**Status UnionSet(Set \*setA, Set \*setB, Set \*setC)；**

**//**本算法用于计算两个集合的并集：setA和setB为待操作的集合，setC为存储结果的集合。将集合A的所有元素添加到集合C中，然后检查集合B的每个元素，如果集合C中不存在，则将其添加，最后更新集合C的大小

（6）**Status IntersectionSet(Set \*setA, Set \*setB, Set \*setC)；**

**//**本算法用于计算两个集合的交集：setA和setB为待操作的集合，setC为存储结果的集合。遍历较大的集合中的每个元素，检查是否在另一个集合中存在，如果存在则将其添加到集合C，并更新集合C的大小

（7）**Status ComplementSet(Set \*setU, Set \*seti, Set \*setC)；**

**//**本算法用于计算集合的补集：setU为全集，seti为目标集合，setC为存储结果的集合。遍历全集中的每个元素，如果该元素不在目标集合中，则将其添加到集合C，并更新集合C的大小

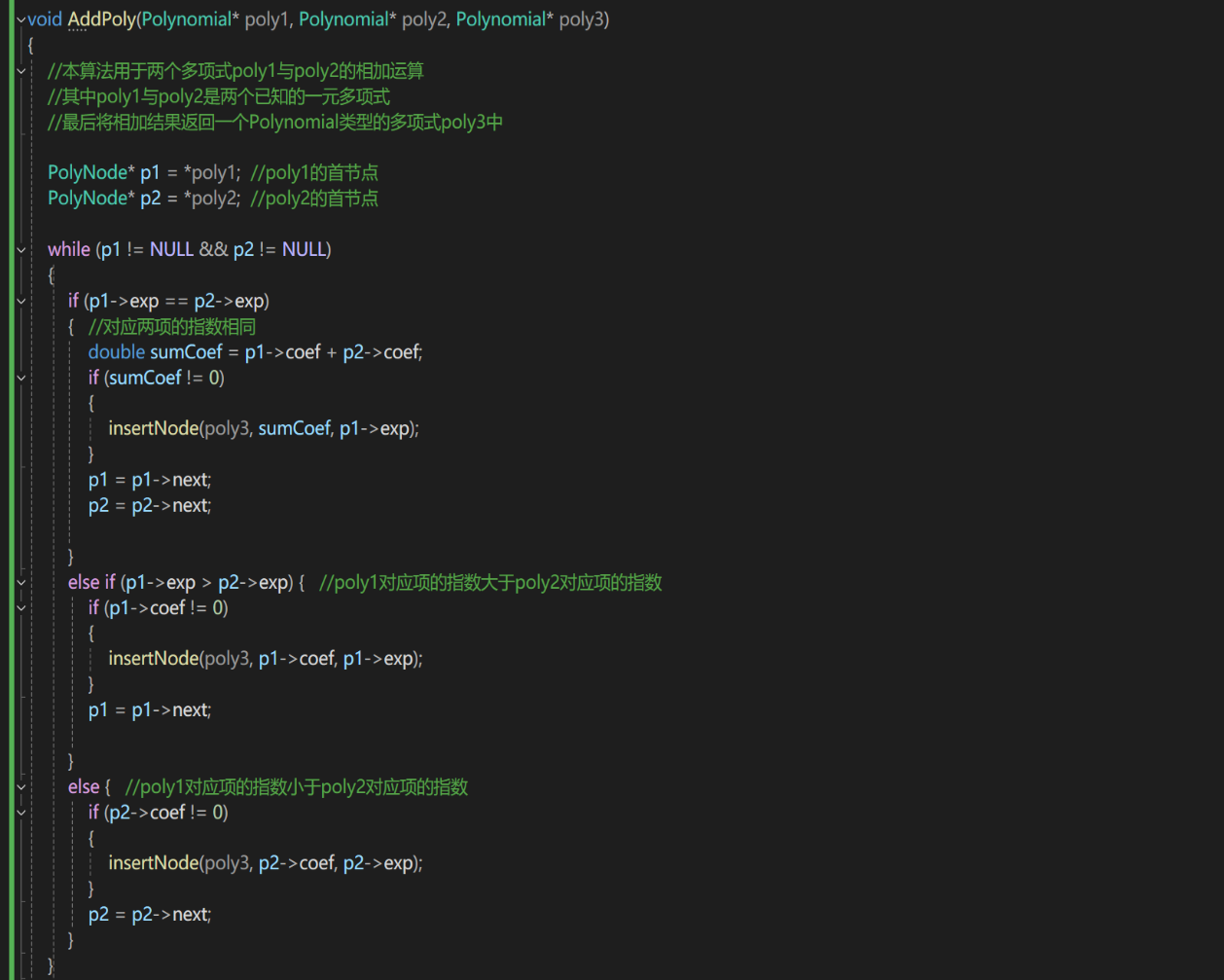
**（三）程序代码**

**1.实验内容1：**

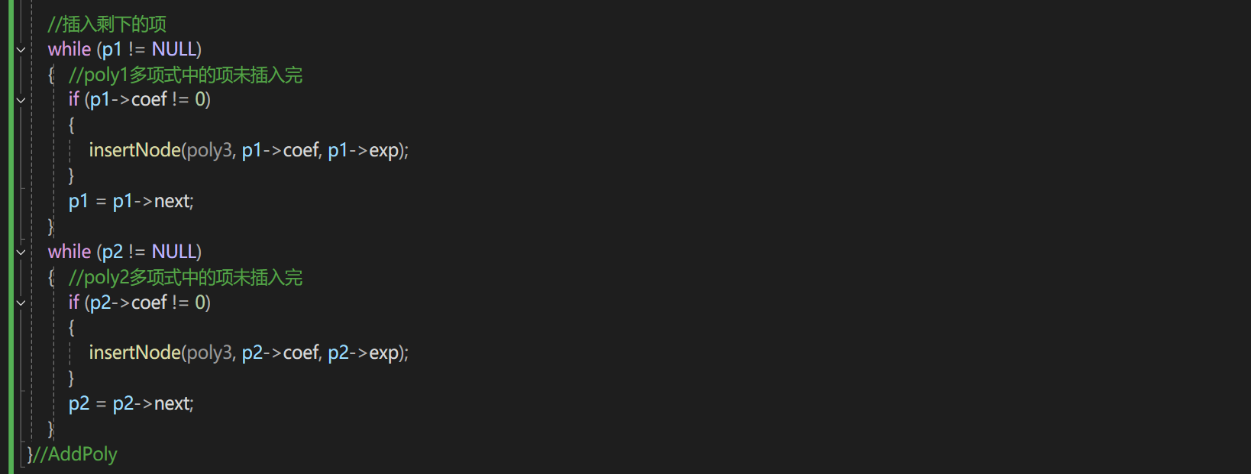
根据以上**实验内容1**的算法描述，求解本问题的程序代码如下：

****

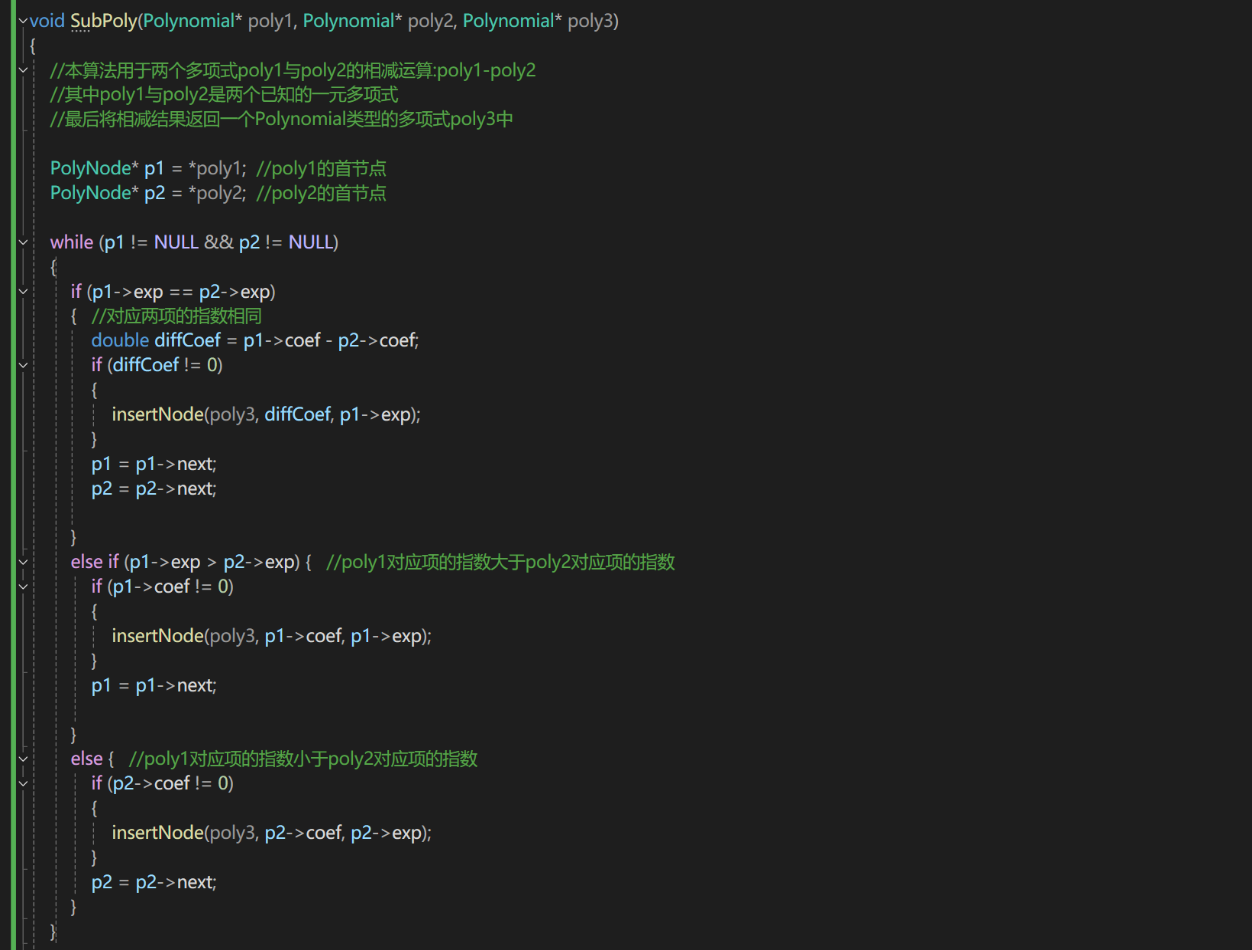
**注：此处只展示三种运算的程序代码图。**

****

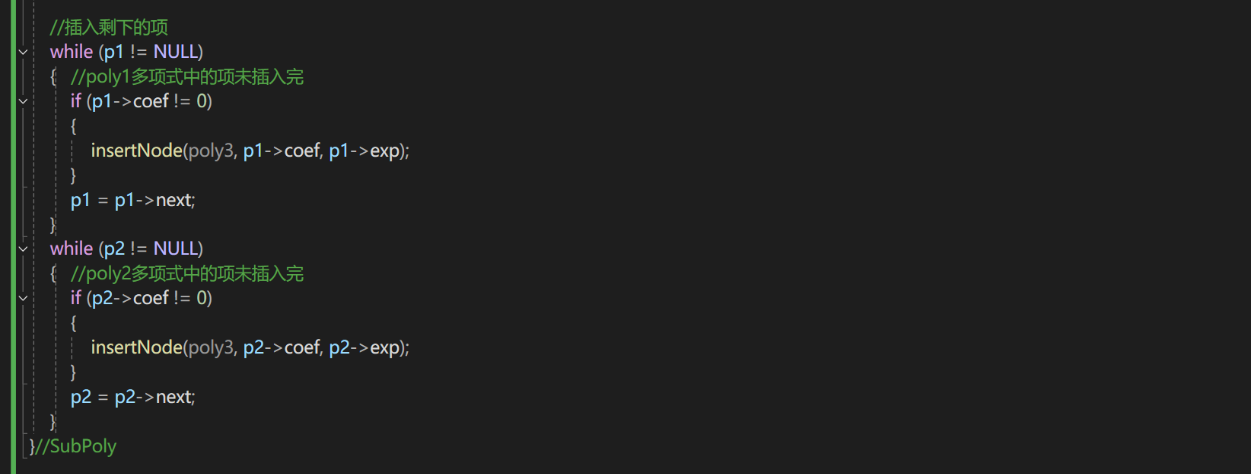
**图5.1加法运算Part1**

****

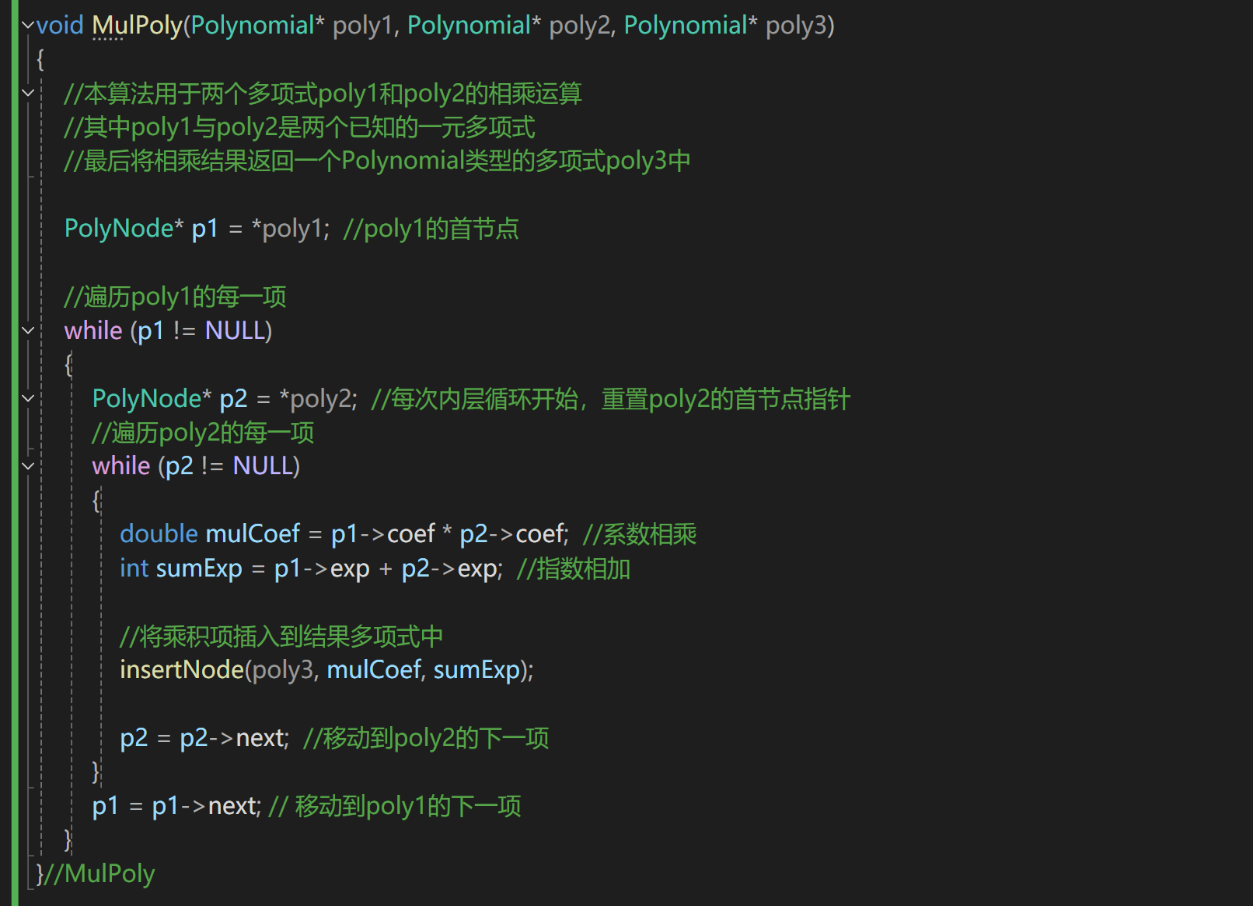
**图5.2加法运算Part2**

****

**图5.3减法运算Part1**

****

**图5.4减法运算Part2**

****

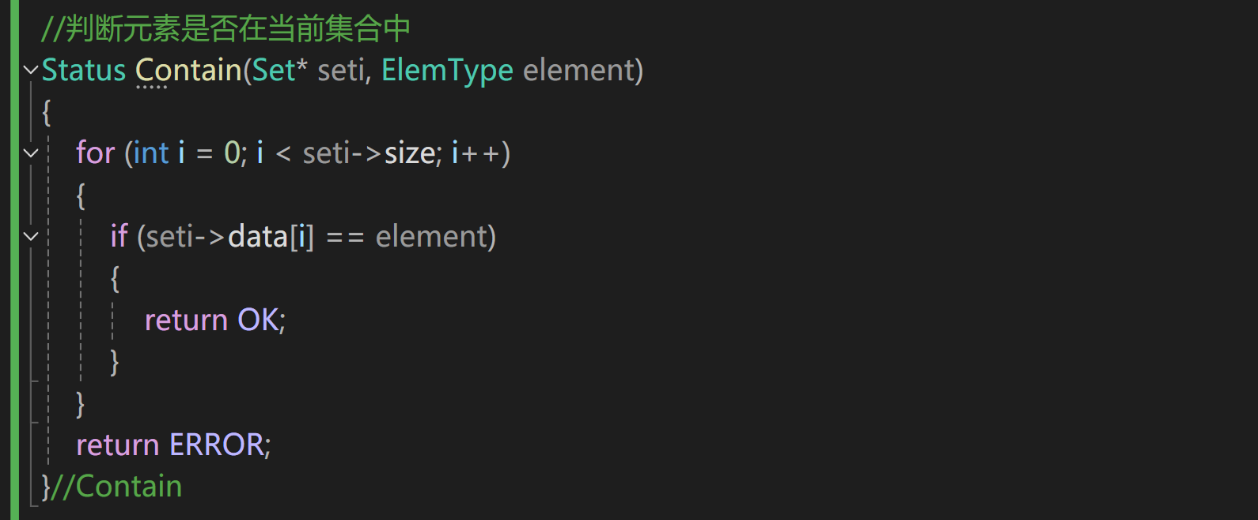
**图5.5乘法运算**

**2.实验内容3：**

根据以上**实验内容3**的算法描述，求解本问题的程序代码如下：

****

**注：此处只展示集合相关运算的程序代码图。**

****

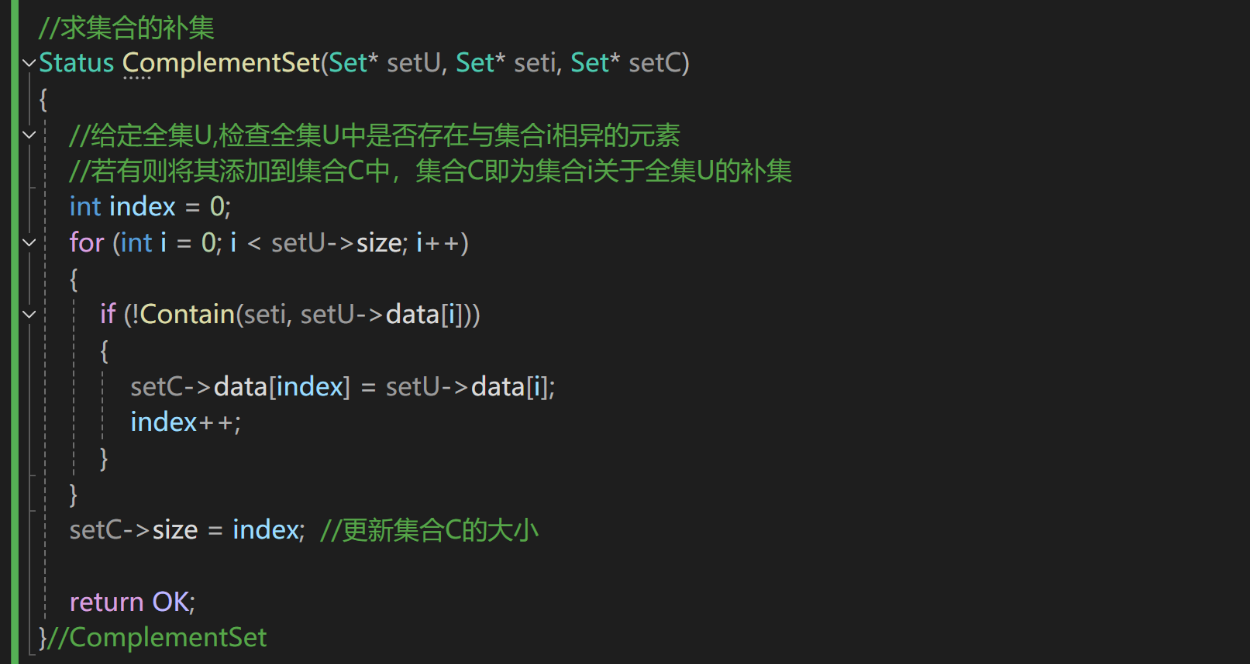
**图5.6 Contain函数**

****

**图5.7求并集**

****

**图5.8求交集**

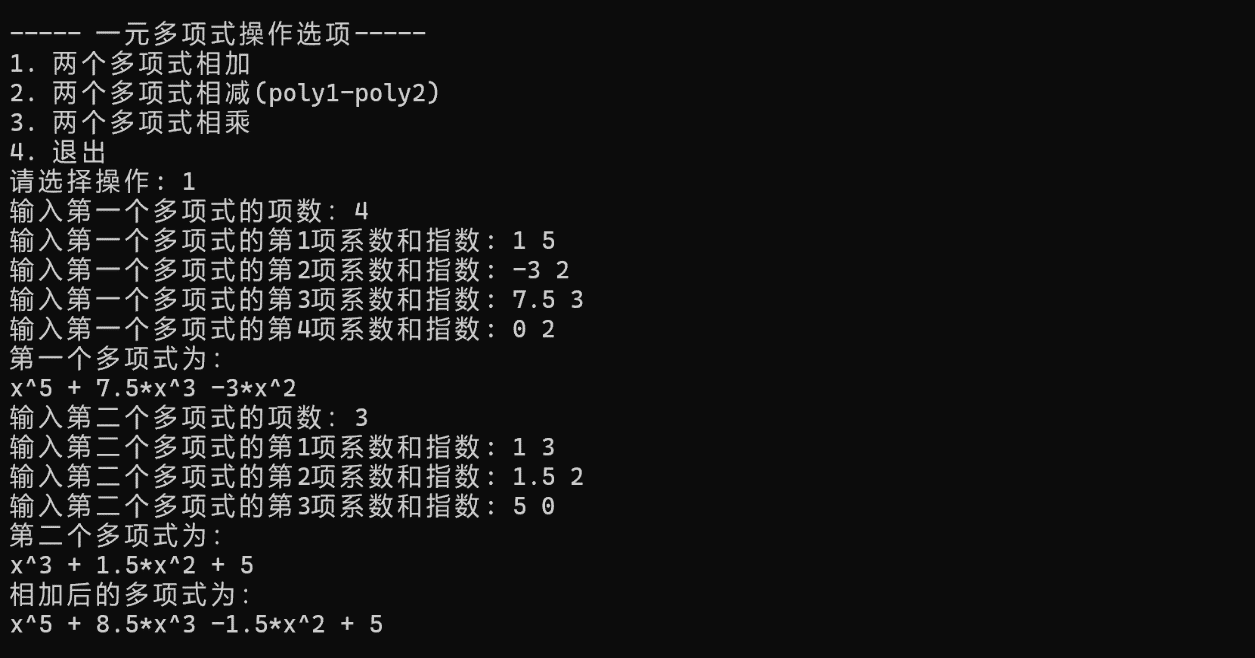
****

**图5.9求补集**

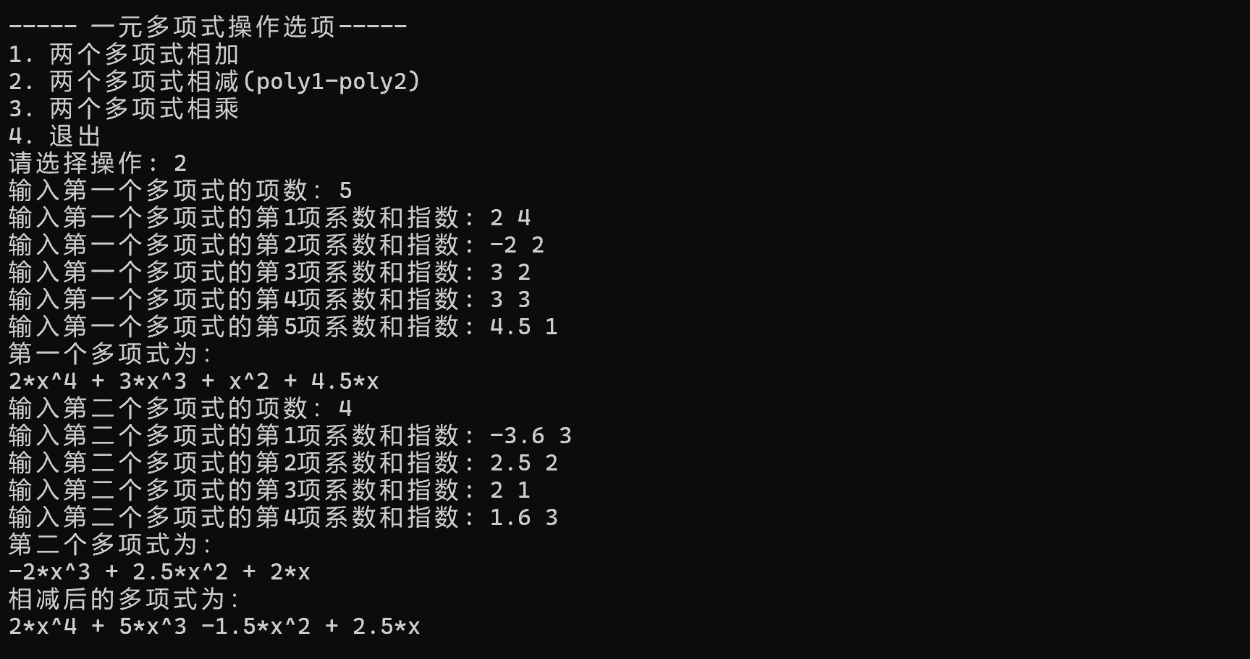
**六、结果分析**

**（一）结果呈现：**

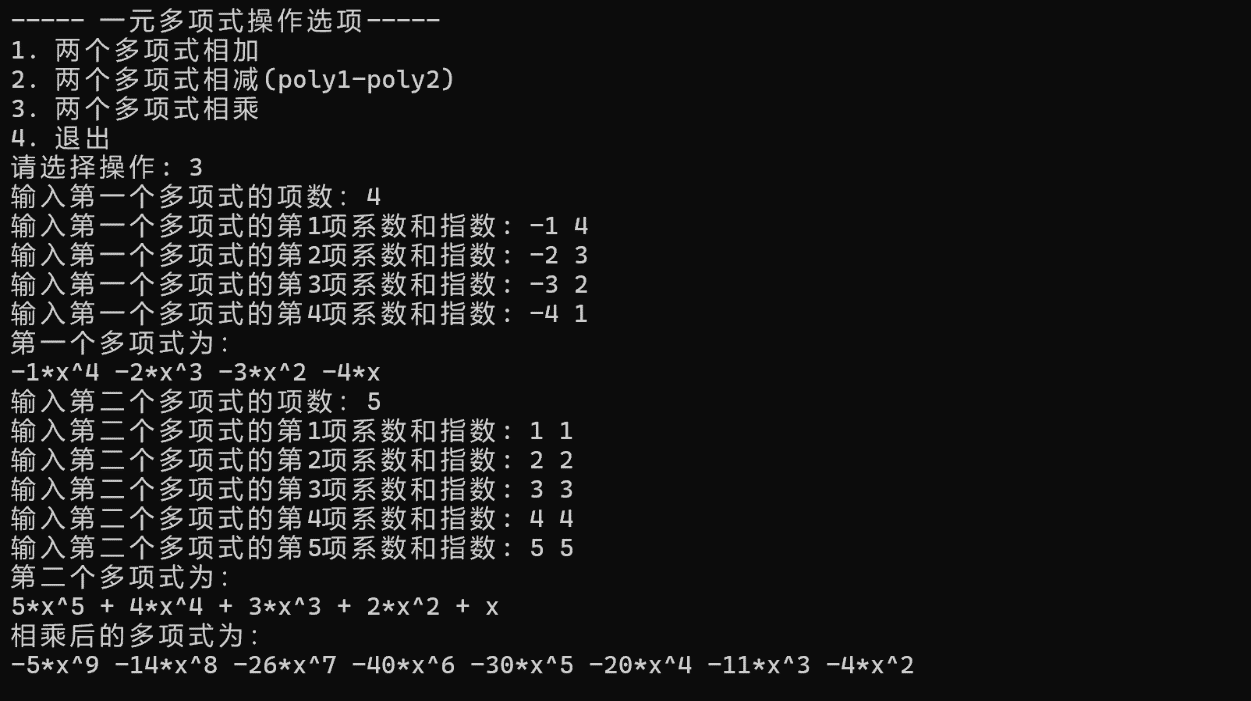
1.根据以上**实验内容1**程序代码运行后进行多次测试，以下是根据**实验内容1的测试用例**进行测试后的结果示意图：

****

**图6.1加法运算测试结果**

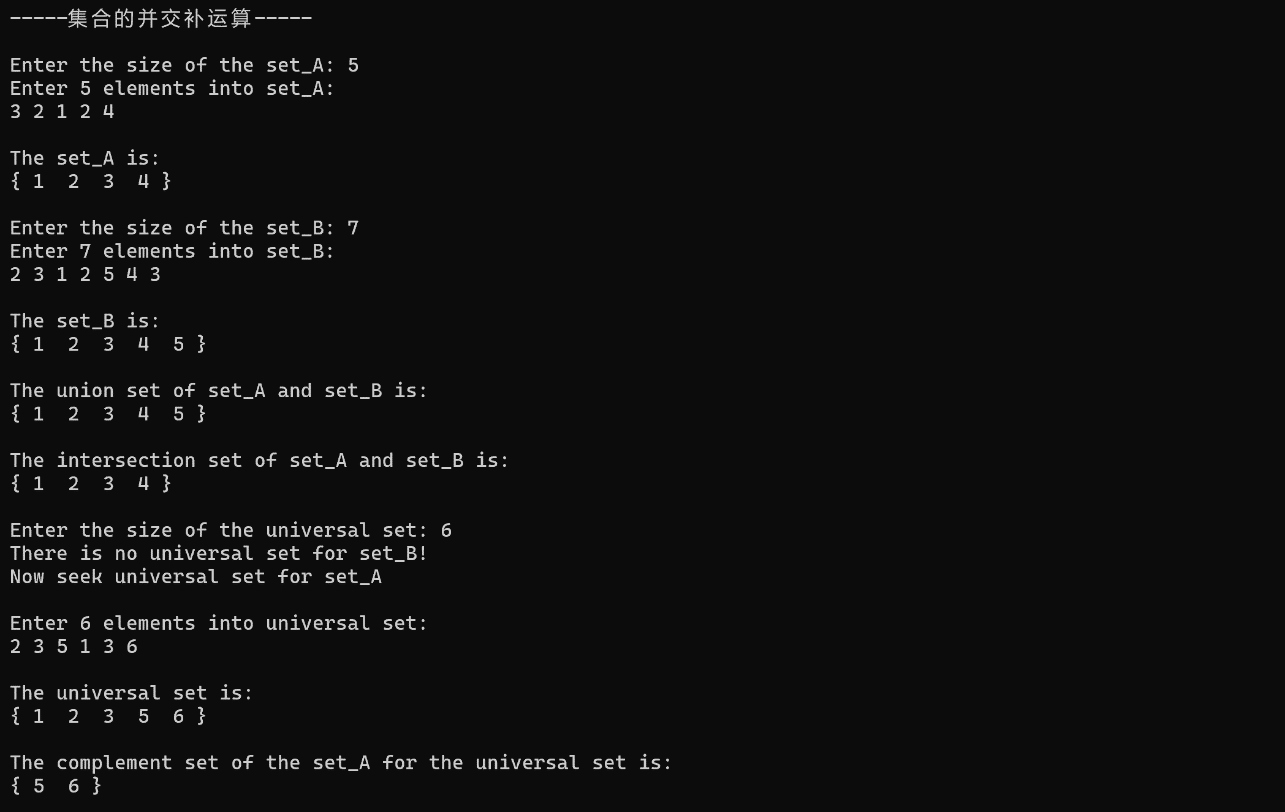
****

**图6.2减法运算结果**

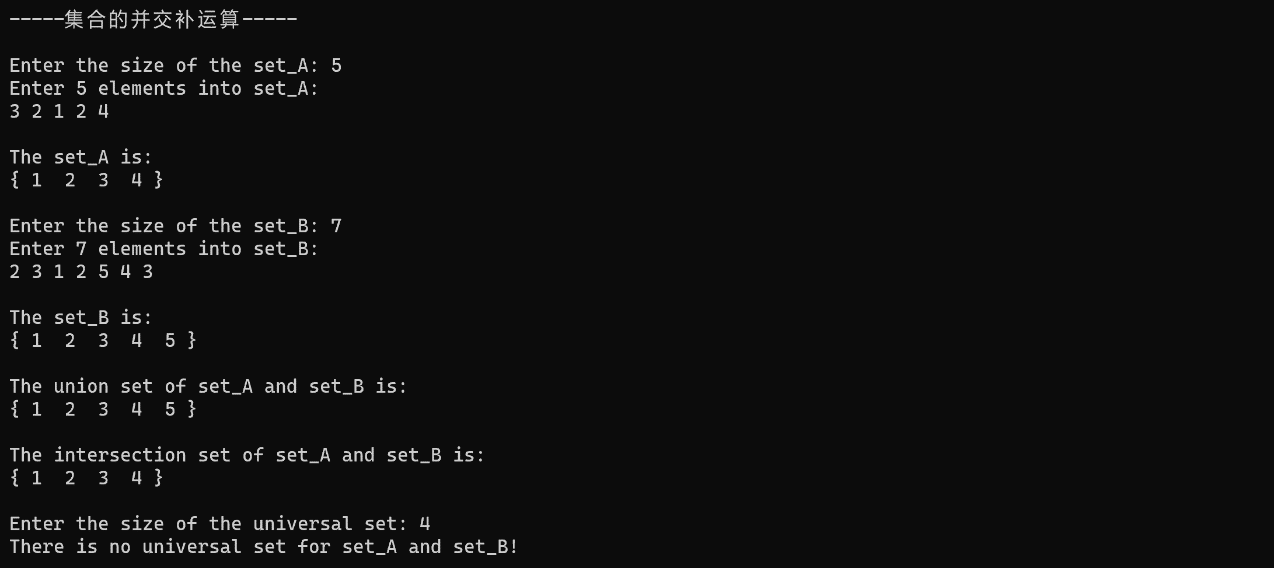
****

**图6.3乘法运算结果**

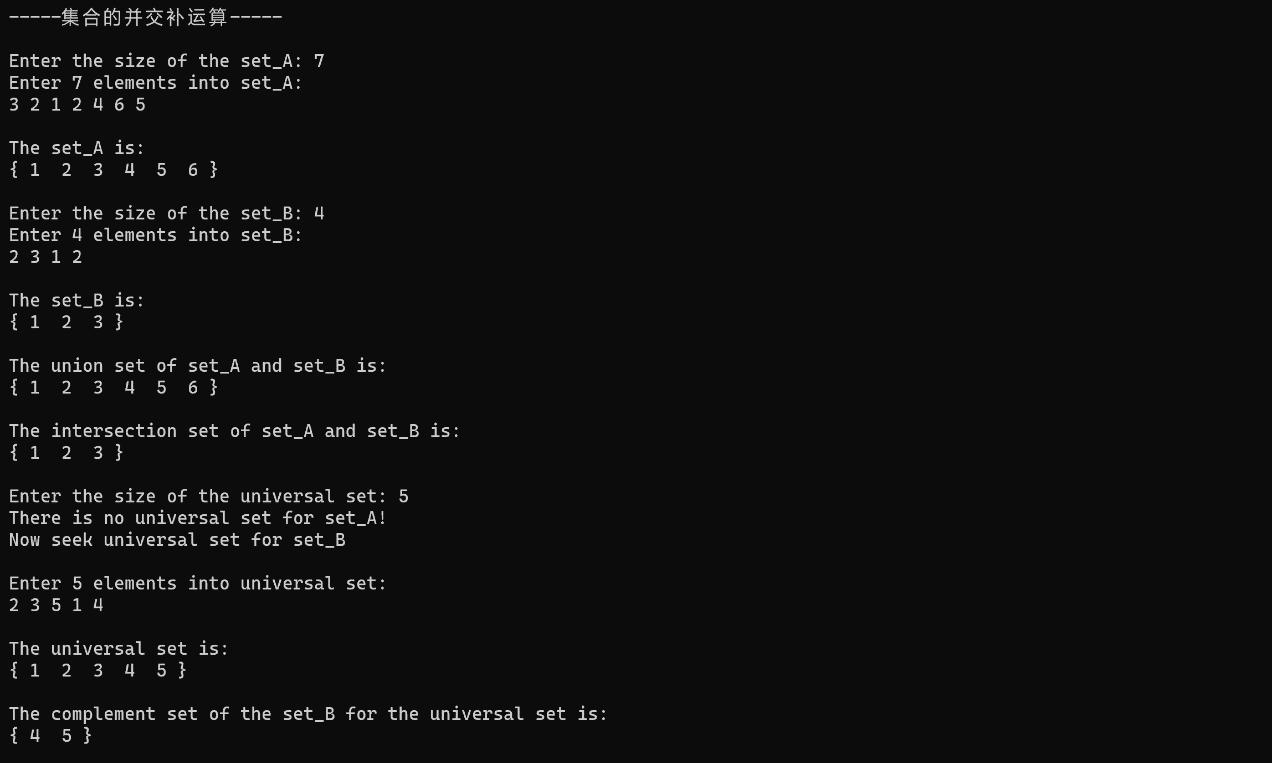
2.根据以上**实验内容3**程序代码运行后进行多次测试，以下是根据**实验内容3的测试用例**进行测试后的结果示意图：

****

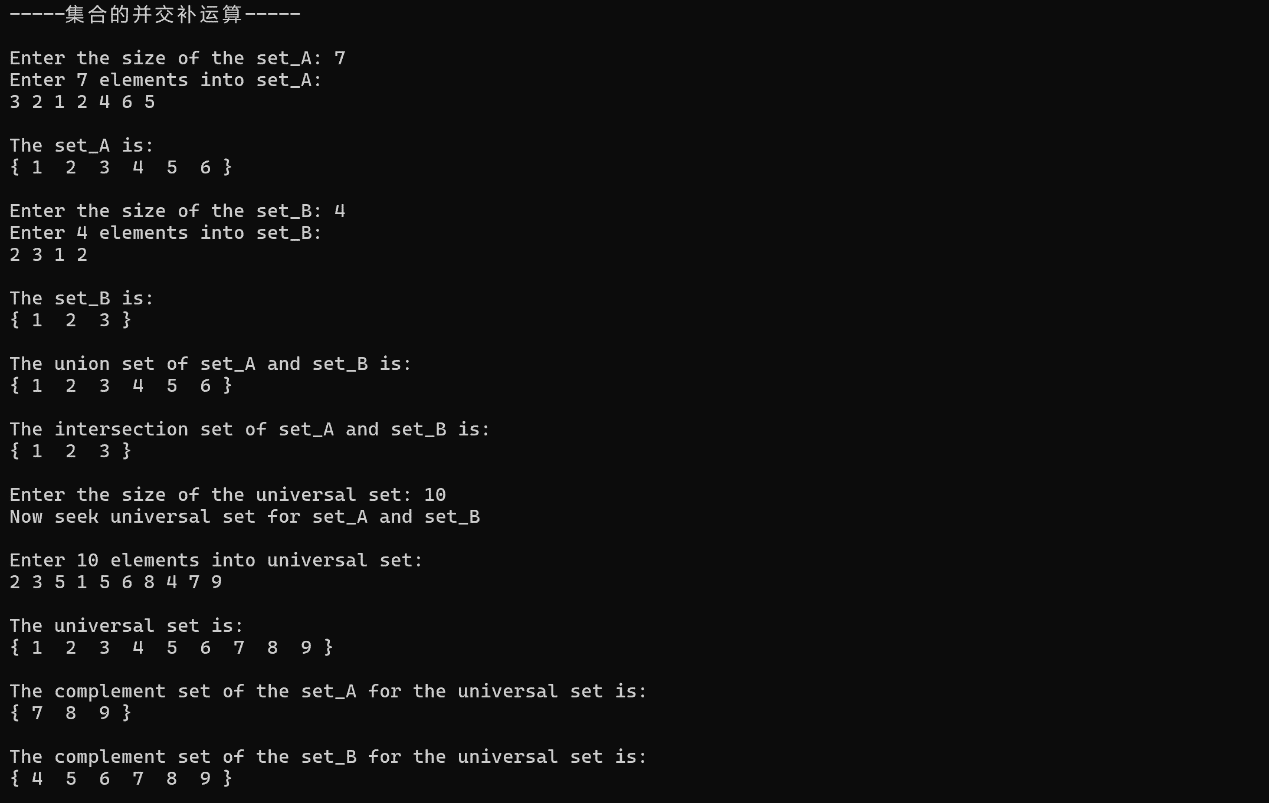
**图6.4 u = 6（全集含有6个元素）**

****

**图6.5 u = 4（全集含有4个元素）**

****

**图6.6 u = 5（全集含有5个元素）**

****

**图6.7 u = 10（全集含有10个元素）**

**（二）结果分析：**

**1.时间复杂度分析**

**（1）实验内容1：**

**【insertNode】—— 插入节点**

该函数在最坏情况下需要遍历整个链表来找到插入位置，故时间复杂度为**O(n)**，其中**n**是当前多项式的项数。

**【AddPoly】—— 相加 ||【SubPoly】—— 相减**

在该函数中，每个多项式的每个项都需要被访问一次,故时间复杂度为**O(m+n)**，其中**m**和**n**分别是两个多项式的项数。

**【MulPoly】—— 相乘**

在该函数中，每个多项式的每个项都需要与另一个多项式的每个项相乘，故时间复杂度为**O(m\*n)**，其中**m**和**n**分别是两个多项式的项数。

**【printPoly】—— 输出多项式表达式**

该函数需遍历整个链表打印每个项，故时间复杂度为**O(n)**，其中**n**是多项式的项数。

**【freePoly】—— 释放链表内存**

同样需要遍历整个链表，时间复杂度为**O(n)**。

**综上所述：在执行一次多项式操作（加、减、乘）的过程中，最耗时的操作是乘法运算，其时间复杂度为 O(m\*n)。因此，总体时间复杂度主要由最高复杂度的乘法操作决定。**

**（2）实验内容3：**

**【InitSet】—— 集合初始化**

分配内存的操作是常量时间，故时间复杂度为**O(1)**。

【**PrintSet**】**—— 集合元素打印**

该函数内部包含两个循环：

第一个用于排序，时间复杂度为**O(n2)**（此处采用冒泡排序）。

第二个用于检查是否存在重复元素并打印不重复的集合元素，时间复杂度为**O(n)**。

整体时间复杂度为**O(n2)**。

【**Contain**】—— **判断元素是否在集合中**

该函数需要遍历集合，故时间复杂度为**O(n)**。

【**UnionSet**】**—— 求并集**

首先添加集合**A**的元素到集合**C**中，时间复杂度为**O(na)**，接着检查集合**B**中的元素是否存在于集合**C**中，每次调用**Contain**的时间复杂度**O(nc)**，因此总时间复杂度为**O(na+nb\*nc)**，其中**na**，**nb，nc**分别是集合**A**，集合**B**，集合**C**的大小。

【**IntersectionSet**】**—— 求交集**

该函数需要遍历集合**B**并在集合**A**中进行查找是否存在相同的元素，故时间复杂度为**O(na\*nb)**，其中**na**和**nb**分别是集合A和集合B的大小。

【**ComplementSet**】**—— 求补集**

该函数需要遍历集合**i**并在全集**U**中进行查找是否存在与集合i中相异的元素，故时间复杂度为**O(nu\*ni)**，其中**nu**是全集的大小，**ni**是需要求补集的集合的大小。

**2.空间复杂度分析**

**（1）实验内容1：**

**【多项式链表】**

每个多项式的存储表示为一个链表，空间复杂度为**O(m)**或**O(n)**，其中，**m**和**n**分别对应两个多项式的项数。

**【存储结果多项式】**

结果多项式的存储会占用**O(m+n)**的空间，因为相加、相减或相乘后可能会产生一个项数最多为**m+n**的新多项式。

**综上所述：总体空间复杂度为 O(m+n)，其中，m和n分别是两个输入的多项式项数。**

**（2）实验内容3：**

**【集合结构体】**

分配的空间为**O(m)**，其中**m**是集合中元素的最大数量，主要体现在**data**指针的动态内存分配。

**【额外空间】**

在主函数中所创建的集合**C**、**D**、**E**、**F**和**U**，每个都占用**O(m)**的空间，其中**m**是集合中元素的最大数量。

**七、感悟体会**

1.**实验内容1**在输入多项式系数和指数时，应该**检查用户的输入是否有效**。例如，确保系数和指数是**数值型**的，指数是**非负整数**。

2.**实验内容3**的整体程序对于大量数据输入可能会很慢，由此可以考虑将**冒泡排序**替换为更高效的排序算法，例如**快速排序**或**归并排序**；同时也可以优化集合操作，例如使用**哈希表**以减少查找时间。