链表实验题

【实验目的】

1. 熟练掌握线性表的基本操作在链式存储结构上的实现；
2. 会用线性链表解决实际问题。

【实验要求】

第 1、2、3 题单链表类型定义如下:

```
typedef int ElemType;
typedef    struct    Node{
      ElemType        data;        // 数据域
      struct    Node    *next;      // 指针域
}LNode, *LinkList;
```

【实验内容】

1. 题目内容：

编程实现单链表的下列基本操作函数。

(1)void CreatList(LinkList &L, int n):

依次输入 n 个数据，并依次建立各个元素结点，逐个插入到链表尾；建立带表头结点的单链表 L;

(2) void ListPrint(LinkList L): 将单链表 L 的数据元素从表头到表尾依次显示。

(3) bool ListInsert (LinkList &L, int i,ElemType e):

在带头结点的单链表 L 中第 i 个数据元素之前插入数据元素 e，并返回 true。如果 i 值非法，则显示“Insertion Location Error”，并返回 false;

(4) bool ListDelete(LinkList &L, int n, ElemType &e) //删除链表的第 n 个元素，用 e 返回其值，并返回 true。如果 i 值非法，则显示“Deletion Location Error”，并返回 false;

(5) int ListLength(LinkList L);//求链表 L 的表长

输入格式：

四行数据，第一行的整数表示要建立的单链表的元素个数，第二行整数表示依次输入的数据，两个整数之间以空格分隔，第三行的两个整数分别表示要插入的数和插入位置，第四行表示要删除数的位置。

输出格式：

三行数据，第一行整数表示已删除的数，第二行表示链表的长，第三行表示目前链表的数据元素(依次从表头至表尾)。

输入样例：
5
4 3 6 7 9
3 20
2
输出样例：
3
5
4 20 6 7 9

a.正常情况

输入样例：
4
4 3 6 7
5 20
6
输出样例：
Deletion Location Error
5
4 3 6 7 20

b.删除位置不合法情况

输入样例：
3
4 3 6
5 20
2
输出样例：
Insertion Location Error
3
2
4 6

c.插入位置不合法情况

2. 题目内容：

设有链表 A 和 B，其中的数据内容均为整型数值，且都按增序排列，生成新的链表 C，满足下列要求：$C = A \cup B$ (C 中元素为 A 和 B 的并集)，且 C 中元素按增序排列。

输入格式：

二行数据，第一行的整数表示 A 表的数据元素，第二行整数 B 表的数据元素，两个整数之间以空格分隔，每一行-1 表示数据的结尾（-1 不属于这个表中的数据元素）。

输出格式：

一行整数，从表头开始依次显示链表 C 的数据元素，数字间用空格分开。

输入样例：
1 3 5 -1
2 4 6 8 10 -1
输出样例：
1 2 3 4 5 6 8 10

输入样例：
1 2 5 6 -1
2 4 6 11 -1
输出样例：
1 2 4 5 6 11

3. 设有链表 A 和 B，其中的数据内容均为整型数值，且都按增序排列，生成新的链表 C，满足下列要求：$C = A \cap B$ (C 中元素为 A 和 B 中都出现的元素) ，且 C 中元素按增序排列。

输入格式：

二行数据，第一行的整数表示 A 表的数据元素，第二行整数 B 表的数据元素，两个整数之间以空格分隔，每一行-1 表示数据的结尾（-1 不属于这个表中的数据元素）。

输出格式：

一行整数，从表头开始依次显示链表 C 的数据元素，数字间用空格分开。

输入样例：
1 2 5 -1
2 4 5 8 10 -1
输出样例：
2 5

4. 约瑟夫(Joseph)问题

[问题描述]

约瑟夫(Joseph)问题的一种描述是：编号为 1，2，…,n 的 n 个人按顺时针方向围坐一圈，每人持有一个密码（正整数），一开始任选一个整数作为报数上限 m,从第一人开始按顺时针方向从自 1 开始顺序报数，报到 m 时停止报数。报 m 的人出列，将他的密码作为新的 m 值，从他的顺时针方向上的下一个人开始重新从 1 报数，如此下去，直至所有人全部出列为止，设计一个程序求出出列顺序。

[基本要求]

采用单向循环链表模拟此过程,按照出列的顺序打印出各人的编号

[实验提示]

单向循环链表的结点类型定义参照如下：

```
typedef struct LNode
{
    int code;     //定义整型变量 code 用来存放序号
    int key;      //定义整型变量 key 用来存放密码
    struct LNode *next;
}LNode, *LinkList;
```

[测试数据]

m 的初值为 6；n=7,7 个人的密码依次为：3，1，7，2，4，8，4，正确的出列顺序应为 6,1,4,7,2,3,5。

输入格式：

二行数据，第一行的两个整数分别为 m 和 n 的值，第二行整数 n 个人的密码，两个整数之间以空格分隔。

输出格式：

表示出列顺序，两个整数之间以空格分隔。

输入样例：
6 7
3 1 7 2 4 8 4
输出样例：
6 1 4 7 2 3 5

输入样例：
1 3
2 4 1
输出样例：
1 3 2

5. 一元多项式的相加。

利用两个带头结点的单链表 La 和 Lb 分别存储两个一元多项式 A(x)和 B(x)，对这两个一元多项式求和，输出结果。

输入格式：

二行数据，每行分别先给出多项式非零项的个数，再以指数递降方式输入一个多项式非零项系数和指数，数字间以空格分隔。

输出格式：

以指数递降方式输出和多项式非零项的系数和指数，数字间以空格分隔。

输入样例：

4 3 4 -5 2 6 1 -2 0

3 5 20 -7 4 3 1

输出样例：

5 20 -4 4 -5 2 9 1 -2 0

输入样例的两个多项式如下：

$A(x)=3x^4-5x^2+6x-2$

$B(x)=5x^{20}-7x^4+3x$