# 实验1

#### 1 实验题目

题目一：**求顺序表中第i个元素（函数），若不存在，报错。**

* + **第一组数据：顺序表长度n≥10，i分别为5，n，0，n+1，n+2**
  + **第二组数据：顺序表长度n=0，i分别为0，2**

题目二：**在第i个结点前插入值为x的结点。**

* + **第一组数据：顺序表长度n≥10，x=100, i分别为5,n,n+1,0,1,n+2**
  + **第二组数据：顺序表长度n=0，x=100，i=5**

题目三：**删除顺序表中第i个元素结点。**

* + **第一组数据：顺序表长度n≥10，i分别为5,n,1,n+1,0**
  + **第二组数据：顺序表长度n=0， i=5**

题目四：**在一个递增有序的顺序表L中插入一个值为x的元素，并保持其递增有序特性。**

* + **顺序表元素为 （10,20,30,40,50,60,70,80,90,100）,**

**x分别为25，85，110和8**

**题目五：将顺序表Ｌ中的奇数项和偶数项结点分解开（元素值为奇数、偶数），分别放入新的顺序表中，然后原表和新表元素同时输出到屏幕上，以便对照求解结果。**

* + **第一组数据：顺序表元素为 （1,2,3,4,5,6,7,8,9,10,20,30,40,50,60）**
  + **第二组数据：顺序表元素为 （10,20,30,40,50,60,70,80,90,100）**

题目六：**求两个递增有序顺序表L1和L2中的公共元素，放入新的顺序表L3中。**

* **第一组**
  + **第一个顺序表元素为 （1，3，6，10，15，16，17，18，19，20）**
  + **第二个顺序表元素为 （1，2，3，4，5，6，7，8，9，10，18，20，30）**
* **第二组**
  + **第一个顺序表元素为 （1，3，6，10，15，16，17，18，19，20）**
  + **第二个顺序表元素为 （2，4，5，7，8，9，12，22）**
* **第三组**
  + **第一个顺序表元素为 （）**
  + **第二个顺序表元素为 （1，2，3，4，5，6，7，8，9，10）**

题目七：**删除递增有序顺序表中的重复元素，并统计移动元素次数，要求时间性能最好。**

* + **第一组数据：顺序表元素为 （1,2,3,4,5,6,7,8,9）**
  + **第二组数据：顺序表元素为 （1,1,2,2,2,3,4,5,5,5,6,6,7,7,8,8,9）**

**第三组数据：顺序表元素为 （1,2,3,4,5,5,6,7,8,8,9,9,9,9,9**

**拓展题：非必做内容，有兴趣的同学选做，**

**<1> 递增有序顺序表L1、L2，对2表进行合并，并使得合并后成为一个集合，集合的元素放回L1表中保存，要求时间性能最好。**

**<2>（递增有序）顺序表表示集合A、B，实现：**

* + **C=A∩B，C=A∪B，C=A-B**
  + **A=A∩B，A=A∪B，A=A-B**

**<3>（递增有序）顺序表表示集合A、B，判定A是否B的子集。**

**<4>一个长度为L（L≥1）的升序序列S，处在第L/2个位置的数称为S 的中位数。例如，若序列S1=(11, 13, 15, 17, 19)，则S1 的中位数是15。两个序列的中位数是含它们所有元素的升序序列的中位数。例如，若S2=(2, 4, 6, 8, 20)，则S1 和S2 的中位数是11。**

**现有两个等长升序序列A 和B，试设计一个在时间和空间两方面都尽可能高效的算法，找出两个序列A 和B 的中位数。要求：**

**（1）给出算法的基本设计思想。**

**（2）根据设计思想，采用C 或C++语言描述算法，关键之处给出注释。**

**（3）说明你所设计算法的时间复杂度和空间复杂度。**

#### 2 实验源码

#pragma once

class seqList

{

public:

seqList();

~seqList();

void setuplist(int a[],int);

void GetNum(int);

void InNum(int, int);

void DlNum(int);

void PrintList();

bool isFull();

bool isEmpty();

void Inlist(int);

void Numsort();

void hebing(seqList\*m,seqList\*n);

void DLcfNum();

void tuozhan1(seqList\*L1,seqList\*L2);

void bingji(seqList\*A,seqList\*B);

void chaji(seqList\*A,seqList\*B);

bool judgeziji(seqList\*A,seqList\*B);

private:

int alist[100];

int length;

};

#include "seqList.h"

#include <iostream>

using namespace *std*;

seqList::seqList()

{

length=0;

}

seqList::~seqList()

{

}

void seqList::setuplist(int a[],int n)

{

if (n > 100)

*cout* << "操作错误" << *endl*;

else

{

length = n;

for (int i = 0; i < n; i++)

{

alist[i] = a[i];

}

}

}

bool seqList::isEmpty()

{

if (length == 0)

return true;

else

return false;

}

bool seqList::isFull()

{

if (length==100)

return true;

else

return false;

}

/\*

求顺序表中第i个元素（函数），若不存在，报错。

\*/

void seqList::GetNum(int i)

{

if (isEmpty()||i < 1 || i > length)

{

*cout* << "操作错误" << *endl*;

}

else

{

*cout*<<"第"<<i<<"个元素为"<< alist[i-1]<<*endl*;

}

}

/\*

在第i个结点前插入值为x的结点

\*/

void seqList::InNum(int i, int x)

{

if (length > 100 || i < 1 || i > length + 1)

{

*cout* << "操作错误" << *endl*;

}

else

{

for (int j = length; j >= i; j--)

{

alist[j] = alist[j - 1];

}

alist[i - 1] = x;

length++;

}

}

/\*

删除顺序表中第i个元素结点

\*/

void seqList::DlNum(int i)

{

if (length == 0 || i < 1 || i > length)

{

*cout* << "操作错误" << *endl*;

}

else

{

int x = alist[i - 1];

for (int j = i; j < length; j++)

alist[j - 1] = alist[j];

length--;

}

}

/\*

在一个递增有序的顺序表L中插入一个值为x的元素，并保持其递增有序特性

\*/

void seqList::Inlist(int x)

{

int i;

if (isFull())

{

*cout* << "操作错误" << *endl*;

}

else

for ( i=length-1;alist[i]>=x;i--)

{

alist[i + 1] = alist[i];

}

alist[i+1] = x;

length++;

}

/\*

将顺序表Ｌ中的奇数项和偶数项结点分解开（元素值为奇数、偶数），分别放入新的顺序表中

然后原表和新表元素同时输出到屏幕上，以便对照求解结果。

\*/

void seqList::Numsort()

{

if (isEmpty())

{

*cout* << "操作错误" << *endl*;

}

seqList A,B;

int i;

int j = 0, k = 0;

for (i=0;i<length;i++)

{

if (alist[i]%2==0)

{

A.alist[j] = alist[i];

j++;

A.length++;

}

else

{

B.alist[k] = alist[i];

k++;

B.length++;

}

}

*cout* << "偶数数表";

for (int i=0;i<A.length;i++)

{

*cout* << A.alist[i]<<" ";

}

*cout* << *endl*;

*cout* << "奇数数表";

for (int i = 0; i < B.length; i++)

{

*cout*<< B.alist[i]<<" ";

}

*cout* << *endl*;

}

/\*

求两个递增有序顺序表L1和L2中的公共元素，放入新的顺序表L3中

\*/

void seqList::hebing(seqList\*m,seqList\*n)

{

seqList C;

int k= 0;

if (m->length==0||n->length==0)

{

*cout* << "操作错误" << *endl*;

}

else

{

for (int i=0;i<m->length;i++)

{

for (int j=0;j<n->length;j++ )

if (m->alist[i]==n->alist[j])

{

C.alist[k] = m->alist[i];

k++;

}

}

*cout* << "合并后的数表" << *endl*;;

for (int i=0;i<k;i++)

{

*cout* << C.alist[i]<<" ";

}

*cout* << *endl*;

}

}

/\*

删除递增有序顺序表中的重复元素，并统计移动元素次数，要求时间性能最好

\*/

void seqList::DLcfNum()

{

int i = 0,j = 1, k = 0;

while (j !=length)

{

if (alist[i] ==alist[j])

{

j++;

}

else

{

alist[i + 1] =alist[j];

i ++;

k++;

}

}

length = i + 1;

*cout*<<"移动次数为" << k;

*cout* << "删除重复后的数表为" << *endl*;

for (i=0;i<length;i++)

{

*cout* << alist[i] << " ";

}

*cout* << *endl*;

}

/\*

拓展题1

<1> 递增有序顺序表L1、L2，对2表进行合并，并使得合并后成为一个集合，集合的元素放回L1表中保存，要求时间性能最好。

\*/

void seqList::tuozhan1(seqList\*L1,seqList\*L2)

{

int i = L1->length - 1, j = L2->length - 1;

while (i >= 0 && j >= 0)

{

if (L1->alist[i] <= L2->alist[j])

{

if (!L1->alist[i + 1])

{

L1->alist[i + 1] = L2->alist[j];

}

else

{

for (int k = L1->length - 1; k >= i + 1; --k)

{

L1->alist[k + 1] = L1->alist[k];

}

L1->alist[i + 1] = L2->alist[j];

}

L1->length++;

j--;

}

else

{

i--;

}

}

for (int i = 0; i < L1->length; i++)

{

*cout* << L1->alist[i]<<" ";

}

}

/\*

（递增有序）顺序表表示集合A、B，实现：

 C=AB，C=AB，C=A-B

 A=AB，A=AB，A=A-B

\*/

void seqList::bingji(seqList\* A, seqList\* B)

{

seqList C;

int i, j, k = 0;

for (i = 0; i < A->length; i++)

C.alist[i] = A->alist[i];

C.length = A->length;

for (i = 0; i < B->length; i++)

{

j = 0;

while (j < A->length && B->alist[i] != A->alist[j])

j++;

if (j == A->length)

C.alist[C.length + k++] = B->alist[i];

}

C.length += k;

for (int x=0;x<C.length;x++)

{

*cout* << C.alist[x] << " ";

}

}

/\*

\*/

void seqList::chaji(seqList\* A, seqList\* B )

{

seqList C;

int i, j, k = 0;

for (i = 0; i < A->length; i++)

{

j = 0;

while (j < B->length && B->alist[j] != A->alist[i])

j++;

if (j == B->length) //表示A->data[i]不在B中，将其放到C中

C.alist[k++] = A->alist[i];

}

C.length = k;

for (int a=0;a<C.length;a++)

{

*cout* << C.alist[a]<<" ";

}

}

/\*

<3>（递增有序）顺序表表示集合A、B，判定A是否B的子集。

\*/

bool seqList::judgeziji(seqList\*A,seqList\*B)

{

int a=0, b=0;

while (a < A->length && b < B->length)

{

if (A->alist[a] == B->alist[b])

{

a++;

b++;

}

else if (A->alist[a] >B->alist[b])//A当前元素大于B当前，

{ //就还得判断B后面的

b++;

}

else //如果相等，则继续判断A下一个元素

{

return false;

}

}

}

/\*

打印数表

\*/

void seqList::PrintList()

{

for (int i = 0; i < length; i++)

{

*cout* << alist[i] << ' ';

}

*cout* << *endl*;

}

// list.cpp : 此文件包含 "main" 函数。程序执行将在此处开始并结束。

//

#include <iostream>

#include <string>

#include "seqList.h"

using namespace *std*;

int main()

{

/\*

测试1

\*/

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

*cout* << "测试1" << *endl*;

seqList tst1,tst1\_1;

int a1[11] = { 1,5,6,9,7,5,7,4,3,0,8 };

tst1.setuplist(a1, 11);

tst1.PrintList();

tst1.GetNum(5);

tst1.GetNum(11);

tst1.GetNum(0);

tst1.GetNum(12);

tst1.GetNum(13);

tst1\_1.GetNum(0);

tst1\_1.GetNum(2);

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

/\*

测试2

\*/

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

*cout* << "测试2" << *endl*;

seqList tst2,tst2\_2;

tst2.setuplist(a1, 11);

*cout* << "插入前";

tst2.PrintList();

tst2.InNum(5, 100);

*cout* << "插入后";

tst2.PrintList();

tst2.InNum(11, 100);

*cout* << "插入后";

tst2.PrintList();

tst2.InNum(12, 100);

*cout* << "插入后";

tst2.PrintList();

tst2.InNum(0, 100);

*cout* << "插入后";

tst2.PrintList();

tst2.InNum(1, 100);

*cout* << "插入后";

tst2.PrintList();

tst2.InNum(13, 100);

*cout* << "插入后";

tst2.PrintList();

tst2\_2.InNum(5, 100);

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

/\*

测试3

\*/

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

*cout* << "测试3" << *endl*;

*cout* << "删除前";

tst2.PrintList();

tst2.DlNum(5);

*cout* << "删除后";

tst2.PrintList();

tst2.DlNum(11);

*cout* << "删除后";

tst2.PrintList();

tst2.DlNum(1);

*cout* << "删除后";

tst2.PrintList();

tst2.DlNum(12);

*cout* << "删除后";

tst2.PrintList();

tst2.DlNum(0);

*cout* << "删除后";

tst2.PrintList();

tst2\_2.DlNum(5);

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

/\*

测试4

\*/

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

*cout* << "测试4" << *endl*;

int a4[10] = { 10,20,30,40,50,60,70,80,90,100 };

seqList tst4;

tst4.setuplist(a4, 10);

*cout* << "插入前";

tst4.PrintList();

tst4.Inlist(25);

*cout* << "插入后";

tst4.PrintList();

tst4.Inlist(85);

*cout* << "插入后";

tst4.PrintList();

tst4.Inlist(110);

*cout* << "插入后";

tst4.PrintList();

tst4.Inlist(8);

*cout* << "插入后";

tst4.PrintList();

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

/\*

tst5

\*/

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

*cout* << "测试5" << *endl*;

int a5[15] = { 1,2,3,4,5,6,7,8,9,10,20,30,40,50,60 };

int a5\_2[] = { 10,20,30,40,50,60,70,80,90,100 };

seqList tst5,tst5\_2;

tst5.setuplist(a5, 15);

*cout* << "分组前";

tst5.PrintList();

*cout* << "分组后"<<*endl*;

tst5.Numsort();

tst5\_2.setuplist(a5\_2, 10);

*cout* << "分组前";

tst5\_2.PrintList();

*cout* << "分组后" << *endl*;

tst5\_2.Numsort();

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

/\*

tst6

\*/

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

*cout* << "测试6" << *endl*;

seqList tst6,tst6\_1,tst6\_2,tst6\_3,tst6\_5,tst6\_4;

int t6a1[] = { 1,3,6,10,15,16,17,18,19,20 };

int t6a2[] = { 1,2,3,4,5,6,7,8,9,10,18,20,30 };

tst6\_1.setuplist(t6a1, 10);

tst6\_2.setuplist(t6a2, 13);

*cout* << "合并前的数表" << *endl*;

tst6\_1.PrintList();

tst6\_2.PrintList();

tst6.hebing(&tst6\_1, &tst6\_2);

int t6a3[] = { 2,4,5,7,8,9,12,22 };

tst6\_3.setuplist(t6a3, 8);

*cout* << "合并前的数表" << *endl*;

tst6\_1.PrintList();

tst6\_3.PrintList();

tst6.hebing(&tst6\_1, &tst6\_3);

int t6a4[] = { 1,2,3,4,5,6,7,8,9,10 };

tst6\_4.setuplist(t6a4, 10);

tst6.hebing(&tst6\_4, &tst6\_5);

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

/\*

测试7

\*/

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

*cout* << "测试7" << *endl*;

seqList tst7,tst7\_2,tst7\_3;

int t7a1[] = { 1,2,3,4,5,6,7,8,9 };

int t7a2[] = { 1,1,2,2,2,3,4,5,5,5,6,6,7,7,8,8,9 };

int t7a3[] = { 1,2,3,4,5,5,6,7,8,8,9,9,9,9,9 };

tst7.setuplist(t7a1,9);

*cout* << "删除重复前" << *endl*;

tst7.PrintList();

*cout* << "删除重复后" << *endl*;

tst7.DLcfNum();

tst7\_2.setuplist(t7a2, 17);

*cout* << "删除重复前" << *endl*;

tst7\_2.PrintList();

*cout* << "删除重复后" << *endl*;

tst7\_2.DLcfNum();

tst7\_3.setuplist(t7a3, 15);

*cout* << "删除重复前" << *endl*;

tst7\_3.PrintList();

*cout* << "删除重复后" << *endl*;

tst7\_3.DLcfNum();

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

/\*

拓展题1

\*/

*cout* << "拓展题1" << *endl*;

seqList tstL1,tstL2,tstl3;

int L1a[] = { 1,3,5,7,9,666,55 };

int L2b[] = { 2,4,6,8,10 };

tstL1.setuplist(L1a, 5);

tstL2.setuplist(L2b, 5);

*cout* << "合并前的数表L1为" << *endl*;

tstL1.PrintList();

*cout* << "合并前的数表L2为" << *endl*;

tstL2.PrintList();

*cout* << "L2合并到L1后的数表为" << *endl*;

tstL1.tuozhan1(&tstL1, &tstL2);

*cout* << *endl*;

*cout* <<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<< *endl*;

/\*

拓展题2

\*/

*cout* << "拓展题2" << *endl*;

seqList TZ2A, TZ2B, TZ2C,TZ2D,TZ2E;

int tza[] = { 1,2,3,5,6,10};

int tzb[] = { 1,2,8,7,9,10 };

TZ2A.setuplist(tza, 6);

TZ2B.setuplist(tzb, 6);

*cout* << "数表一为" << *endl*;

TZ2A.PrintList();

*cout* << "数表二为" << *endl*;

TZ2B.PrintList();

*cout* << "并集A∪B"<<*endl*;

TZ2C.bingji(&TZ2A, &TZ2B);

*cout* << *endl*;

*cout* << "差集A-B" << *endl*;

TZ2D.chaji(&TZ2A, &TZ2B);

*cout* << *endl*;

*cout* << "交集A∩B";

TZ2E.hebing(&TZ2A, &TZ2B);

*cout* << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << *endl*;

/\*

拓展题3

\*/

*cout* << "拓展题3" << *endl*;

seqList TZ3A, TZ3B,TZ3C;

int TZ3a[] = { 1,2,3 };

int TZ3b[] = { 4,5,6 };

TZ3A.setuplist(TZ3a, 3);

TZ3B.setuplist(TZ3b, 3);

*cout* << "数表A为" << *endl*;

TZ3A.PrintList();

*cout* << "数表B为" << *endl*;

TZ3B.PrintList();

if (TZ3C.judgeziji(&TZ3A, &TZ3B))

{

*cout* << "A是B的子集" << *endl*;

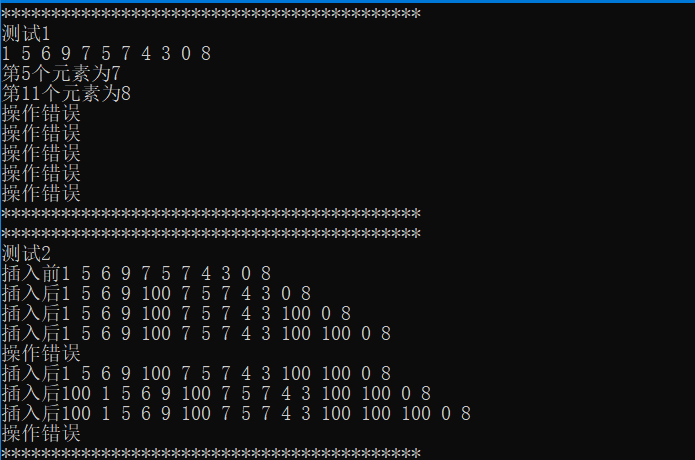
}

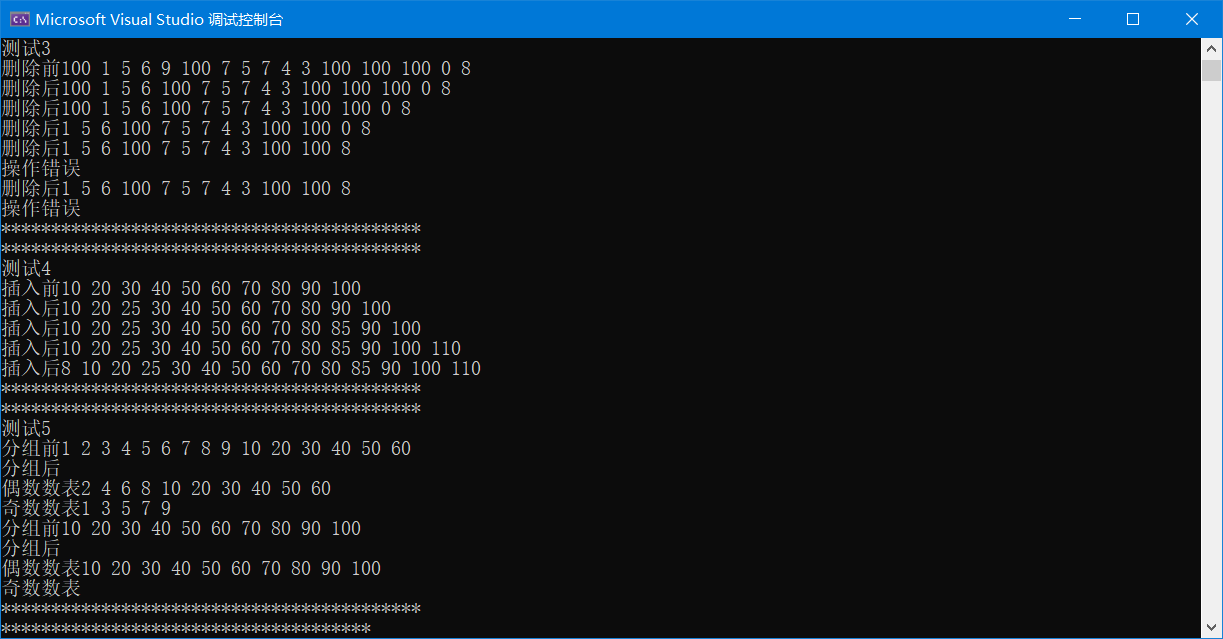
else *cout* << "A不是B的子集" << *endl*;

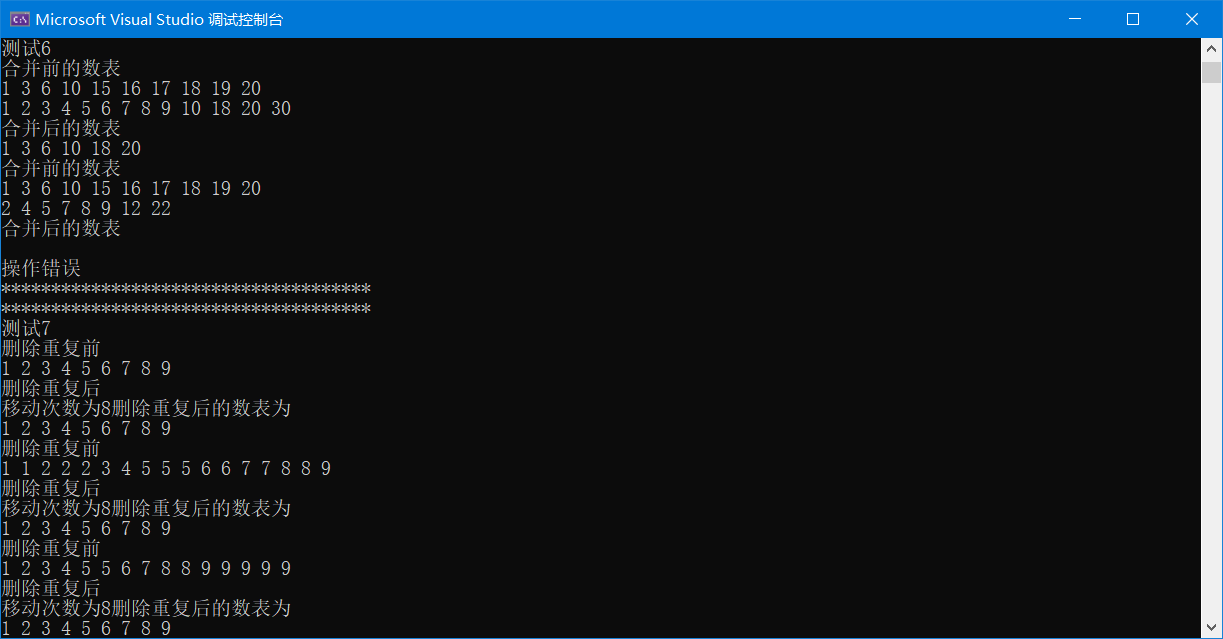
return 0;

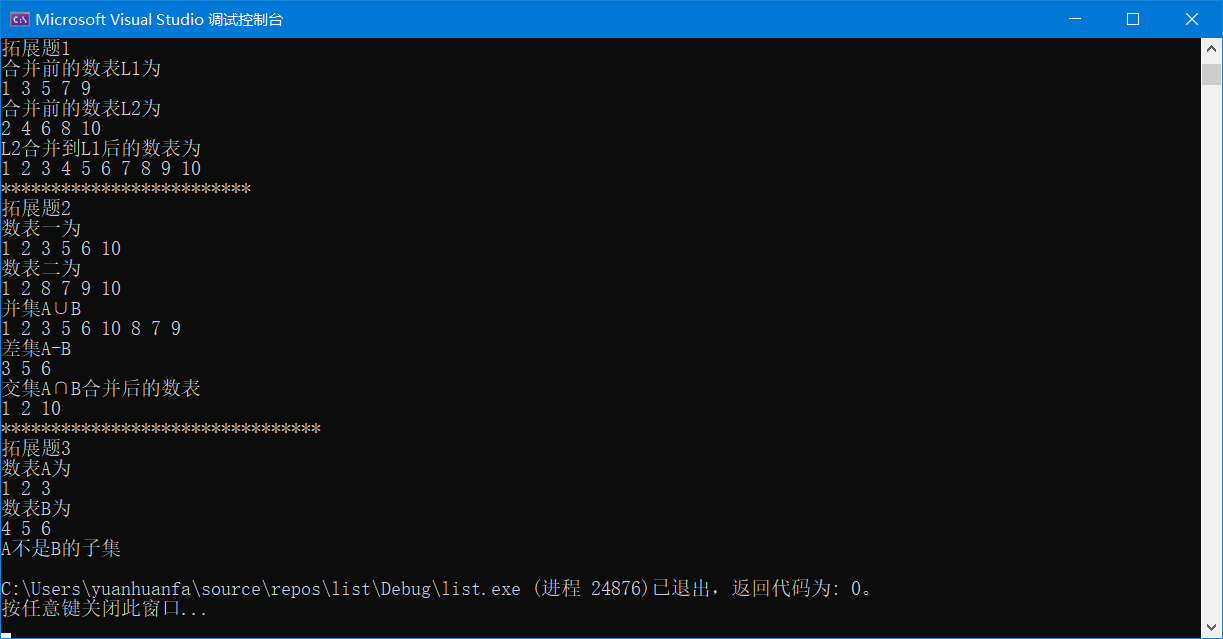
}

#### 3 实验截图









#### 4 心得体会

这次的顺序表实验让我学会了顺序表的增减、删除、合并等基本操作，其中也遇到了各种乱码等bug，特别是在第七问的剔除重复元素合并两个顺序表为一个顺序表时，花费了很长时间，一开始完全不懂问题中的移动次数是指的什么，后来才明白，而且顺序表在插入删除时很容易出现越界等错误，必须要进行判断