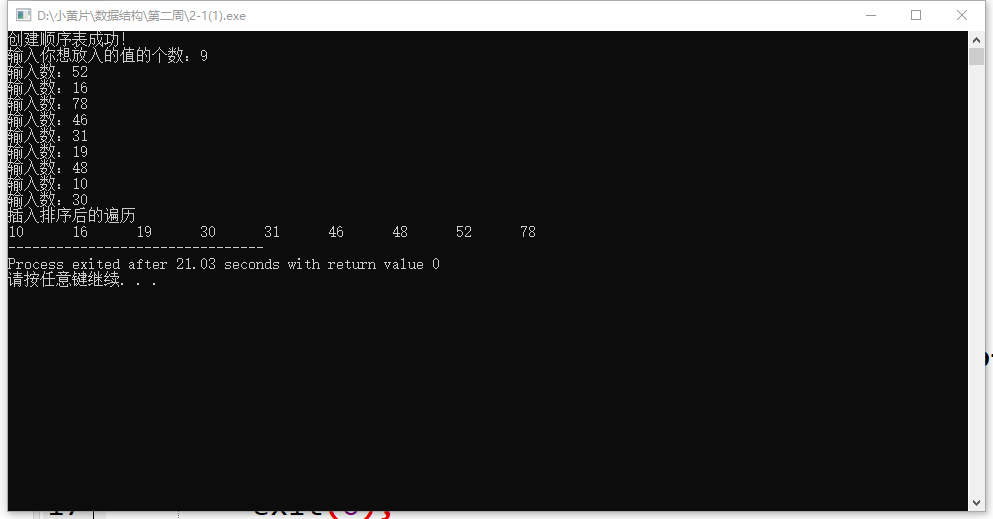
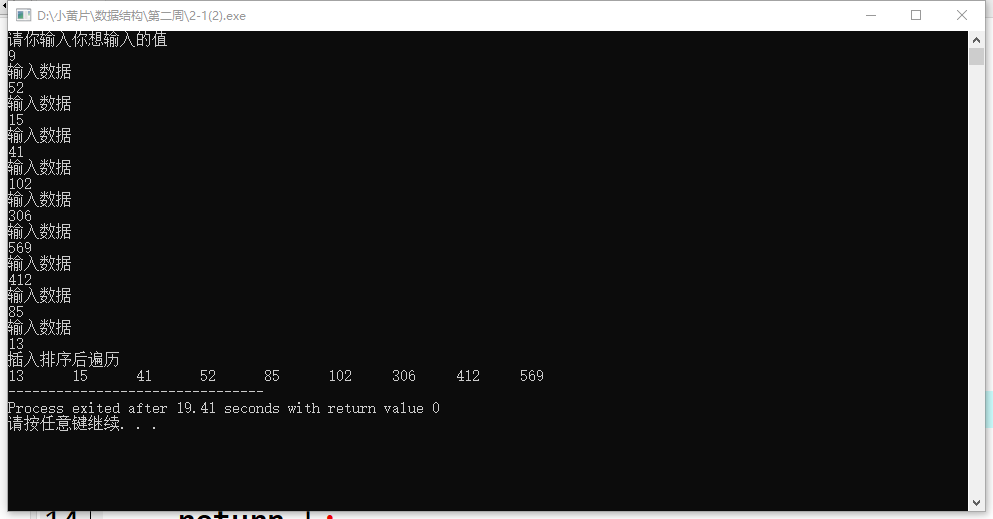
**数据结构上机实验报告**

1. 题目：设元素值为整型的线性表L，分别采用顺序结构和链式结构存储，编写程序，用插入排序算法实现线性表的表排序。
2. 算法思想：运用顺序表，每输入一个数据，就找到这个数据该插入的位置从而实现插入
3. 运行结果：

<1>顺序表：

<2>链表：



1. 结果分析：建立一个顺序表，输入n，即总共的数据，分别输入数值，插入排序后，遍历一次
2. 源程序：

<1>顺序表：

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

#define LIST\_INIT\_SIZE 100

#define LISTINCERMENT 10

typedef struct List{

int \*elem;

int length;

int listsize;

}Sqlist;

void InitList(Sqlist \*list)

{

list->elem=(int \*)malloc(LIST\_INIT\_SIZE\*sizeof(int));

if(!list->elem)

{

printf("创建顺序表失败！\n");

exit(0);

}

list->length=0;

list->listsize=LIST\_INIT\_SIZE;

printf("创建顺序表成功！\n");

}

void Insert\_Sort(Sqlist \*list,int m)

{

int \*p;

int \*newbase;

p=list->elem;

int i,j;

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

{

if(m<\*(list->elem+i)) break;

}

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

{

\*(list->elem+j)=\*(list->elem+j-1);

}

\*(list->elem+i)=m;

}

void createList(Sqlist \*list,int n)

{

int i;

int \*newbase;

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

{

if(list->length>=list->listsize)

{

newbase=(int\*)realloc(list->elem,(list->listsize+LISTINCERMENT)\*sizeof(int));

list->elem=newbase;

list->listsize+=LISTINCERMENT;

}

printf("输入数：");

scanf("%d",list->elem+i);

list->length++;

Insert\_Sort(list,\*(list->elem+i));

}

}

void visit(int \*p)

{

printf("%d\t",\*p);

}

void ListTraverse(Sqlist p)

{

int i=0;

for(;i<p.length;i++)

{

visit(p.elem+i);

}

}

int main()

{

Sqlist L;

InitList(&L);

int n;

printf("输入你想放入的值的个数：");

scanf("%d",&n);

createList(&L,n);

printf("插入排序后的遍历\n");

ListTraverse(L);

return 0;

}

<2>链表：

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

typedef struct lnode

{

int data;

struct lnode \*next;

}LNode,\*LinkList;

LinkList InitList()

{

LinkList L;

L=(LinkList)malloc(sizeof(LNode));

L->next=NULL;

return L;

}

void Insert\_Sort(LinkList list,LNode\* q)

{

LNode \*p,\*s;

p=list;

s=p->next;

if(s==NULL){

p->next=q;

q->next=s;

return;

}

while(s&&s->data<q->data)

{

p=s;

s=p->next;

}

q->next=s;

p->next=q;

}

void createList(LinkList list,int n)

{

LNode \*p,\*q;

p=list;

int i;

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

{

q=(LNode \*)malloc(sizeof(LNode));

printf("输入数据\n");

scanf("%d",&q->data);

Insert\_Sort(list,q);

}

}

int visit(int m)

{

printf("%d\t",m);

}

void ListTraverse(LinkList list)

{

LNode \*p,\*q;

p=list;

q=p->next;

while(q)

{

visit(q->data);

p=q;

q=p->next;

}

}

int main()

{

LinkList L;

int n;

L=InitList();

printf("请你输入你想输入的值\n");

scanf("%d",&n);

createList(L,n);

printf("插入排序后遍历\n");

ListTraverse(L);

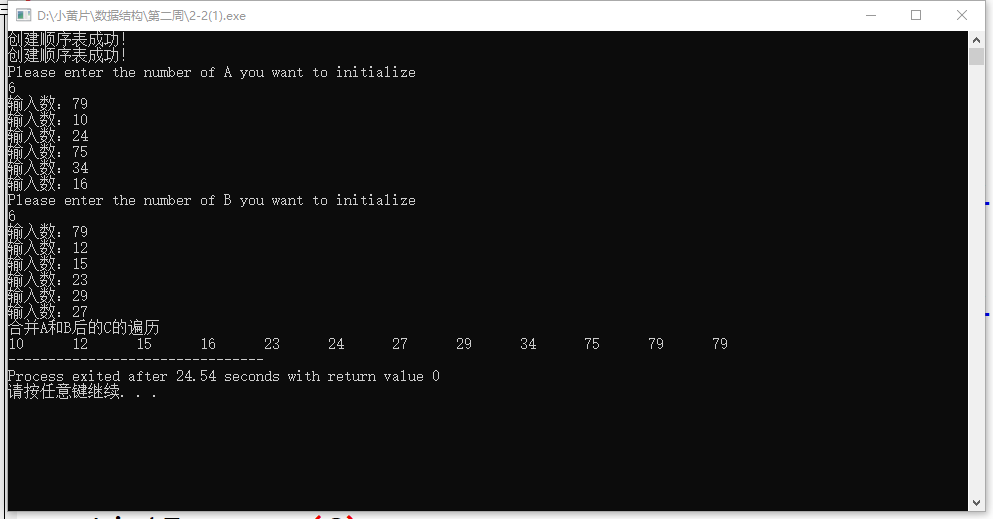
return 0;

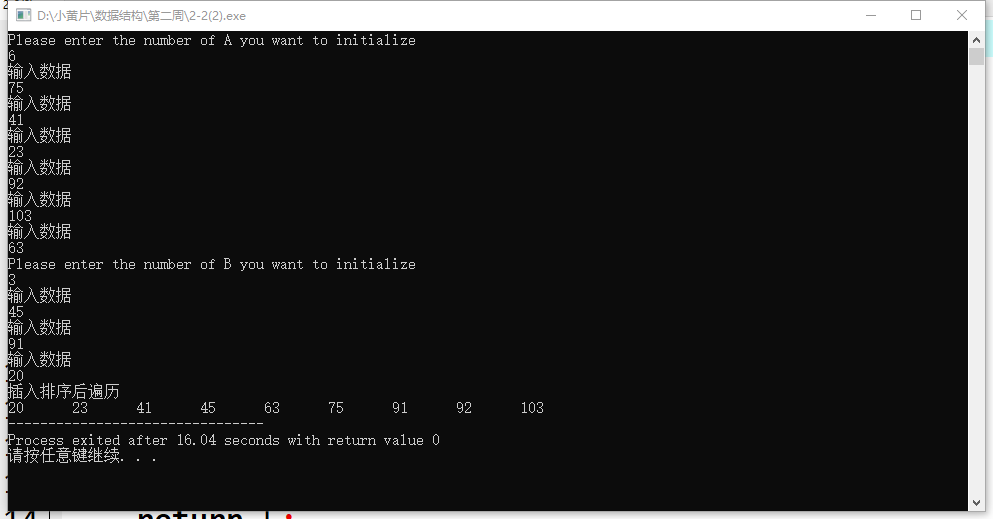
}

1. （1）题目：设元素值为整型的线性表A,、B，分别采用顺序结构和链式结构存储，编写程序，实现下列功能：假设以递增有序排列的线性表A和B 分别表示两个集合，现要求在A的空间上构成一个新线性表C，其元素为A和B元素的交集，且表C中的元素也是依值递增有序排列。

（2）算法思想：给两个顺序表A，B，在A的空间上进行操作，对B的元素进行遍历的同时对A进行遍历，查找插入的位置。

（3）运行结果：

<1>顺序表：

<2>链表：

（4）结果分析：<1>建立两个顺序表，对A列表输入6个数据，对B输入6数据，最后对C进行遍历。

<2>分别建立两个链表A,B对A输入了6个数据，对B输入3个数据，最后对B进行遍历。

源代码：

<1>：顺序表：

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

#define LIST\_INIT\_SIZE 100

#define LISTINCERMENT 10

typedef struct List{

int \*elem;

int length;

int listsize;

}Sqlist;

void InitList(Sqlist \*list)

{

list->elem=(int \*)malloc(LIST\_INIT\_SIZE\*sizeof(int));

if(!list->elem)

{

printf("创建顺序表失败！\n");

exit(0);

}

list->length=0;

list->listsize=LIST\_INIT\_SIZE;

printf("创建顺序表成功！\n");

}

void Insert\_Sort(Sqlist \*list,int m)

{

int \*p;

p=list->elem;

int i,j;

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

{

if(m<\*(list->elem+i)) break;

}

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

{

\*(list->elem+j)=\*(list->elem+j-1);

}

\*(list->elem+i)=m;

}

void createList(Sqlist \*list,int n)

{

int i;

int \*newbase;

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

{

if(list->length>=list->listsize)

{

newbase=(int\*)realloc(list->elem,(list->listsize+LISTINCERMENT)\*sizeof(int));

list->elem=newbase;

list->listsize+=LISTINCERMENT;

}

printf("输入数：");

scanf("%d",list->elem+i);

list->length++;

Insert\_Sort(list,\*(list->elem+i));

}

}

void MergeList(Sqlist \*LC,Sqlist \*LA,Sqlist \*LB)

{

int i=0;

int \*newbase;

while(i<LB->length)

{

if(LC->length>=LC->listsize)

{

newbase=(int\*)realloc(LC->elem,(LC->listsize+LISTINCERMENT)\*sizeof(int));

LC->elem=newbase;

LC->listsize+=LISTINCERMENT;

}

Insert\_Sort(LC,\*(LB->elem+i));

i++;

LC->length++;

}

}

void visit(int \*p)

{

printf("%d\t",\*p);

}

void ListTraverse(Sqlist p)

{

int i=0;

for(;i<p.length;i++)

{

visit(p.elem+i);

}

}

int main()

{

Sqlist A,B,C;

InitList(&A);

InitList(&B);

int num\_A,num\_B;

printf("Please enter the number of A you want to initialize\n");

scanf("%d",&num\_A);

createList(&A,num\_A);

printf("Please enter the number of B you want to initialize\n");

scanf("%d",&num\_B);

createList(&B,num\_B);

C=A;

MergeList(&C,&A,&B);

printf("合并A和B后的C的遍历\n");

ListTraverse(C);

return 0;

}

<2>链表：

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

typedef struct lnode

{

int data;

struct lnode \*next;

}LNode,\*LinkList;

LinkList InitList()

{

LinkList L;

L=(LinkList)malloc(sizeof(LNode));

L->next=NULL;

return L;

}

void Insert\_Sort(LinkList list,LNode\* q)

{

LNode \*p,\*s,\*t;

p=list;

s=p->next;

t=(LNode\*)malloc(sizeof(LNode));

t->data=q->data;

if(s==NULL){

p->next=t;

t->next=s;

return;

}

while(s&&s->data<q->data)

{

p=s;

s=p->next;

}

t->next=s;

p->next=t;

}

void createList(LinkList list,int n)

{

LNode \*p,\*q;

p=list;

int i;

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

{

q=(LNode \*)malloc(sizeof(LNode));

printf("输入数据\n");

scanf("%d",&q->data);

Insert\_Sort(list,q);

}

}

void MergeList(LinkList C,LinkList B)

{

LNode \*pb,\*qb,\*mb;

pb=B;

qb=pb->next;

if(qb==NULL) return;

mb=qb->next;

while(mb)

{

Insert\_Sort(C,qb);

qb=mb;

mb=mb->next;

}

Insert\_Sort(C,qb);

}

int visit(int m)

{

printf("%d\t",m);

}

void ListTraverse(LinkList list)

{

LNode \*p,\*q;

p=list;

q=p->next;

while(q)

{

visit(q->data);

p=q;

q=p->next;

}

}

int main()

{

LinkList A,B,C;

int num\_A,num\_B;

A=InitList();

B=InitList();

printf("Please enter the number of A you want to initialize\n");

scanf("%d",&num\_A);

createList(A,num\_A);

printf("Please enter the number of B you want to initialize\n");

scanf("%d",&num\_B);

createList(B,num\_B);

C=A;

MergeList(C,B);

printf("插入排序后遍历\n");

ListTraverse(C);

return 0;

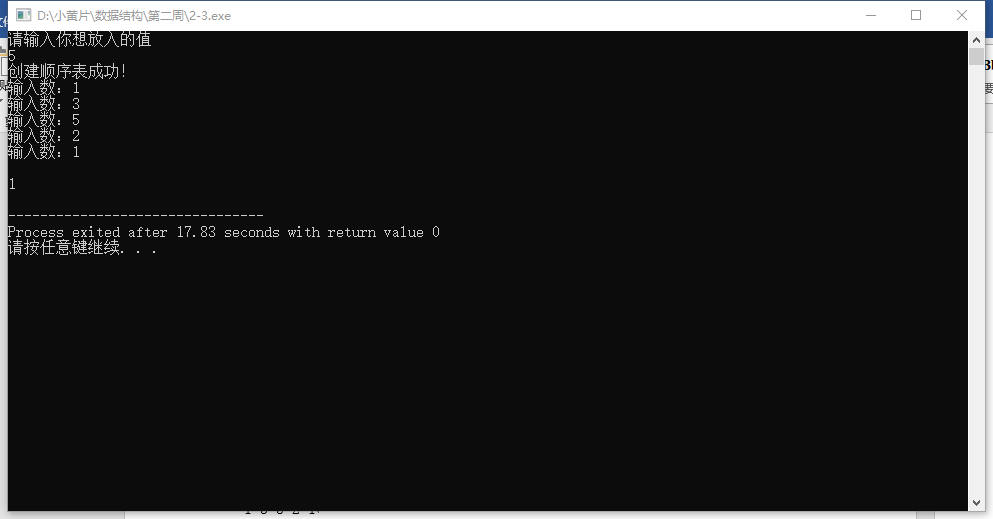
}

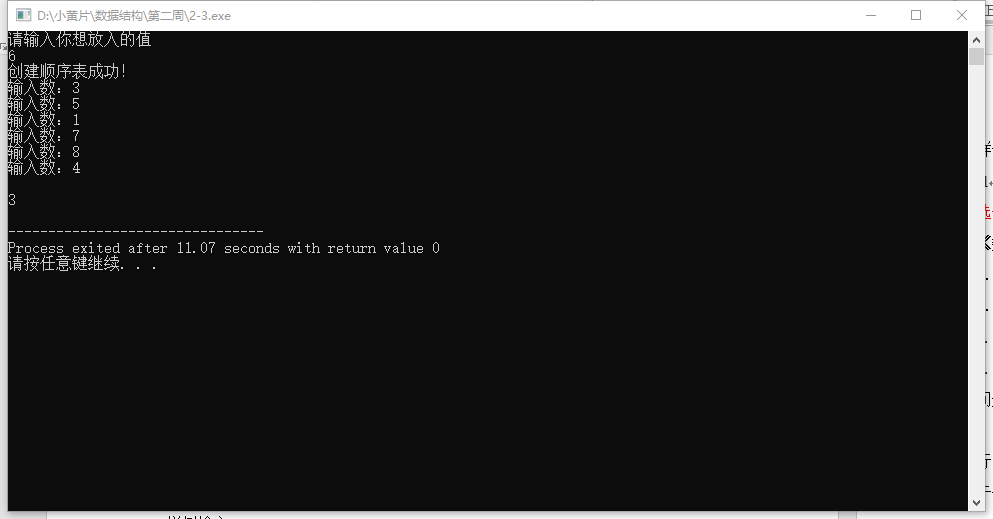
三：

（1）题目：**问题描述：**先输入一个十进制整数n，再输入n个正整数，求它们相邻数之差可知是否为上升或下降，由上升转下降或由下降转上升为折点，求折点数。

**问题分析：**如果一个点的值比左右两个都大或都小，则为折点。

(2)算法思想：运用线性表，遍历每个数据，对每个数据的前一个位置和后一个位置进行判断，是否同时大于这个数据，或者是否同时小于这个数据

(3)运行结果：



(4)结果分析：输入值后，对每个数据进行遍历，遍历完后输出有几个这样的折点。

源程序：

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

#define LIST\_INIT\_SIZE 100

#define LISTINCERMENT 10

typedef struct List{

int \*elem;

int length;

int listsize;

}Sqlist;

void InitList(Sqlist \*list)

{

list->elem=(int \*)malloc(LIST\_INIT\_SIZE\*sizeof(int));

if(!list->elem)

{

printf("创建顺序表失败！\n");

exit(0);

}

list->length=0;

list->listsize=LIST\_INIT\_SIZE;

printf("创建顺序表成功！\n");

}

void createList(Sqlist \*list,int n)

{

int i;

int \*newbase;

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

{

if(list->length>=list->listsize)

{

newbase=(int\*)realloc(list->elem,(list->listsize+LISTINCERMENT)\*sizeof(int));

list->elem=newbase;

list->listsize+=LISTINCERMENT;

}

printf("输入数：");

scanf("%d",list->elem+i);

list->length++;

}

}

int TurnPoint(Sqlist \*L,int n)

{

int i=1;

int total=0;

for(;i<n-1;i++)

{

if(\*(L->elem+i-1)>\*(L->elem+i)&&\*(L->elem+(i+1)%n)>\*(L->elem+i)) total++;

if(\*(L->elem+i-1)<\*(L->elem+i)&&\*(L->elem+(i+1)%n)<\*(L->elem+i)) total++;

}

return total;

}

int main()

{

int n;

int total;

Sqlist L;

printf("请输入你想放入的值\n");

scanf("%d",&n);

InitList(&L);

createList(&L,n);

total=TurnPoint(&L,n);

printf("\n%d\n",total);

return 0;

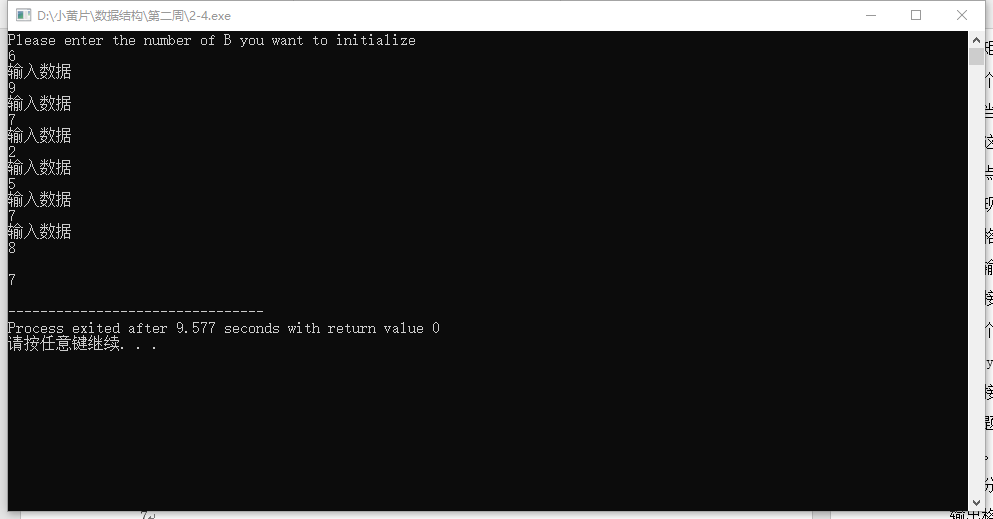
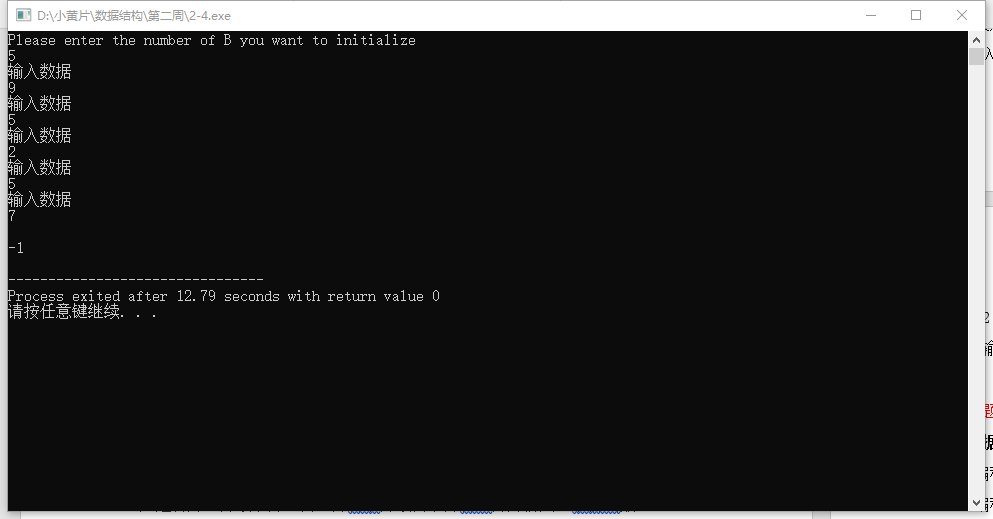
}

四、

（1）题目：**问题描述：**首先输入正整数n，接着输入n个正整数，如果其中存在一个数，比该数大的个数等于比该数小的个数，则输出该数，如果不存在则输出-1。

**问题分析：**这个问题可以用排序来实现。

（2）算法思想：用链表来存储数据并用插入排序拍好顺序，再开另外一个链表，通过插入排序来排序，若遇到一样的则跳过，将新开的链表设置为C

（3）运行结果： 

（4）结果分析：

<1> 给到6个数据，为9，7，2，5，7，8，新开一个链表，其中排序为2,5,7,8,9 故7满足题目条件

<2>给到5个数据，分别为9,5,2,5,7，最终在另一个链表中，顺序为，2,5,7,9，故没有数据满足题目，故输出-1。

源程序：

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

typedef struct lnode

{

int data;

struct lnode \*next;

}LNode,\*LinkList;

LinkList InitList()

{

LinkList L;

L=(LinkList)malloc(sizeof(LNode));

L->next=NULL;

return L;

}

void Insert\_Sort(LinkList list,LNode\* q)

{

LNode \*p,\*s,\*t;

p=list;

s=p->next;

t=(LNode\*)malloc(sizeof(LNode));

t->data=q->data;

if(s==NULL){

p->next=t;

t->next=s;

return;

}

while(s&&s->data<q->data)

{

p=s;

s=p->next;

}

if(s&&s->data==q->data){

p->next=s->next;

free(s);

s=p->next;

}

t->next=s;

p->next=t;

}

void createList(LinkList list,int n)

{

LNode \*p,\*q;

p=list;

int i;

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

{

q=(LNode \*)malloc(sizeof(LNode));

p->next=q;

printf("输入数据\n");

scanf("%d",&q->data);

p=q;

}

p->next=NULL;

}

void MergeList(LinkList C,LinkList B)

{

LNode \*pb,\*qb,\*mb;

pb=B;

qb=pb->next;

if(qb==NULL) return;

mb=qb->next;

while(mb)

{

Insert\_Sort(C,qb);

qb=mb;

mb=mb->next;

}

Insert\_Sort(C,qb);

}

void Judge(LinkList C)

{

LNode \*p,\*p1,\*q,\*q1;

p1=q1=C->next;

int SmallCNT=0,BigCNT=0;

bool m=0;

while(p1)

{

while(q1)

{

if(q1->data>p1->data) BigCNT++;

if(q1->data<p1->data) SmallCNT++;

q1=q1->next;

}

if(SmallCNT==BigCNT&&SmallCNT!=0&&BigCNT!=0)

{

printf("\n%d\n",p1->data);

m=1;

}

SmallCNT=0;

BigCNT=0;

q1=C->next;

p1=p1->next;

}

if(!m) printf("\n-1\n");

}

int main()

{

LinkList B,C;

int num\_B;

C=InitList();

B=InitList();

printf("Please enter the number of B you want to initialize\n");

scanf("%d",&num\_B);

createList(B,num\_B);

MergeList(C,B);

Judge(C);

return 0;

}