**北京科技大学实验报告**

学院：计通学院 专业：物联网工程 班级： 物联1602

姓名： 周康能 学号： 41624657 实验日期： 2018年 5 月 21 日

**实验名称：《数据结构》实验1链表的应用**

**实验目的：熟悉链表的基本操作，掌握线性表的链式存储结构及应用。**

**实验内容：**

输入数据（设为整型）建立单链表，并求相邻k个节点data值之和为最大的第一节点。

要求：

（1）建立链表、求最大值功能采用独立函数实现；

（2）可根据用户需求多次建表；

（3）数据的输入可从键盘输入，也可从txt文件输入；

（4）程序结束时要释放链表空间。

**问题分析与算法思路：**

此题较为基础，用到的数据结构和算法较为简单，数据结构为简单的链式结构。

算法为遍历链表，记下相邻的节点，以及其的和，接着不断往后面走，到读到NULL的时候停止。其中，若有和的值超过之前的值，则替换结点。

此算法可以准确算出相邻k个节点的和为最大对应的第一节点，不会误判。且此算法由于是链式的，时间复杂度和空间复杂度都比较合适，且创建和释放链表函数对应，不会存在内存泄露现象。

首先应该申明链式结构体，然后分键盘输入和txt输入分别写判断语句，他们对应不同的建表方法。

总的来说，本程序的总体函数分为三个：

1、建立链表（分键盘输入的建立和txt输入的建立，已经封装成两个函数）

2、打印链表（显示出链表，便于直观观察）

3、计算相邻K个节点的和的函数

4、销毁链表的函数

**算法描述：**

//------------------------------------------------------------------------------------------------------------//

typedef struct tagNONE

{

int date;

struct tagNONE \*next;

}ListNode,\*LinkList;

此为定义链表的结构体，data是存储节点对应的值，next为指向下一节点的指针。

//------------------------------------------------------------------------------------------------------------//

LinkList Creatlist()

{

此处定义节点

while (1)

{

是数则存入，是#则开始分析

}

Middd->next= NULL; //最后要指向NULL，很重要

return Head; //最后返回一个头节点

}

此为键盘输入的建表函数的简单描述，txt的类似，具体见详细程序，这里不再累述，具体为一个个读入输入的数据，若是数，则存入链表，若是#则开始分析。

//------------------------------------------------------------------------------------------------------------//

int Adjmax(LinkList L, int k) //求相邻K个节点的和的函数

{

求k个节点的和

并记录头节点的次序

While（1）

{

节点后移

每后移一次，计算一次节点的和

若和比之前的和大，则替换和的值以及节点的次序

}

输出最大的和以及其对应的节点次序  
}

//------------------------------------------------------------------------------------------------------------//

void Destroy\_List(LinkList L)

{

while (L)

{

LinkList S = L;

L = L->next;

free(S);

}

}

销毁链表，逐个节点释放，防止有内存泄漏的情况出现，注意释放和清零的区别。

**程序实现：**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Copyright(c) :Ellis

\* All rights reserved.

\*

\* 文件名称:USTB2018\_data\_structure\_test1

\* 简要描述:求相邻k个节点数值之和最大的第一节点

\* 输入数据（设为整型）建立单链表，并求相邻k个节点data值之和为最大的第一节点。例如输入k = 2，数据为2 6 4 7 3 #（#为结束符），

\* 建立下面链表，运行结果 输出（序号3，data值4）

\*

\* 创建日期:20180406\_22:57

\* 作者:Ellis

\* 说明:完成建表

\*

\* 修改日期:20180407\_10:02

\* 作者:Ellis

\* 说明:完成任务，修改完注释

\* 注意：本程序因输入时有“#”符号，故所有函数都有意忽略了最后的节点

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : 宏定义、头文件、命名空间区

\* 创建日期: 20180406

\* 作者 : Ellis

\* 说明 : 无

\*

\* 修改日期:

\* 作者 :

\* 说明 :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define \_CRT\_SECURE\_NO\_WARNINGS //解决scanf不安全问题

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#include <cstdio>

#include <iostream>

#include <string>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : 结构体定义区

\* 创建日期: 20180406

\* 作者 : Ellis

\* 说明 : 链表式的结构体，数据+后继指针

\*

\* 修改日期:

\* 作者 :

\* 说明 :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef struct tagNONE

{

int date;

struct tagNONE \*next;

}ListNode,\*LinkList;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : 函数定义区

\* 创建日期: 20180406

\* 作者 : Ellis

\* 说明 : 定义各种函数

\*

\* 修改日期:

\* 作者 :

\* 说明 :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

LinkList Creatlist(); //创建链表

LinkList Creatlist\_txt();

void Test\_Print\_List(LinkList pList); //打印处于链表

int input(); //输入函数

int Adjmax(LinkList L, int k); //求相邻k个节点数值之和最大的第一节点

void Destroy\_List(LinkList L); //销毁链表

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:int main(int argc, char\*\* argv)

Author :Ellis

Date :20180406

Description :主函数

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(int argc,char\*\* argv)

{

while (1)

{

LinkList p;

while (1)

{

printf("Please enter the way you choose:\n");

printf("tips: '1' to use keyboard,'2' to use txt.\n");

int s = input();

if (s == 1)

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

p = Creatlist(); //建表

break;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

}

else if (s == 2)

{

p = Creatlist\_txt();

break;

}

else

{

printf("try again.\n");

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

Test\_Print\_List(p); //打印

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

printf("Please enter k : \t");

int k = input();

//getchar();

printf("\n");

printf(">>>The result is :...\n");

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

printf("和为%d\n", Adjmax(p, k)); //打印出结果

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

Destroy\_List(p); //销毁链表

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

printf(">>>Continue or not ?... \n");

printf("Enter Y to continue/t Enter N to exit...\n");

while (1)

{

getchar();

if (getchar() == 'Y')

{

//continue;

break;

}

else if (

getchar() == 'N')

{

system("pause");

exit(1);

}

else

{

printf("Error!Try again.\n");

continue;

//system("pause");

//exit(1);

}

}

}

system("pause");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:LinkList Creatlist()

Author :Ellis

Date :20180406

Description :建表函数

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

LinkList Creatlist()

{

printf(">>>================List Building Start=============\n");

LinkList Middd = NULL; //不断移动的节点

LinkList Head = NULL; //始终指向链表的头

LinkList P\_List = NULL; //不断创建的新节点

Head = Middd = (LinkList)malloc(sizeof(ListNode));

printf(">>>Please enter the numbers:...\n");

while (1)

{

int s = input();

P\_List = (LinkList)malloc(sizeof(ListNode));

P\_List->date = s;

Middd->next = P\_List;

Middd = P\_List;

if (getchar() == '#')

{

break;

}

}

Middd->next= NULL;

return Head;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:LinkList Creatlist\_txt()

Author :Ellis

Date :20180406

Description :建表函数

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

LinkList Creatlist\_txt()

{

FILE \*file;

char name\_of\_txt[20];

printf("Please enter the name of the txt...\n");

scanf("%s", name\_of\_txt);

file = fopen(name\_of\_txt, "r");

if (!file)

{

printf("Could not open this txt!\n");

system("pause");

}

else

{

printf("Open the txt successfully!\n");

}

int ch;

printf(">>>================List Building Start=============\n");

LinkList Middd = NULL; //不断移动的节点

LinkList Head = NULL; //始终指向链表的头

LinkList P\_List = NULL; //不断创建的新节点

Head = Middd = (LinkList)malloc(sizeof(ListNode));

while (

(ch = fgetc(file)) != EOF //逐个字符去读取

)

{

if (ch == ' ')

{

continue;

}

//int s = input();

P\_List = (LinkList)malloc(sizeof(ListNode));

P\_List->date = ch - 48;

Middd->next = P\_List;

Middd = P\_List;

if (ch == '#')

{

break;

}

}

Middd->next = NULL;

return Head;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void Test\_Print\_List(LinkList pList)

Author :Ellis

Date :20180406

Description :打印函数

Inputs :

Outputs :

Notes :打印出一切链表

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Test\_Print\_List(LinkList pList)

{

printf("================Start to print=============.\n");

LinkList nList = pList->next;

while (nList->next)

{

printf("%d\t", nList->date);

nList = nList->next;

}

printf("\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:int input()

Author :Ellis

Date :20180406

Description :输入函数

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int input()

{

int w;

scanf("%d", &w);

return w;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:int Length(LinkList x)

Author :Ellis

Date :20180407

Description :求长度函数

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Length(LinkList x)

{

LinkList H = x->next;

int i = -1;

while (H)

{

++i;

H = H->next;

}

return i;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:int Adjmax(LinkList L, int k)

Author :Ellis

Date :20180407

Description :求最大值函数

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Adjmax(LinkList L, int k)

{

int length = Length(L);

if (k > length)

{

printf("Error! \n");

system("pause");

exit(1);

}

LinkList LL = L;

int sum\_real = 0;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//第一次分开来写，因为比较难搞定头节点问题

int count = k;

int j = 1; //序列号

int realj = 0; //需要输出的

LinkList mid = L; //注意，不能乱动L，要用中间变量

while (count)

{

sum\_real = sum\_real + mid->next->date;

mid = mid->next;

--count;

}

L = L->next; //重要

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

int nCount = length - k;

int sum = 0; //和

while (nCount)

{

count = k;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

LinkList mid\_list = L; //注意，不能乱动L，要用中间变量

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//求出k个连续量之和

while (count)

{

count--;

sum = sum + mid\_list->next->date;

mid\_list = mid\_list->next;

}

j++; //序号加1

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

L = L->next; //重要，L不断后移

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

nCount--;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//求最大值

if (sum > sum\_real)

{

realj = j; //若更新最大值，则赋值

sum\_real = sum;

sum = 0;

}

else

{

sum = 0;

continue;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

}

printf("序号%d\n", realj);

for (int s = 0; s < realj;s++)

{

LL = LL->next;

}

printf("值%d\n", LL->date);

return sum\_real;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void Destroy\_List(LinkList L)

Author :Ellis

Date :20180407

Description :销毁链表函数

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Destroy\_List(LinkList L)

{

while (L)

{

LinkList S = L;

L = L->next;

free(S);

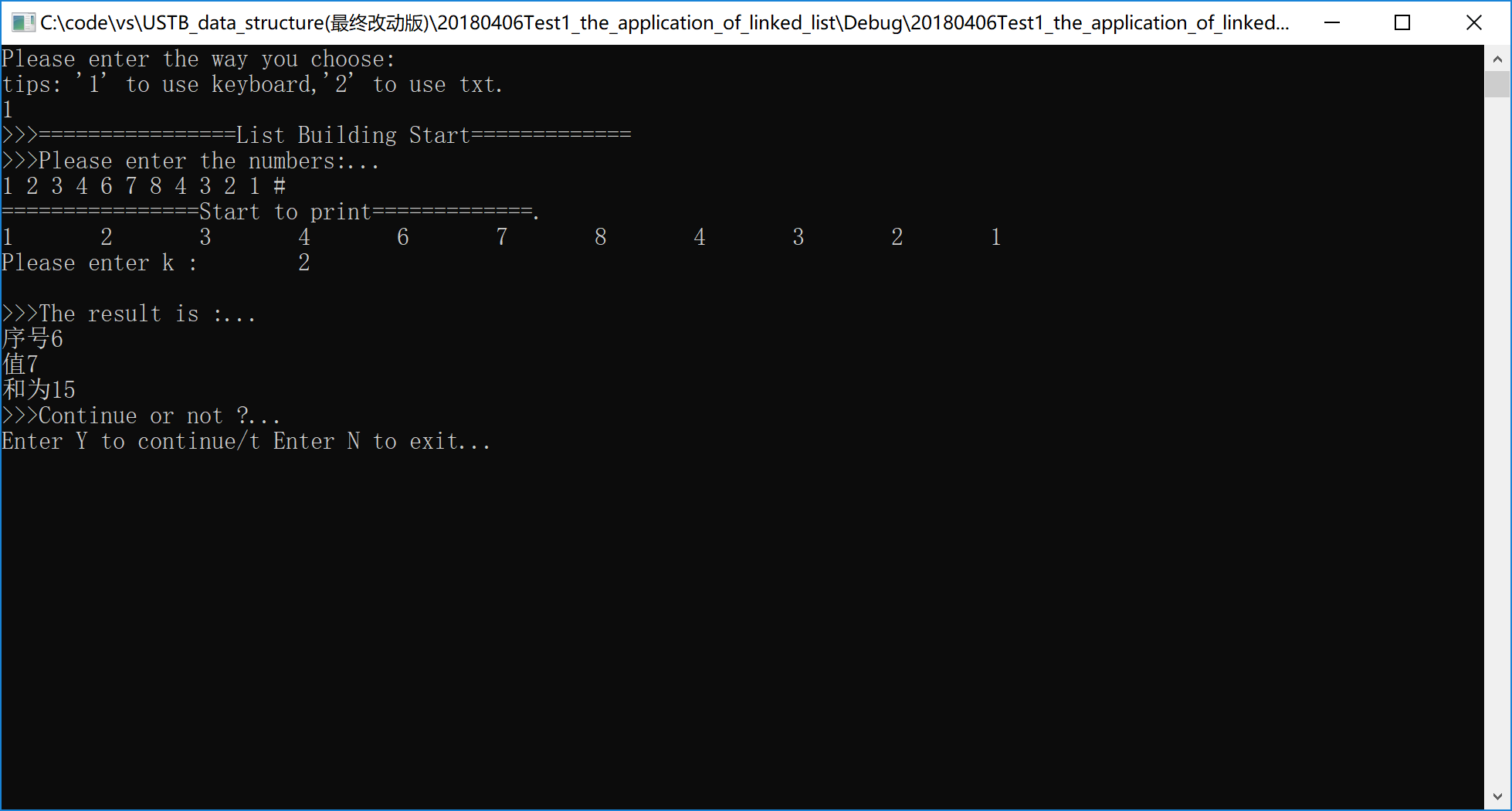
}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

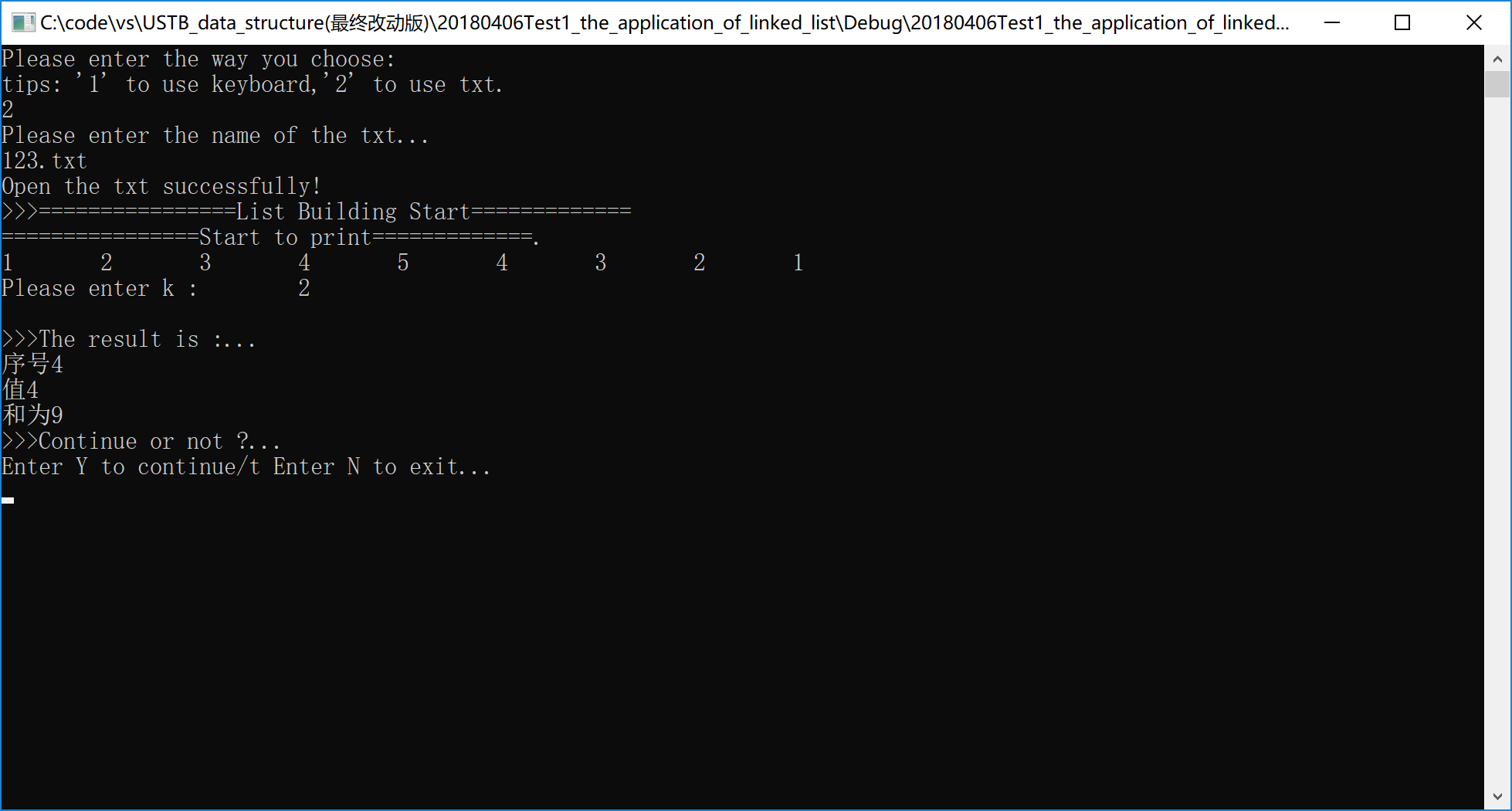
**测试：**

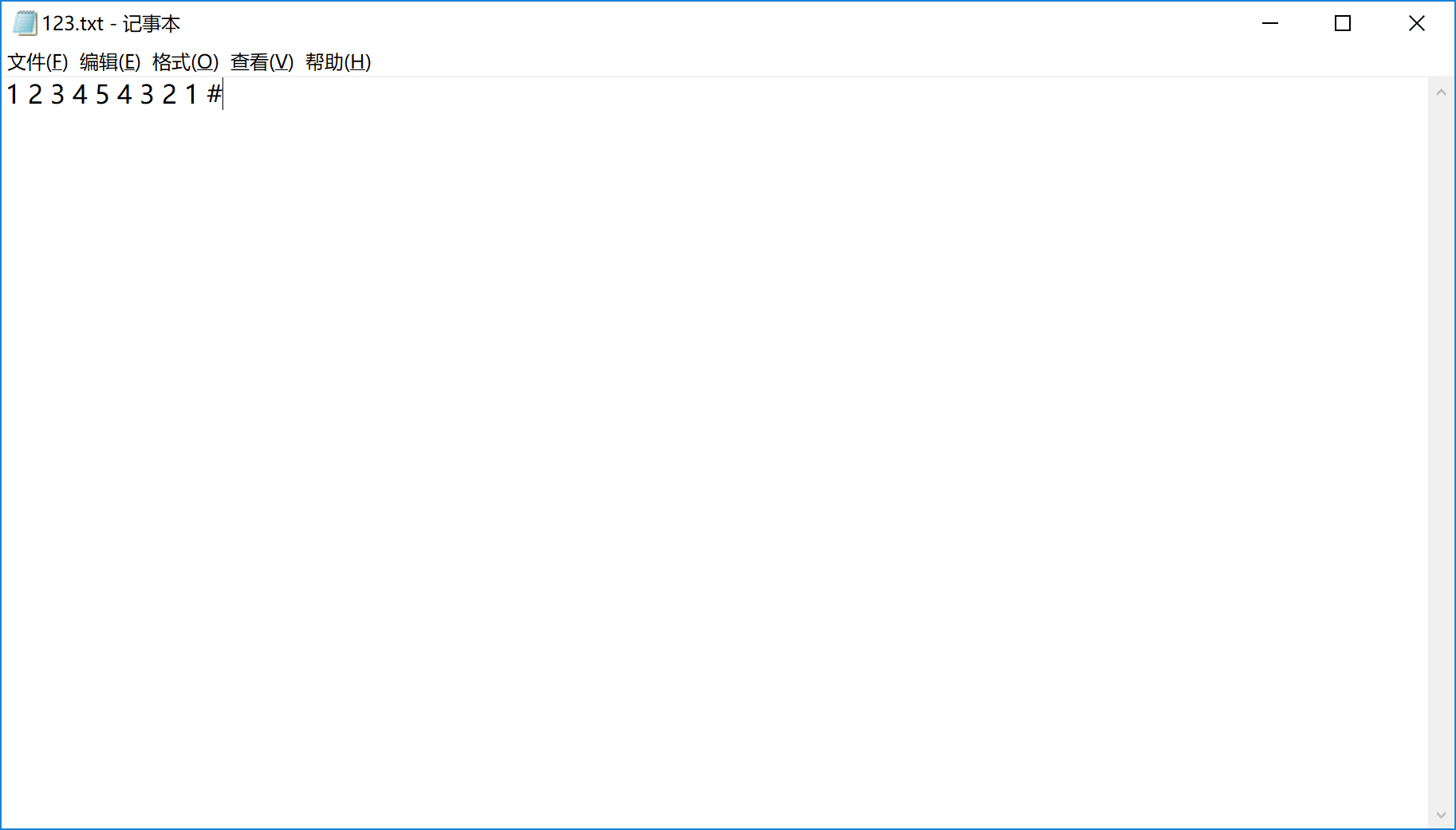
首先，测试键盘输入正确性：



结构正确。

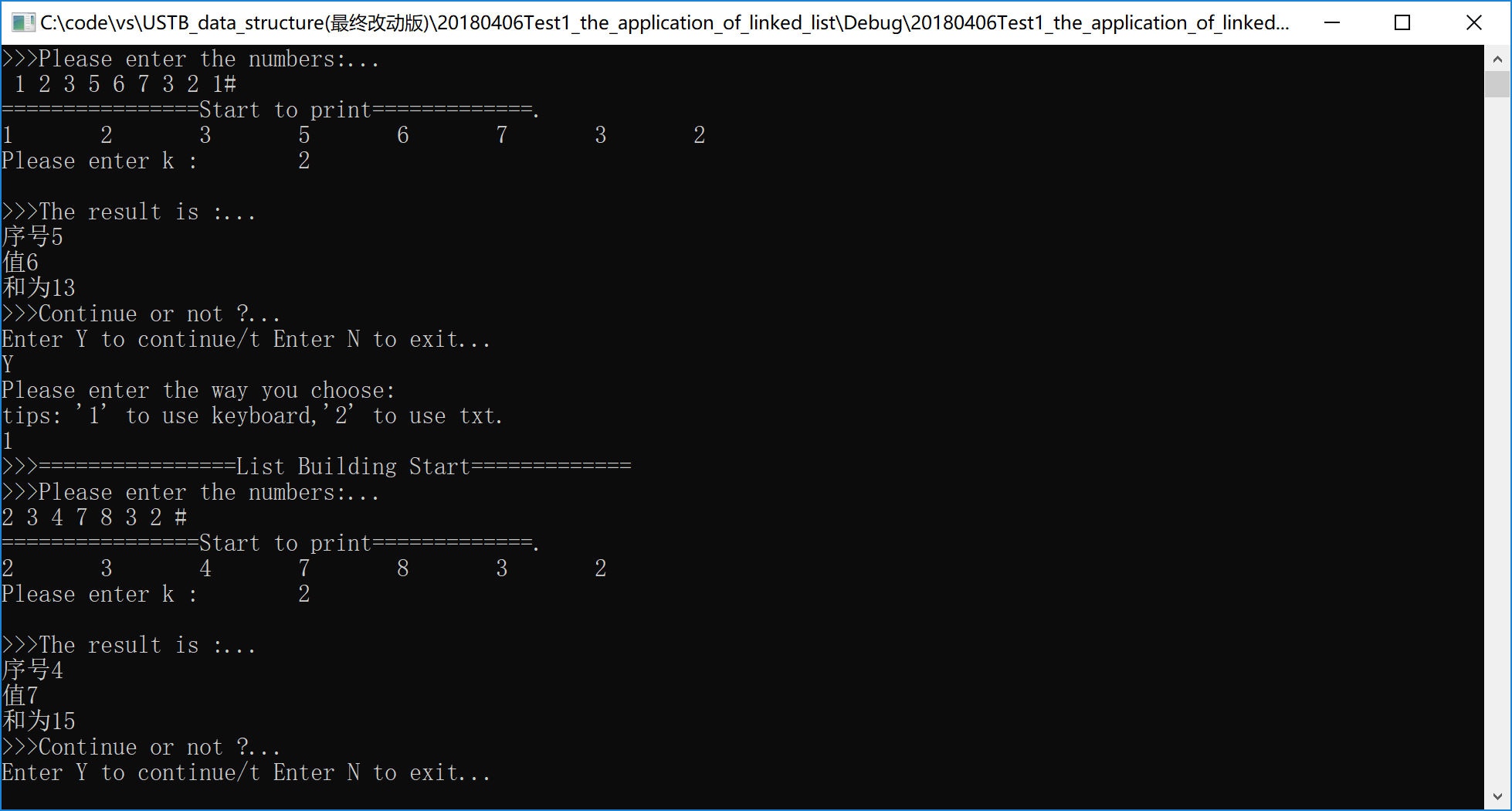
然后，测试txt输入正确性：

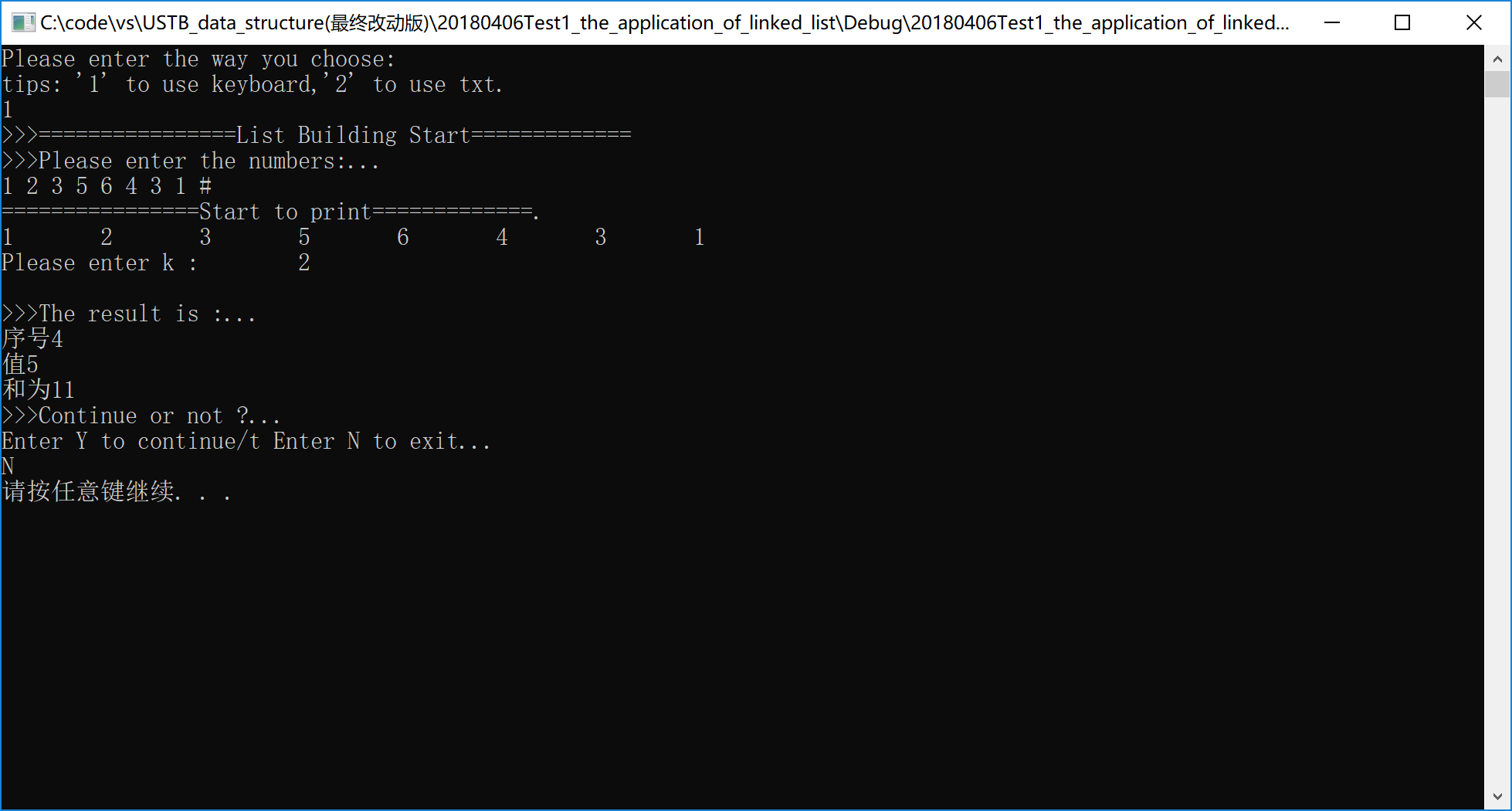




结果正确。

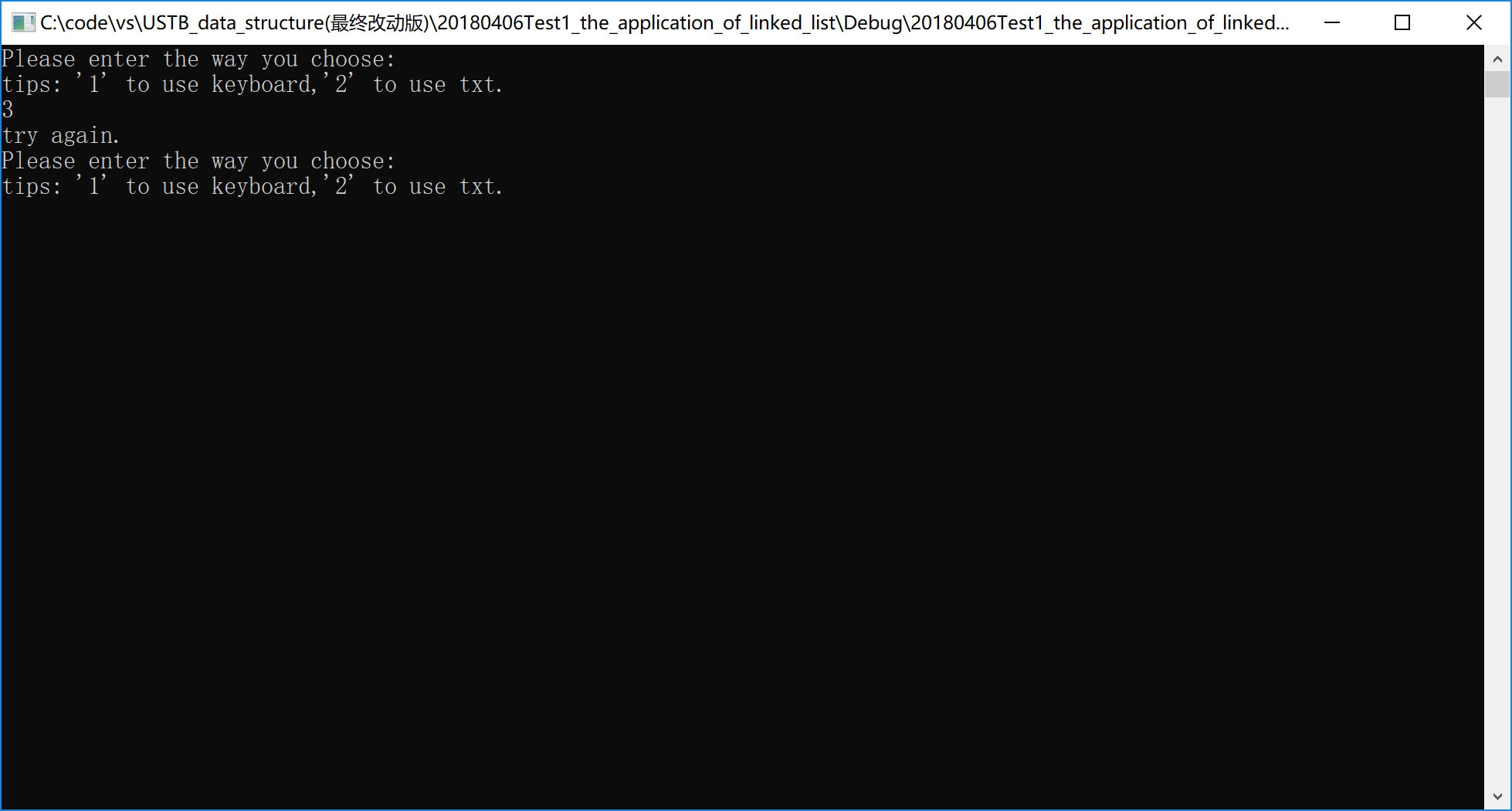
测试重试：



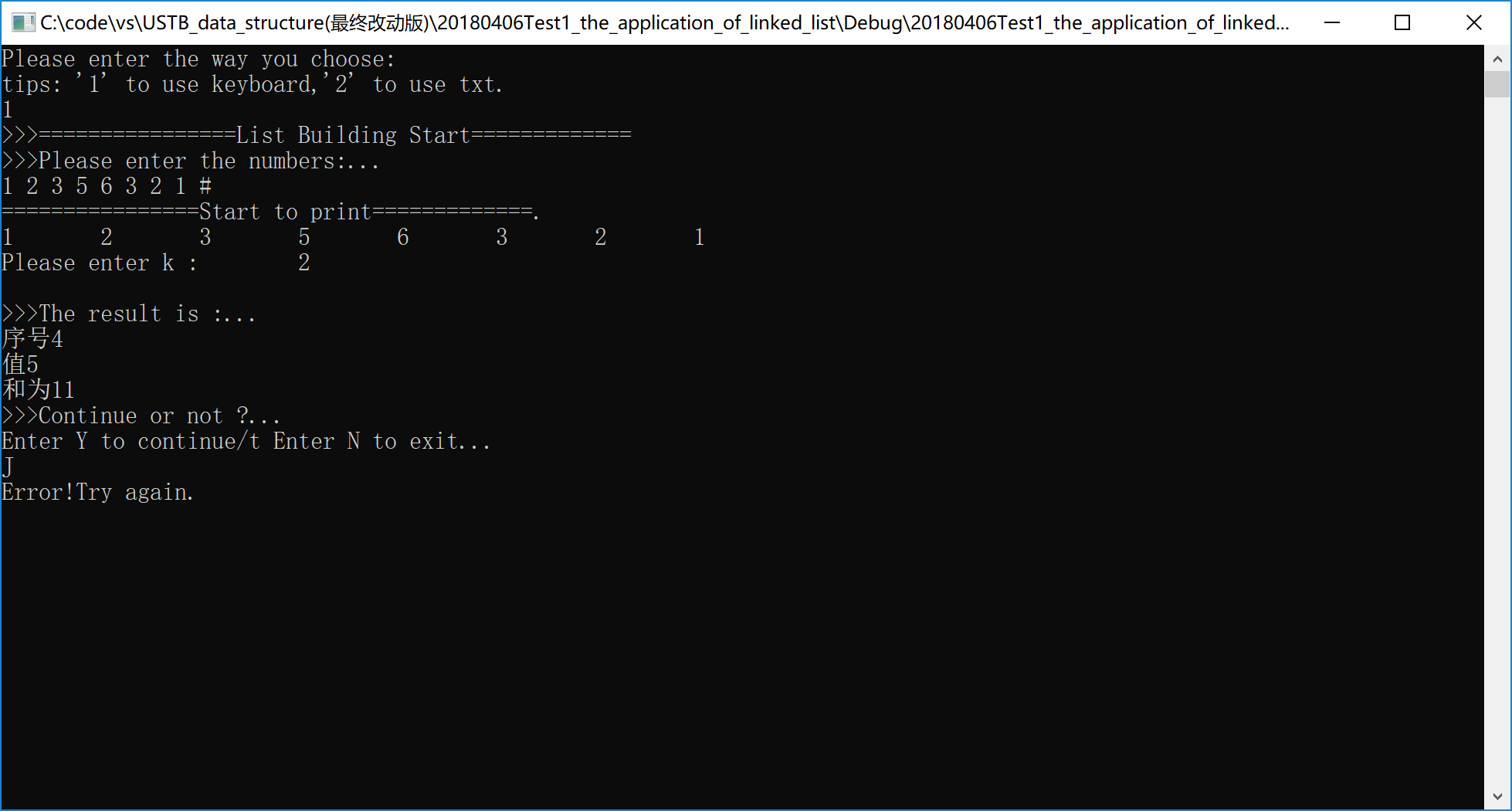


测试错误输入的情况：

1. 选择输入方式的错误

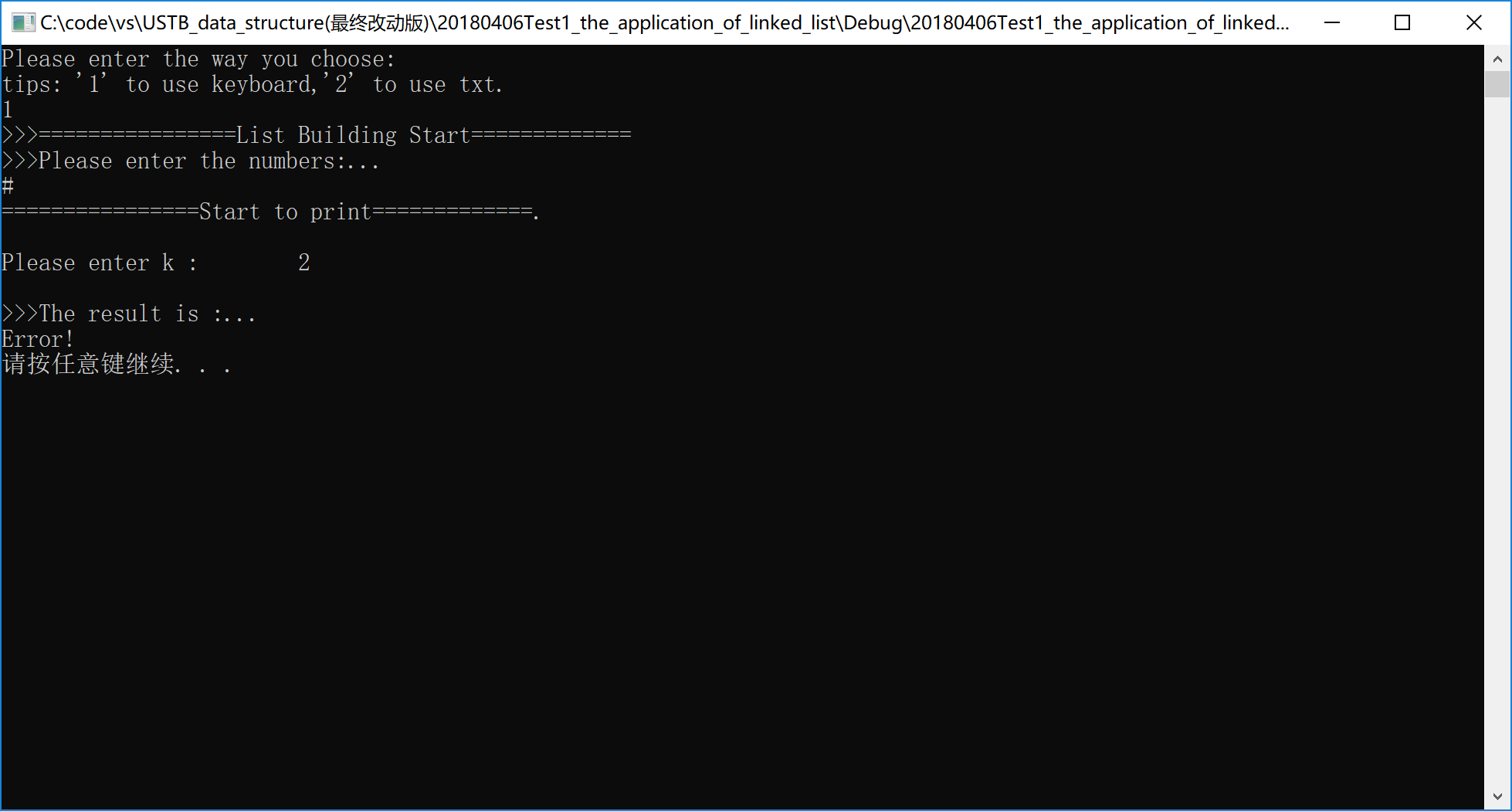


1. 重试时的错误输入



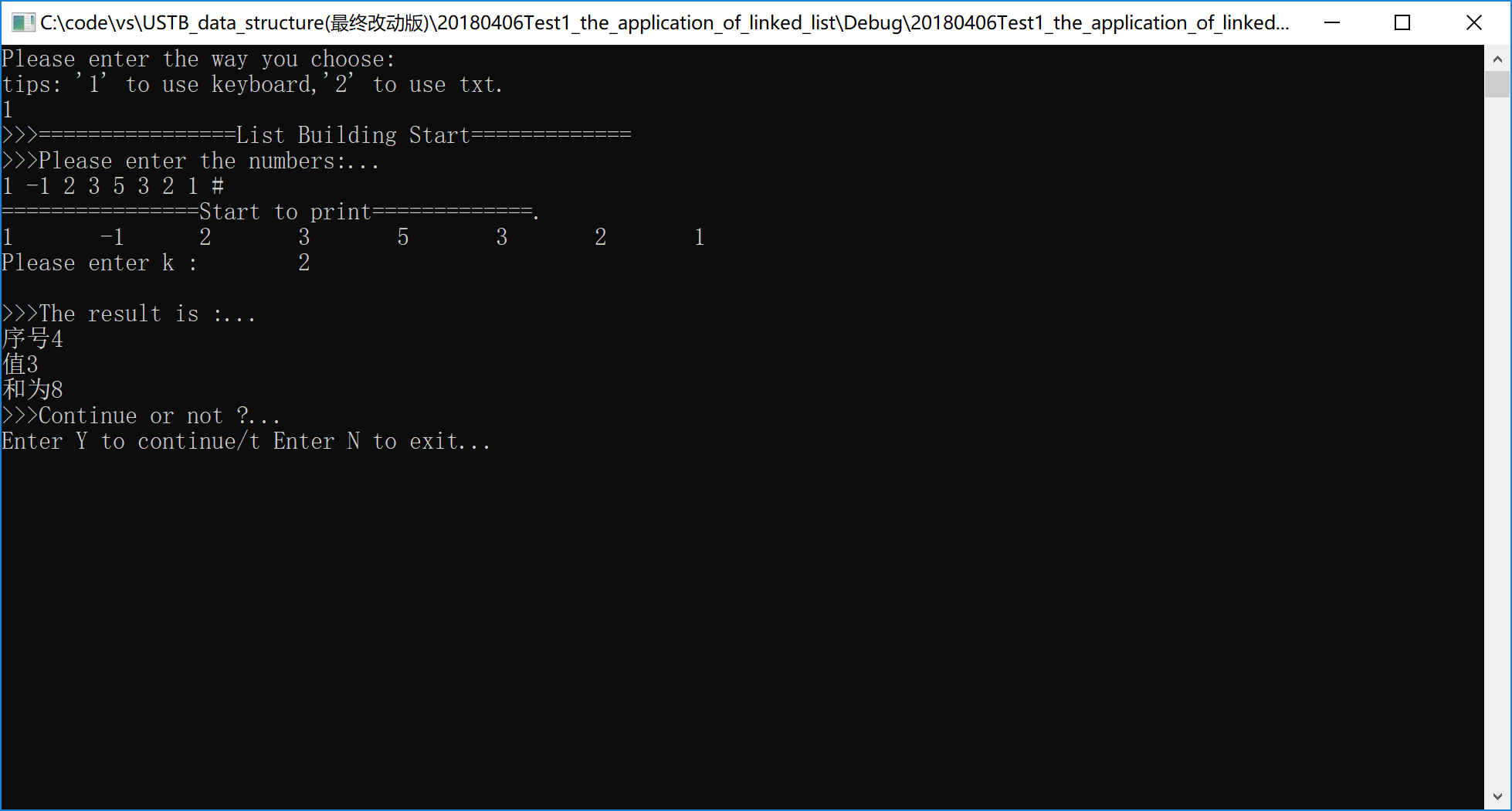
测试稳定性：

1. 空链表情况：



测试稳定

2、负数情况



分析算法的时间复杂度、空间复杂度、有效性、不足和改进意见。如果同时采用了多种实现方法，进行对比说明、分析。

**最终报告中请删除此框！**

测试稳定

**算法的有效性分析：**

时间复杂度及空间复杂度：由于使用的是链式结构，时间复杂度为O(n)，且空间复杂度合适。

有效性：链表创建和释放函数对应，避免了内存泄漏情况。算法正确，稳定性好，执行效率高，注释完整，分模块编写，容易阅读。且格式鲜明，作者，修改，日期等内容丰富且明确，有问题出现时容易查找，符合大型项目编写的格式。

不足及改进意见：代码虽然注释完整，容易阅读，但是行数太多，要注意简洁和方便这两者的平衡。