Lab Title: Basic Operation of Linear Tables

答案的计划：
1. Master the definition of linear table;
2. Master the basic operations of linear tables, such as creating, searching, inserting and deleting.

Experimental content:
Define a sequential list and linked list containing student information (student number, name, grade) with the following functions:
(1) Enter student information one by one according to the number of designated students;
(2) Display the relevant information of all students in the student table one by one;
(3) Search by name and return the student's student ID and grade;
(4) According to the specified location, the corresponding student information (student number, name, grade) can be returned;
(5) Given a student information, insert it into the specified position in the table;
(6) Delete student records at designated locations;
(7) The number of students in the statistics table.

Experiment Tips:
Definition of Student Information:

```
typedef struct {
    char no[8]; 8-digit student number
    char name[20]; name
    int price; grades
}Student;
```

The definition of a sequential table

```
typedef struct {
    Student *elem; Points to the base address of the data element
    int length; The current length of the linear table
}SqList;
```

Definition of linked list:

```
typedef struct LNode{
    Student data; Data fields
    struct LNode *next;
}LNode,*LinkList;
```

Experimental requirements:
(1) The program should be annotated appropriately, and the program should be written in indented format.
(2) The program must have a certain robustness, that is, when the input data is illegal, the program can also respond appropriately, such as the incorrect position specified when inserting or deleting, etc.

(3) The program should be user-friendly, and the user can operate according to the corresponding prompt information when the program is running.
(4) Write the experiment report in detail according to the experiment report template, and give the algorithm of the linked list to search by name and the flow chart of the insertion algorithm in the experiment report.
(5) Upload the source program and lab report to the folder of the corresponding class of FTP. The source program of the sequential list is saved as SqList.cpp, the source program of the linked list is saved as LinkList.cpp, and the experiment report is named: Experiment Report 1.doc. The source program and lab report are compressed into one file (or together if a header file is defined), named as follows: student number name.rar, such as 070814101 Xue Li .rar .

CODE:

SqList.cpp:

```
#include "StdAfx.h"
#include "SqList.h"

Status InitList(SqList &L)
{
    L.elem=new ElemType[MAXSIZE]; //Allocate an array space with a size of MAXSIZE for the sequence table
    if(!L.elem)
        exit(OVERFLOW); //Storage allocation failed to exit
    L.length=0; //The length of the empty list is 0
    return OK;
}

Status GetElem(SqList L,int i,char str1[],char str2[],int &e)
{
    if((i<1)||(i>L.length)) //Judge whether the value of i is reasonable, if not, return ERROR
        return ERROR;
    strcpy(str1,L.elem[i-1].no);
    strcpy(str2,L.elem[i-1].name);
    e=L.elem[i-1].price; //elem[i-1] unit stores the i-th data element
    return OK;
}

int LocateElem1(SqList L, int e)
{
    for(int i=0;i<L.length;i++)
    {
        if(L.elem[i].price==e)
            return i+1; //Search successfully, return serial number i+1
    }
    return 0; //Search failed, return 0
}

int LocateElem2(SqList L,char str[])
{
    for(int i=0;i<L.length;i++)
    {
        if(strcmp(L.elem[i].no,str)==0)
            return i+1; //Search successfully, return serial number i+1
    }
    return 0; //Search failed, return 0
```

```
}

int LocateElem3(SqList L,char str[])
{
	for(int i=0;i<L.length;i++)
	{
		if(strcmp(L.elem[i].name,str)==0)
			return i+1; //Search successfully, return serial number i+1
	}
	return 0; //Search failed, return 0
}

Status ListInsert(SqList &L,int i,char str1[],char str2[],int &e)
{
	if((i<1)||(i>L.length+1))
		return ERROR; //i value is illegal
	if(L. length==MAXSIZE)
		return ERROR; //The current storage space is full
	for(int j=L.length-1;j>=i-1;j--)
	{
		L.elem[j+1]=L.elem[j]; //Insert elements and subsequent elements move backward
	}
	strcpy(L.elem[i-1].no,str1);
	strcpy(L.elem[i-1].name,str2);
	L.elem[i-1].price=e; //Put the new element at the i-th position
	++L.length; //Table length plus 1
	return OK;
}

Status ListDelete(SqList &L,int i)
{
	if((i<1)||(i>L.length))
		return ERROR; //i value is illegal
	for(int j=i;j<=L.length-1;j++)
		L.elem[j-1]=L.elem[j]; //The element after the deleted element moves forward
	--L.length; // table length minus 1
	return OK;
}

int GetLength(SqList L)
{
	return L. length;
}

void OutputSqList(SqList L)
{
	for(int i=0;i<L.length;i++)
	{
printf("%7s%7s%5d\n",L.elem[i].no,L.elem[i].name,L.elem[i].price);
	}
}
```

LinkList.cpp:

```
#include "StdAfx.h"
#include "LinkList.h"

Status InitLinkList(LinkList &L)
{
```

```
    L=new LNode; //Generate a new node as the head node, use the head pointer L to point to the head node
    L->next=NULL; //The pointer field of the head node
    return OK;
}

Status GetElemLink(LinkList L,int i,ElemType &e)
{ //Get the value of the element according to the sequence number i in the singly linked list L with the head node, and use e to return the value of the i-th element in L
LNode *p;
p=L->next;int j=1; //Initialization, p points to the head node, and the initial value of counter j is assigned to 1
    while(p&&j<i) //Scan backward along the chain domain until p is empty or p points to the i-th element
    {
        p=p->next; //p points to the next node
        ++j; //Counter j is correspondingly increased by 1
    }
    if(!p||j>i) //i value is illegal i>n or i<=0
        return ERROR; //Get the data field of the i-th node
    e=p->data;
    return OK;
}

LNode *LinkLocateElem(LinkList L,char str[])
{ //Find the element whose value is e in the singly linked list L with the head node
LNode *p;
p=L->next; //Initialization, p points to the head node
while(p&&strcmp(p->data.name,str)!=0) //Scan the chain domain backward until p is empty or the data domain pointed to by p is equal to e
{
p=p->next; //p points to the next node
}
return p; //The search succeeds and returns the node address p whose value is e, and if the search fails, p is NULL
}

Status LinkListInsert(LinkList &L,int i,char str1[],char str2[],int &e)
{ //Insert a new node with the value e at the i-th position in the singly linked list L with the head node
LNode *p,*s;
int j;
p=L;j=0;
while(p&&(j<i-1))
{
p=p->next; //Find the i-1th node, p points to the node
++j;
}
    if(!p||j>i-1) //i>n+1 or i<1
        return ERROR;
s=new LNode; //generate new node*s
    strcpy(s->data.no,str1);
    strcpy(s->data.name, str2);
s->data.price=e; //Set the data field of node *s to e
s->next=p->next; //point the pointer field of node *s to node ai
p->next=s; //Point the pointer field of node *p to node *s
    return OK;
}
```

```
Status LinkListDelete(LinkList &L,int i)
{ //In the singly linked list L with the head node, delete the i-th element
	LNode *p,*q;
	int j=0;
	p=L;
	while((p->next)&&(j<i-1)) //Find the i-th node, p points to the node
	{
		p=p->next;
		++j;
	}
	if(!(p->next)||(j>i-1)) //When i>n or i<1, the deletion position is unreasonable
		return ERROR;
	q=p->next; //Temporarily save the address of the deleted node for release
	p->next=q->next; //Change the pointer field of the predecessor node of the deleted node
	delete q; //Release the space of the deleted node
	return OK;
}

void CreateList(LinkList &L,int n)
{
	LNode *p,*r;
	L=new LNode;
	L->next=NULL;
	r=L;
	for(int i=0;i<n;i++)
	{
		p=new LNode;
		scanf("%s%s%5d",p->data.no,p->data.name,&p->data.price);
		p->next=NULL;
		r->next=p;
		r=p;
	}
}

void OutputLinkList(LinkList L)
{
	LNode *p;
	p=L->next;
	while(p)
	{
		printf("%7s%7s%5d\n",p->data.no,p->data.name,p->data.price);
		p=p->next;
	}
}

int ListLength(LinkList L)
{
	LinkList p;
	p=L->next;
	int i=0;
	while(p)
	{
		i++;
		p=p->next;
	}
	return i;
}
```

StdAfx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
//	LInkListAPP.pch will be the pre-compiled header
//	stdafx.obj will contain the pre-compiled type information

#include "StdAfx.h"

// TODO: reference any additional headers you need in STDAFX.H
// and not in this file
```

LinkListAPP.cpp

```
// LInkListAPP.cpp : Defines the entry point for the console application.
//

#include "StdAfx.h"
#include "SqList.h"
#include "LinkList.h"

int main()
{
	int n,i;
	char program;
	printf("Welcome to the student information management system\n");
	printf("Please enter the following linear table operation:\n");
	printf("A: Execute sequential table operations:\n");
	printf("B: Perform linked list operation:\n");
	printf("Start to perform operation:");
	scanf("%c",&program);
	switch(program)
	{
	// The following performs the operations of the sequence table:
case 'A':
printf("Number of students:");
	scanf("%d",&n);
	printf("Please enter the student ID, name and grade:\n");
SqList L;
	InitList(L);
	L. length=n;
	for(i=0;i<n;i++)
	{
		scanf("%s%s%d",L.elem[i].no,L.elem[i].name,&L.elem[i].price); //Enter students one by one
according to the specified number of students information
	}
printf("Execute different functions below:\n");
	printf("1: Display the relevant information of all students in the student table one by one\n");
	printf("2: Enter the student's name to find other information of the student\n");
	printf("3: The corresponding student information (student number, name, grade) can be
returned according to the specified location\n");
	printf("4: Given a student information, insert it into the specified position in the table\n");
	printf("5:Delete the student record at the specified location\n");
	printf("6: The number of students in the statistical table\n");
	printf("Start to perform operation:");
	int control;
	while(scanf("%d",&control)!=EOF)
	{
switch(control)
		{
```

```
    case 1: //Display the relevant information of all students in the student table one by one
        printf("Please output the student ID, name and grade: \n");
        OutputSqList(L);
        break;
    case 2: //Enter the student's name to find other information about the student
        printf("Please enter the student's name: ");
        char StudentName[20];
        scanf("%s",&StudentName);
        int size;
    size=LocateElem3(L,StudentName); //The return value is the location of the student
        printf("Output student number and grade:\n");
        printf("%7s%5d\n",L.elem[size-1].no,L.elem[size-1].price);  //Output the student number
and grade of the searched student
        break;
case 3: //According to the specified location, the corresponding student information (student number,
name, grade) can be returned
        printf("Please enter the student's location:");
        int location;
        scanf("%d",&location);
        Student type;
        GetElem(L, location, type. no, type. name, type. price);
        printf("Output student ID, name, grade:\n");
        printf("%7s%7s%5d\n",type.no,type.name,type.price);
        break;
    case 4: //Given a student information, insert it into the specified position in the table;
        printf("Insert a new student information:");
        Student newtype;
        scanf("%s%s%d",&newtype.no,&newtype.name,&newtype.price);
        int NewLocation;
        scanf("%d",&NewLocation);
            int tm;
            tm=ListInsert(L,NewLocation,newtype.no,newtype.name,newtype.price);
            if(tm)
                printf("insert successfully\n");
            else
                printf("Insert failed\n");
            OutputSqList(L);
        break;
    case 5: //Delete the student record at the specified location;
        printf("Delete a student information:");
        int DeleteLocation;
        scanf("%d",&DeleteLocation);
            int tn;
            tn=ListDelete(L,DeleteLocation);
            if(tn)
                printf("Delete successfully\n");
            else
                printf("Delete failed\n");
        OutputSqList(L); //output the sequence list after deletion
        break;
    case 6: //The number of students in the statistics table
        printf("Statistics of the number of students:");
        int StudentLength;
        StudentLength=GetLength(L);
        printf("%d\n",StudentLength);
        break;
        }
    }
```

```
        break;
        case 'B':
        //The following operations are performed on the linked list:
printf("Number of students:");
        scanf("%d",&n);
        printf("Please enter the student ID, name and grade:\n");
        LinkList Q;
        CreateList(Q,n);
printf("Execute different functions below:\n");
printf("1: Display the relevant information of all students in the student table one by one\n");
        printf("2: Enter the student's name to find other information of the student\n");
        printf("3: The corresponding student information (student number, name, grade) can be
returned according to the specified location\n");
        printf("4: Given a student information, insert it into the specified position in the table\n");
        printf("5:Delete the student record at the specified location\n");
        printf("6: The number of students in the statistical table\n");
        printf("Start to perform operation:");
int choice;
        while(scanf("%d",&choice)!=EOF)
        {
        switch(choice)
                {
        case 1: //Display the relevant information of all students in the student table one by one
printf("Please output the student ID, name and grade: \n");
                OutputLinkList(Q);
                break;
        case 2: //Enter the student's name to find other information about the student
printf("Please enter the student's name: ");
                char StudentNameList[20];
                scanf("%s",&StudentNameList);
                LNode *ptsize;
                ptsize=LinkLocateElem(Q,StudentNameList);
printf("Output student number and grade:\n");
                printf("%7s%5d\n", ptsize->data.no, ptsize->data.price);
                break;
        case 3: //According to the specified location, the corresponding student information (student
number, name, grade) can be returned
printf("Please enter the student's location:");
                int location;
                scanf("%d",&location);
                Student typeLink;
                GetElemLink(Q, location, typeLink);
                printf("Output student ID, name, grade:\n");
                printf("%7s%7s%5d\n",typeLink.no,typeLink.name,typeLink.price);
                break;
        case 4: //Given a student information, insert it into the specified position in the table;
printf("Insert a new student information:");
                Student newtypeLink;
                scanf("%s%s%d", newtypeLink.no, newtypeLink.name, &newtypeLink.price);
                int NewLocationLinkList;
                scanf("%d",&NewLocationLinkList);
int cm;

        cm=LinkListInsert(Q,NewLocationLinkList,newtypeLink.no,newtypeLink.name,newtypeLink.price)
;
                        if(cm)
                                printf("insert successfully\n");
                        else
```

```
                printf("Insert failed\n");
            OutputLinkList(Q); //Output the linked list after insertion
            break;
        case 5: //Delete the student record at the specified location;
printf("Delete a student information:");
            int DeleteLocationLinkList;
            scanf("%d",&DeleteLocationLinkList);
                int cn;
cn=LinkListDelete(Q,DeleteLocationLinkList);
                if(cn)
                    printf("Delete successfully\n");
                else
                    printf("Delete failed\n");
            OutputLinkList(Q); //Output the deleted linked list
            break;
        case 6: //The number of students in the statistics table
printf("Statistics of the number of students:");
            int Length;
            Length=ListLength(Q);
            printf("%d\n",Length);
            break;
            }
        }
        break;
        }
        return 0;
}
```

Sqlist.h

```
//definition of sequence table
typedef struct
{
Student *elem; //Point to the base address of the data element
int length; //The current length of the linear table
}SqList;


//Common functions of the sequence table
Status InitList(SqList &L); //Initialization of the sequence table
Status GetElem(SqList L,int i,char str1[],char str2[],int &e); //Value of the sequence table
Status LocateElem1(SqList L,int e); //Search for the sequence table
Status LocateElem2(SqList L,char str[]);
Status LocateElem3(SqList L,char str[]);
Status ListInsert(SqList &L,int i,char str1[],char str2[],int &e); //Insertion of sequence table
Status ListDelete(SqList &L,int i); //Delete the sequence list
int GetLength(SqList L); //length of sequence list
void OutputSqList(SqList L); //output of sequence table
```

LinkList.h

```
//Definition of linked list
typedef struct LNode
{
    ElemType data; //The data field of the node
    struct LNode *next; //pointer field of node
}LNode,*LinkList; //LinkList is the pointer type to the structure LNode

//Common functions for linked lists
```

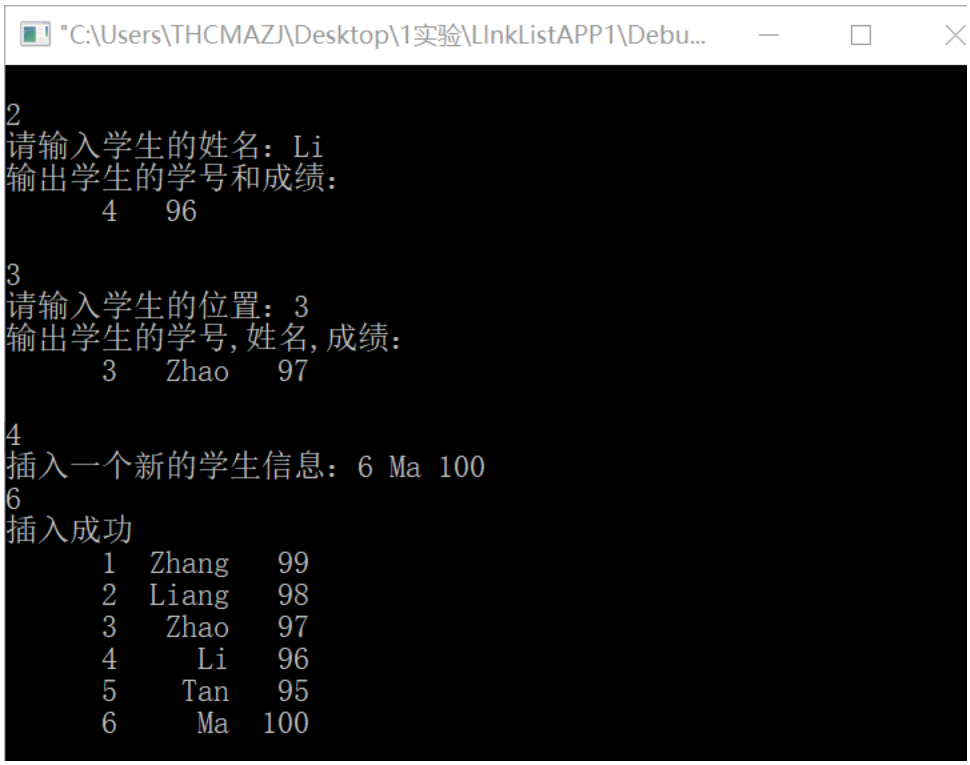
```
Status InitLinkList(LinkList &L); //Initialization of the linked list
Status GetElemLink(LinkList L,int i,ElemType &e); //The value of the linked list
LNode *LinkLocateElem(LinkList L,char str[]); //Lookup of linked list
Status LinkListInsert(LinkList &L,int i,char str1[],char str2[],int &e); //Insertion of linked list
Status LinkListDelete(LinkList &L,int i); //Delete the linked list
void CreateList(LinkList &L,int n); //Creation of linked list
int ListLength(LinkList L);
void OutputLinkList(LinkList L); //The output of the linked list
```

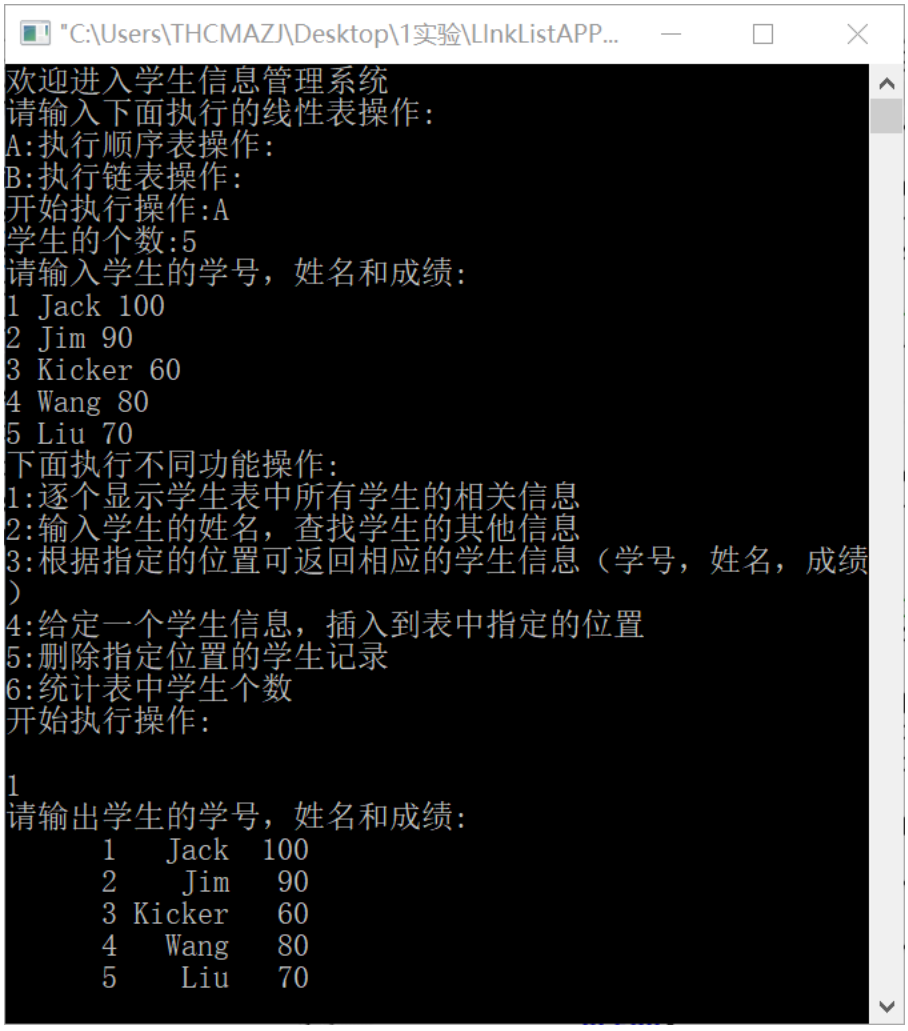
StdAfx.h

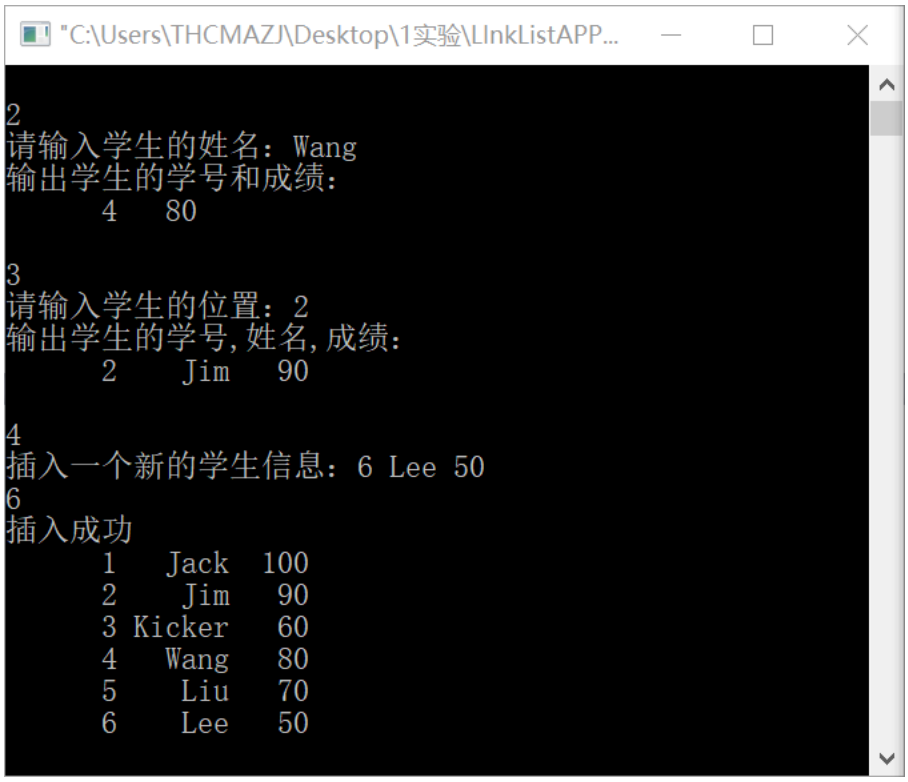
```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#if !defined(AFX_STDAFX_H__CC442F04_527D_4B85_8BEE_0B10560AB8BB__INCLUDED_)
#define AFX_STDAFX_H__CC442F04_527D_4B85_8BEE_0B10560AB8BB__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define WIN32_LEAN_AND_MEAN		// Exclude rarely-used stuff from Windows headers

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define OK 1
#define ERROR 0
#define OVERFLOW -2
#define MAXSIZE 100

typedef int Status;

//Definition of student information:
typedef struct
{
char no[8]; //8 student number
char name[20]; //name
int price; //score
}Student;


typedef Student ElemType;

// TODO: reference additional headers your program requires here

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_STDAFX_H__CC442F04_527D_4B85_8BEE_0B10560AB8BB__INCLUDED_)
```

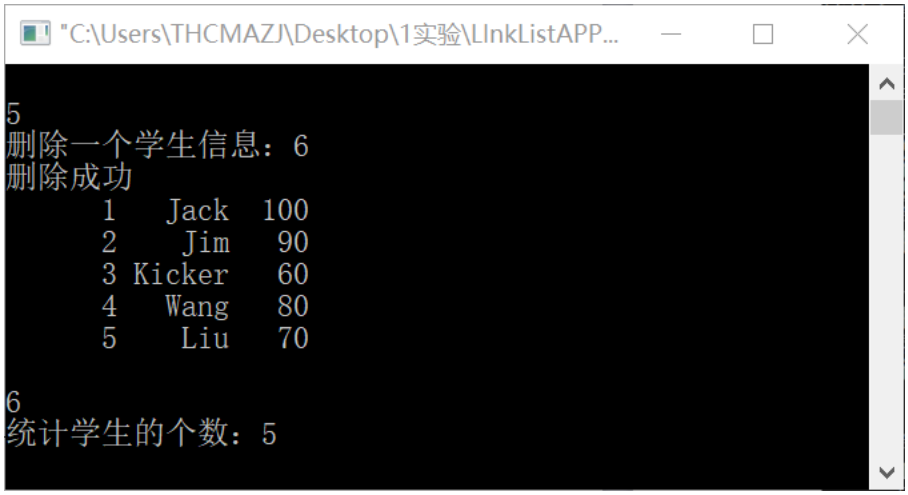
Test Results:

Sequence table:

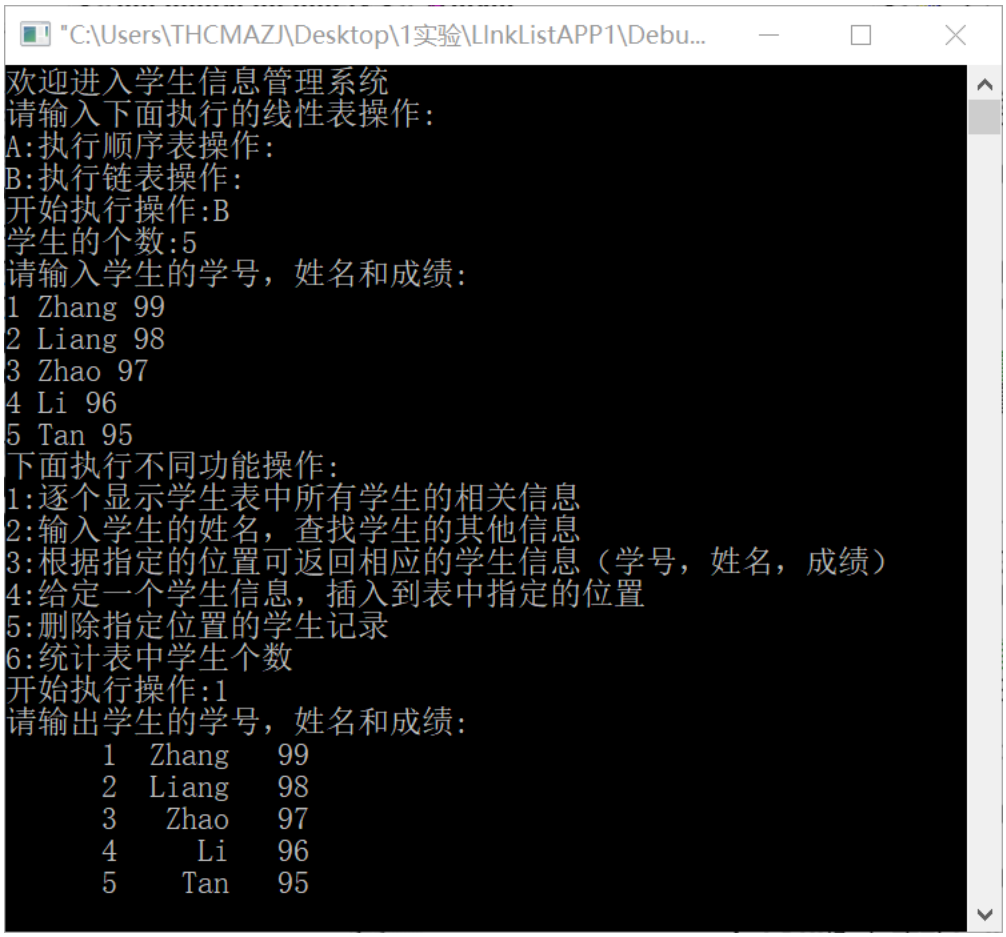


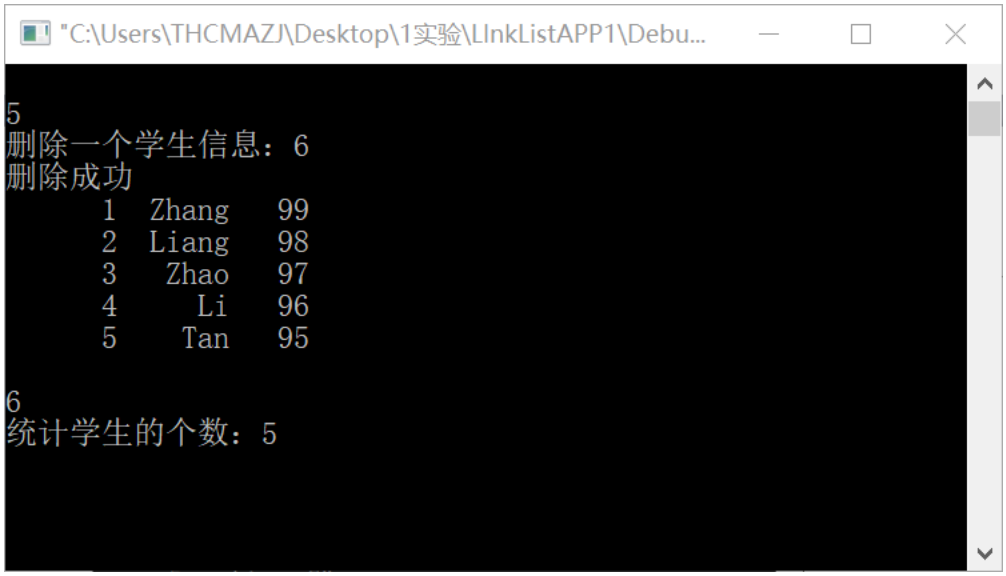

"C:\Users\THCMAZJ\Desktop\1实验\LInkListAPP...
欢迎进入学生信息管理系统
请输入下面执行的线性表操作:
A:执行顺序表操作:
B:执行链表操作:
开始执行操作:A
学生的个数:5
请输入学生的学号，姓名和成绩:
1 Jack 100
2 Jim 90
3 Kicker 60
4 Wang 80
5 Liu 70
下面执行不同功能操作:
1:逐个显示学生表中所有学生的相关信息
2:输入学生的姓名，查找学生的其他信息
3:根据指定的位置可返回相应的学生信息（学号，姓名，成绩
)
4:给定一个学生信息，插入到表中指定的位置
5:删除指定位置的学生记录
6:统计表中学生个数
开始执行操作:
1
请输出学生的学号，姓名和成绩:
1 Jack 100
2 Jim 90
3 Kicker 60
4 Wang 80
5 Liu 70


"C:\Users\THCMAZJ\Desktop\1实验\LInkListAPP...
2
请输入学生的姓名: Wang
输出学生的学号和成绩:
4 80
3
请输入学生的位置: 2
输出学生的学号,姓名,成绩:
2 Jim 90
4
插入一个新的学生信息: 6 Lee 50
6
插入成功
1 Jack 100
2 Jim 90
3 Kicker 60
4 Wang 80
5 Liu 70
6 Lee 50


"C:\Users\THCMAZJ\Desktop\1实验\LInkListAPP...
5
删除一个学生信息：6
删除成功
1 Jack 100
2 Jim 90
3 Kicker 60
4 Wang 80
5 Liu 70
6
统计学生的个数：5

linked list:


"C:\Users\THCMAZJ\Desktop\1实验\LInkListAPP1\Debu...
欢迎进入学生信息管理系统
请输入下面执行的线性表操作:
A:执行顺序表操作:
B:执行链表操作:
开始执行操作:B
学生的个数:5
请输入学生的学号，姓名和成绩:
1 Zhang 99
2 Liang 98
3 Zhao 97
4 Li 96
5 Tan 95
下面执行不同功能操作:
1:逐个显示学生表中所有学生的相关信息
2:输入学生的姓名，查找学生的其他信息
3:根据指定的位置可返回相应的学生信息（学号，姓名，成绩）
4:给定一个学生信息，插入到表中指定的位置
5:删除指定位置的学生记录
6:统计表中学生个数
开始执行操作:1
请输出学生的学号，姓名和成绩:
1 Zhang 99
2 Liang 98
3 Zhao 97
4 Li 96
5 Tan 95


"C:\Users\THCMAZJ\Desktop\1实验\LInkListAPP1\Debu...
5
删除一个学生信息：6
删除成功
1 Zhang 99
2 Liang 98
3 Zhao 97
4 Li 96
5 Tan 95
6
统计学生的个数：5