**一、实验目的**

1．熟悉和回顾编程的一般方法

2．循序渐进编写实际编程问题

3．进一步学习和提高综合编程能力和解决编程中出现的问题的能力

**二、实验内容**

**１.　设计和完成：简单学生成绩管理信息系统**

学生成绩信息包括：学号，姓名，课程名，平时成绩，实验成绩，考试成绩，总评成绩。

**２.　要求实现如下功能：**

1. 能够实现学生成绩信息的插入、删除和修改；
2. 能够实现各种查询（分别根据学生学号、姓名、课程名称等）；
3. 能够实现按照考试成绩、总评成绩进行排序；
4. 能够查询某门课程的最高分、最低分并输出相应学生信息；
5. 能够查询某门课程的优秀率（90 分及以上）、不及格率；

**３.　说明：**

1）次试验是综合设计实验，用6个学时（3次实验课）完成 ，程序的最终运行结果界面如：图10 功能模块界面，根据输入序号选择执行相应功能（仅供参考）。

2）也可以自己选定其它类似题目，功能要求参考学生成绩管理系统。

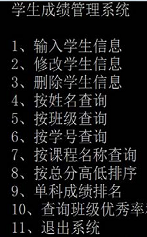
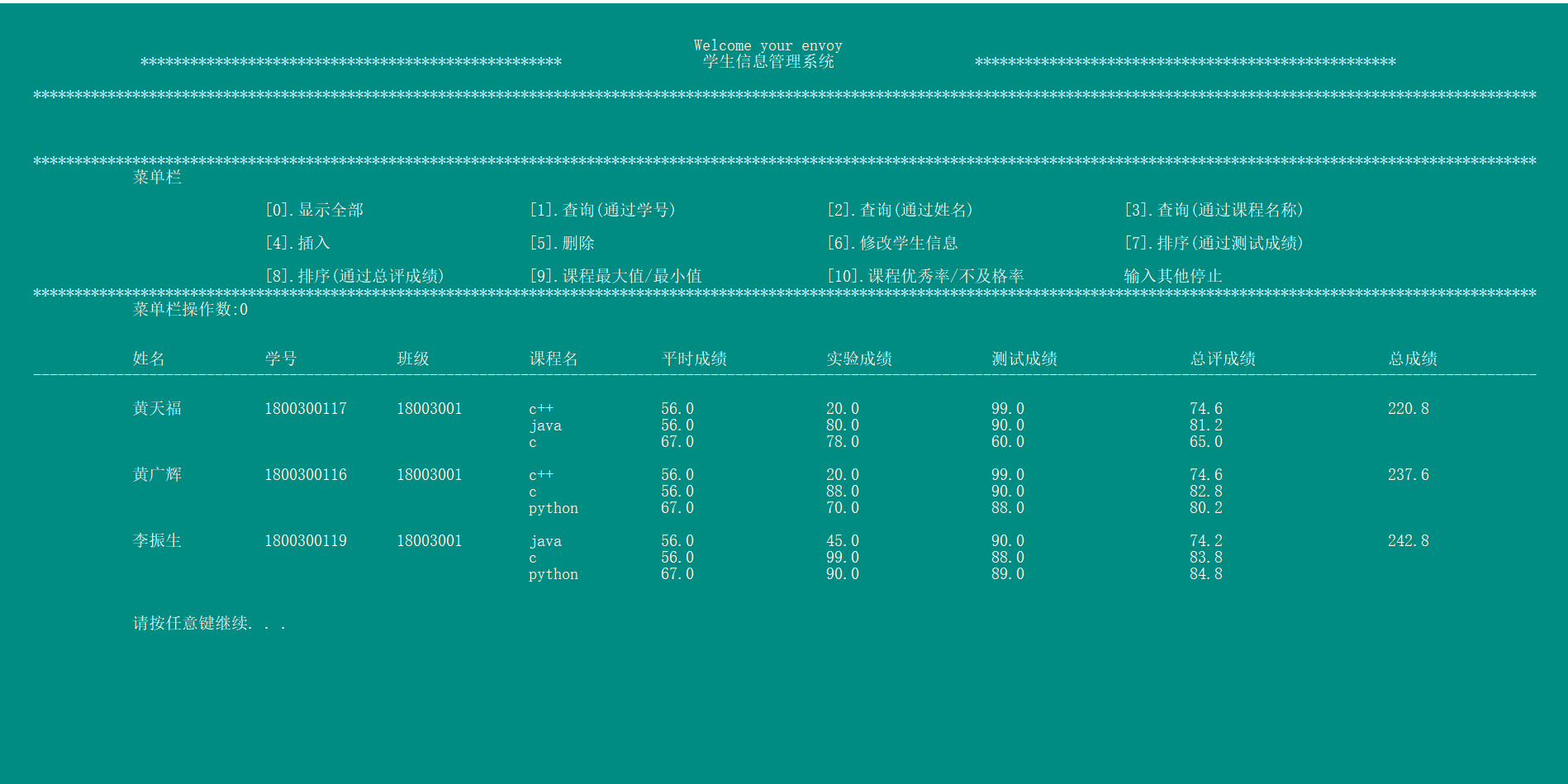


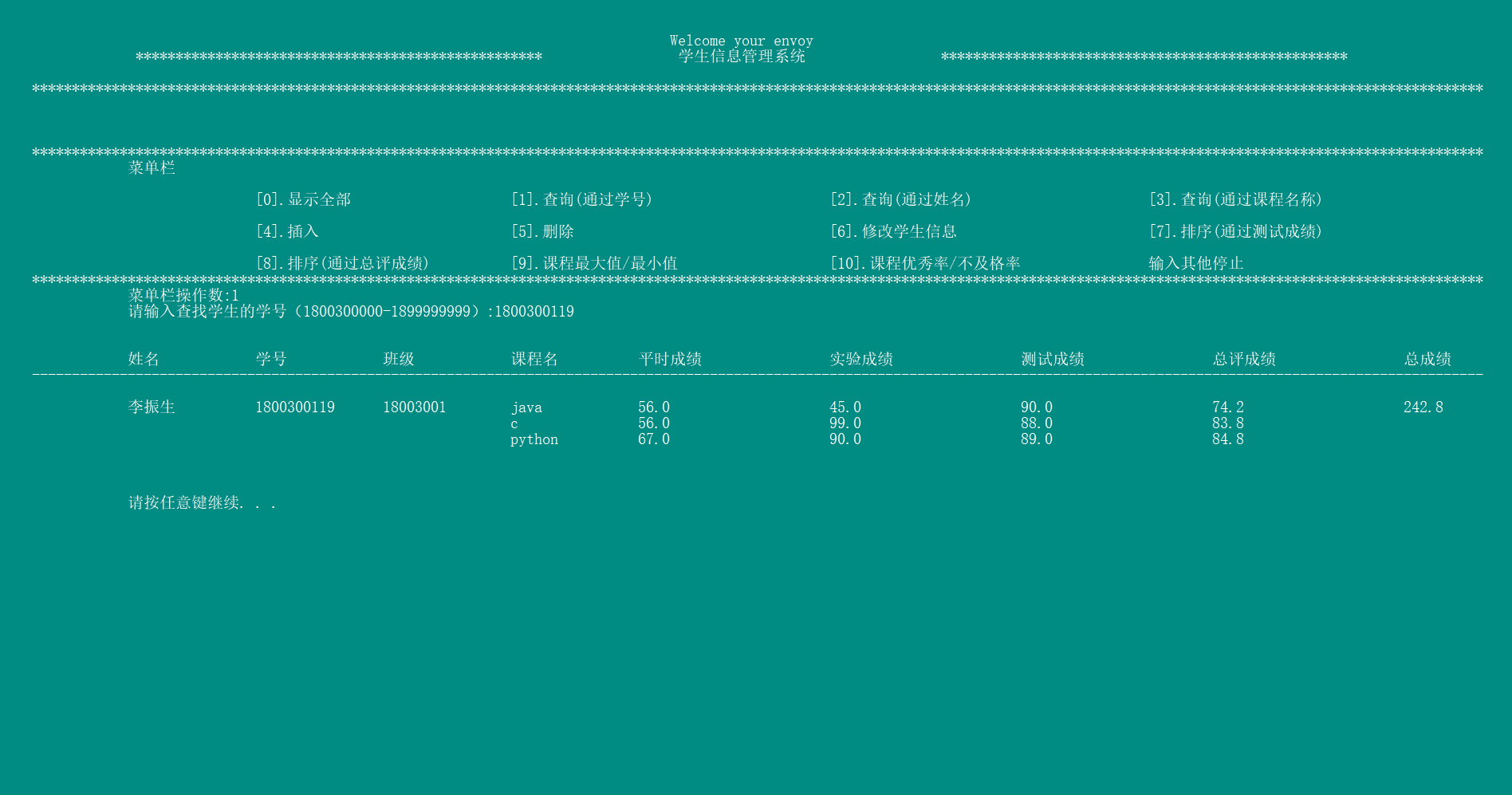
图10 功能模块界面

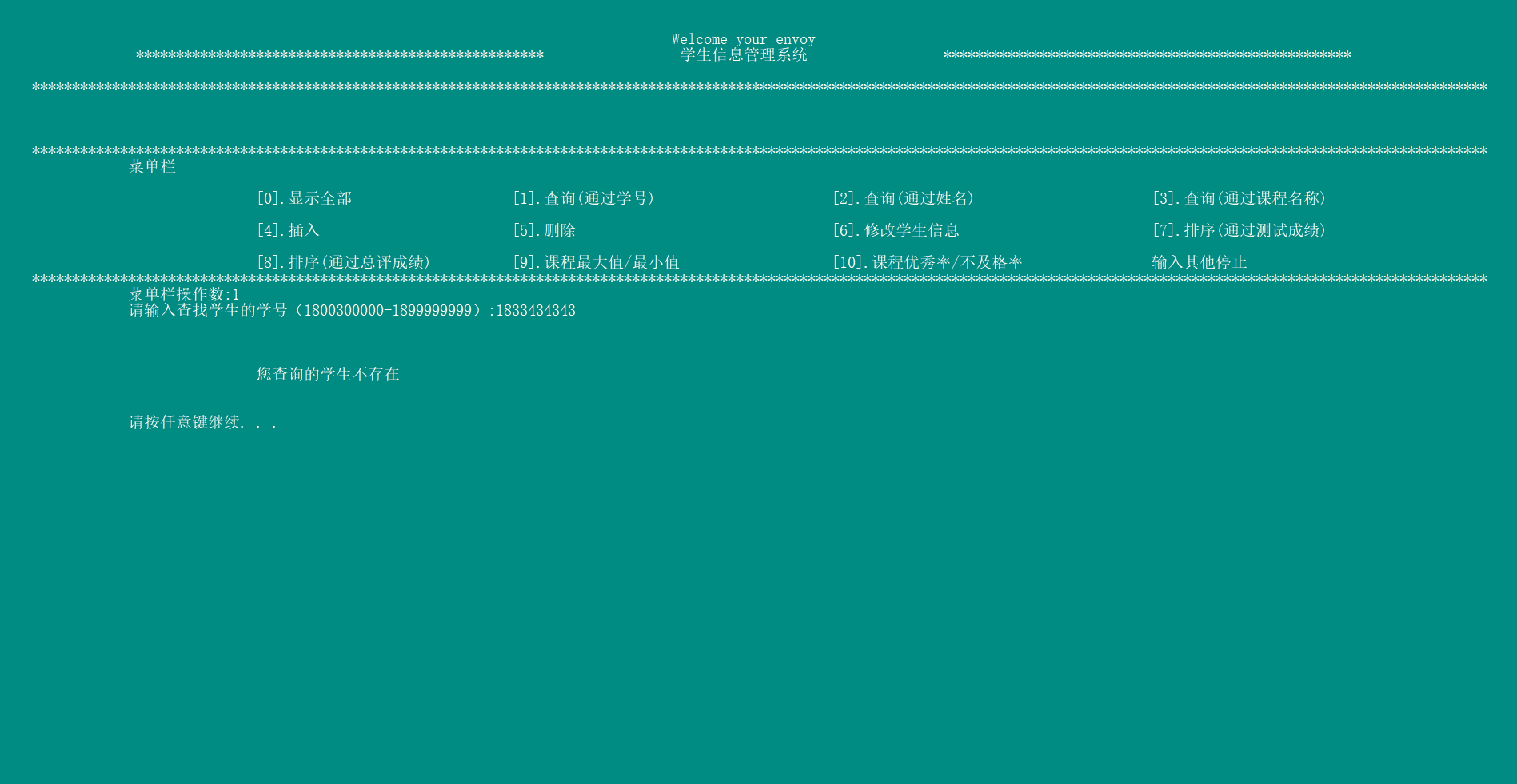
截图:

执行菜单[0]（显示全部）

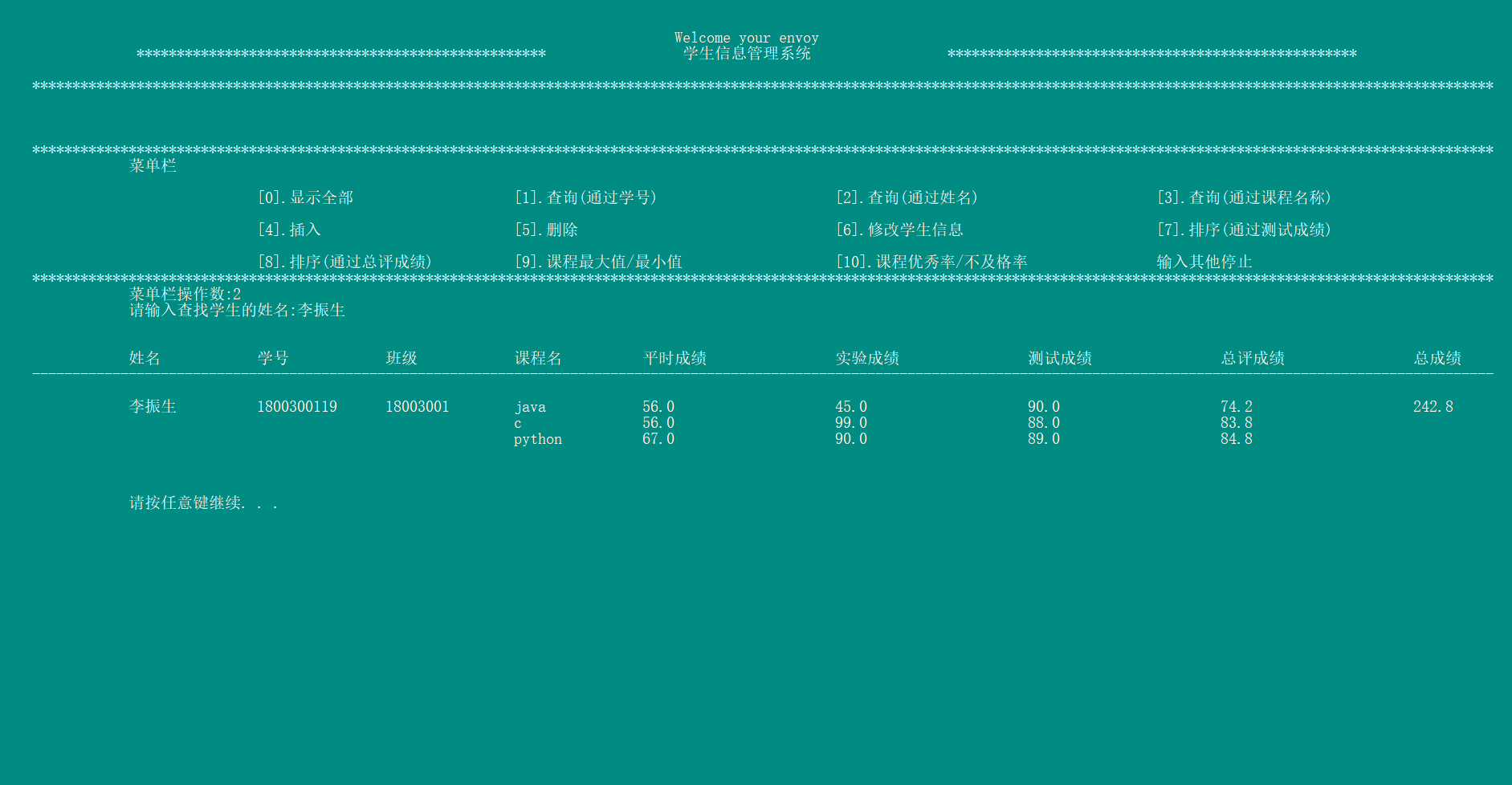


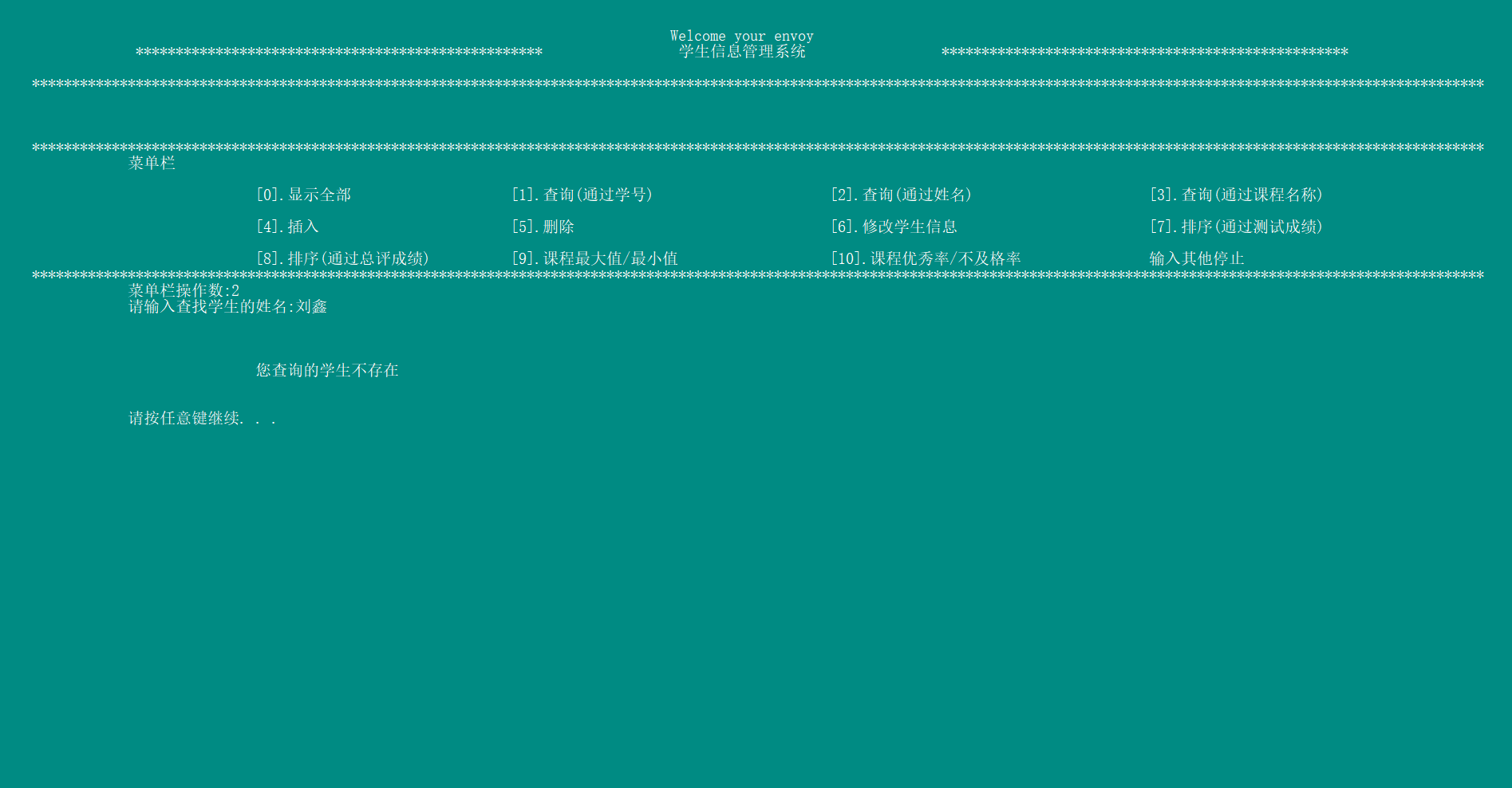
执行菜单[1]（查询(通过学号)）



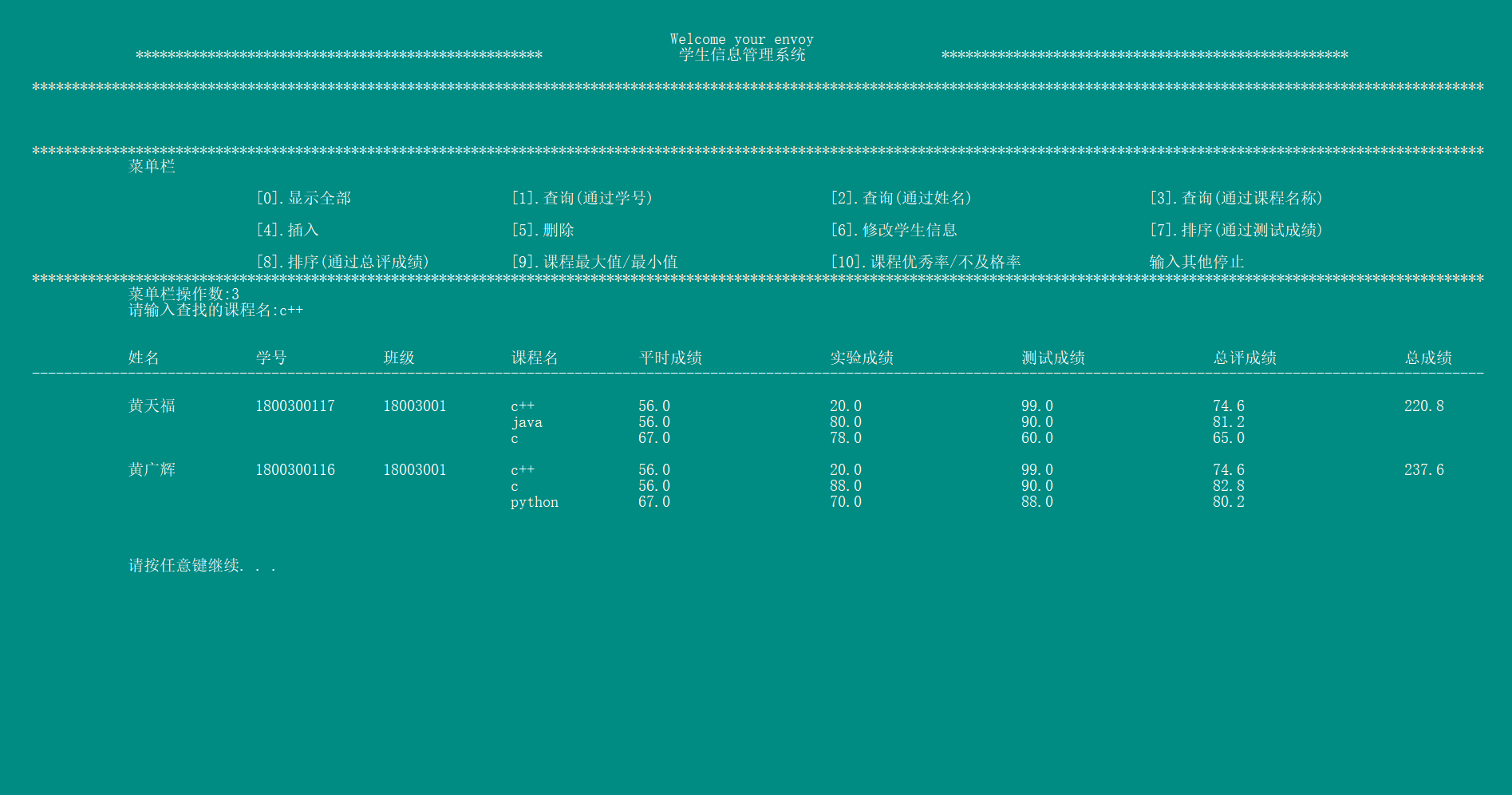


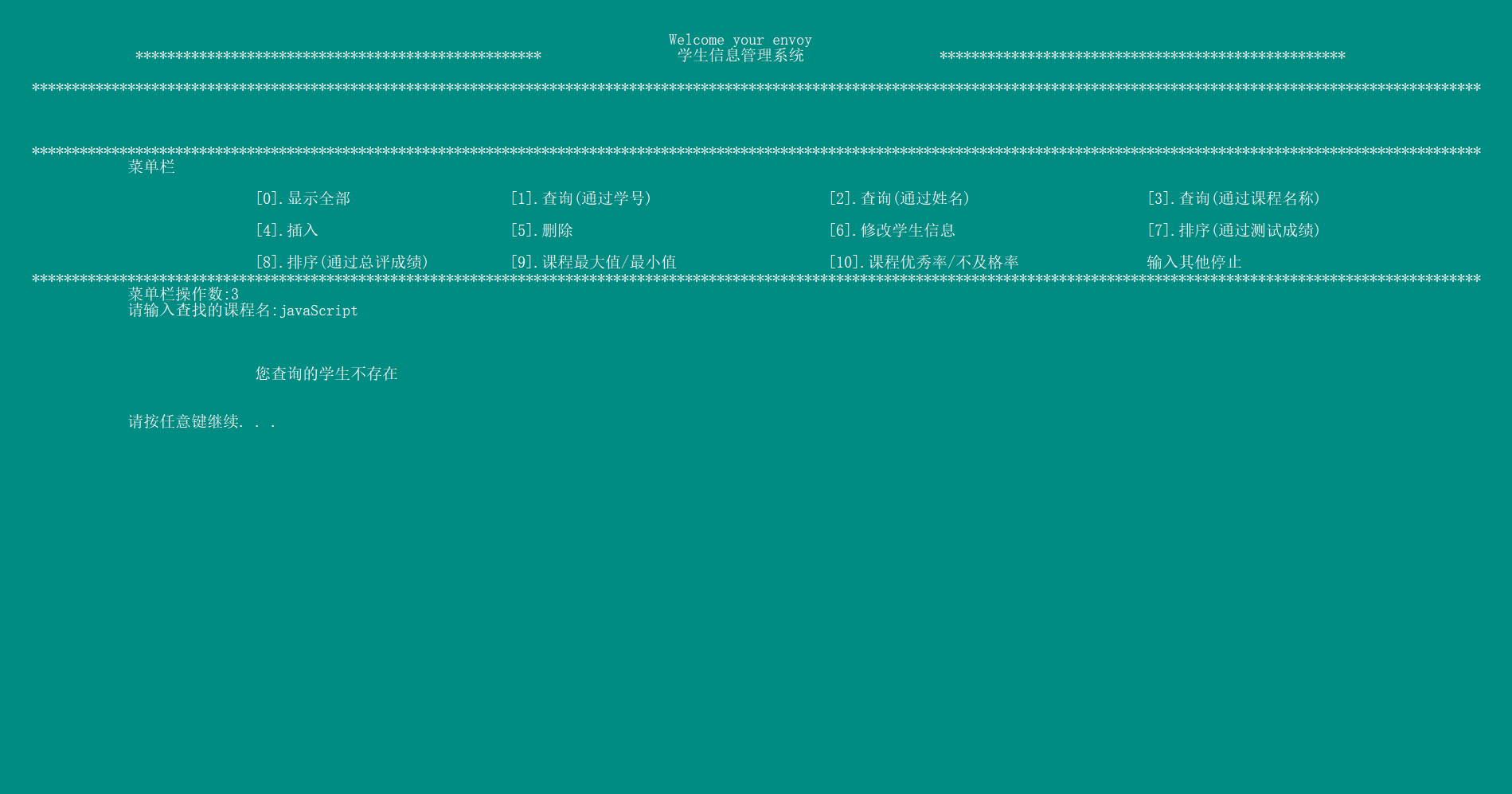
执行菜单[2]（查询(通过姓名)）



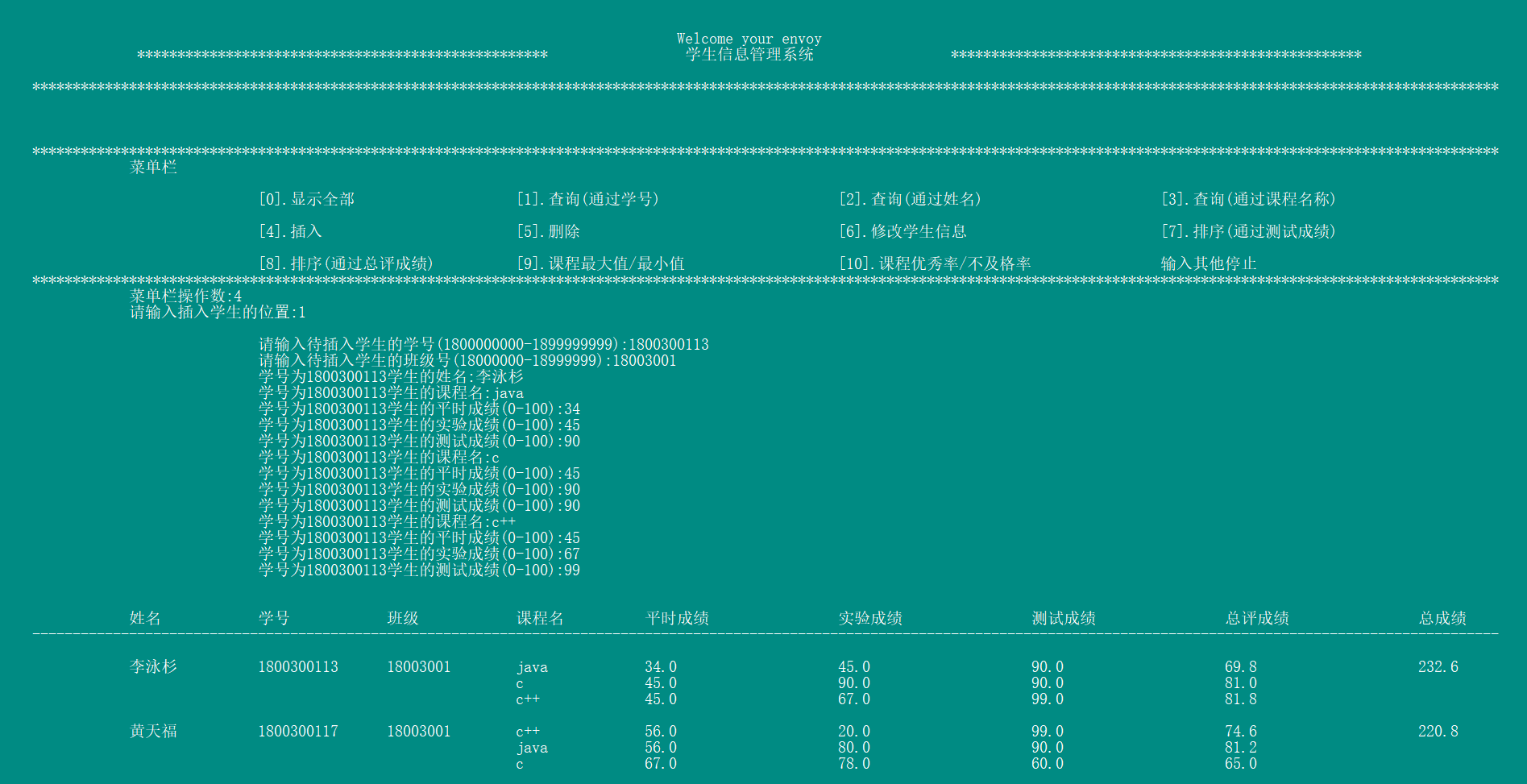


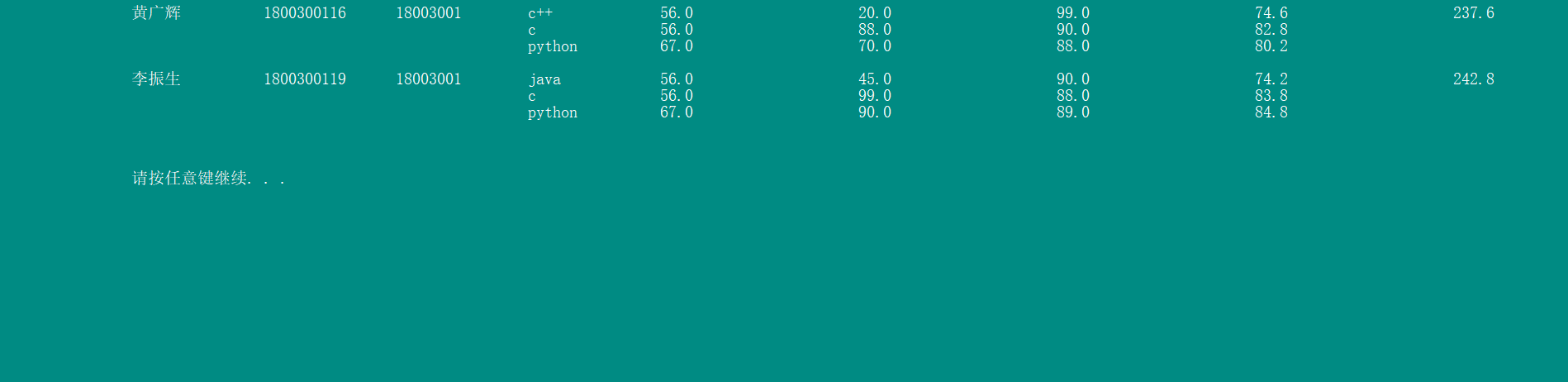
执行菜单[3]（查询(通过课程名称)）



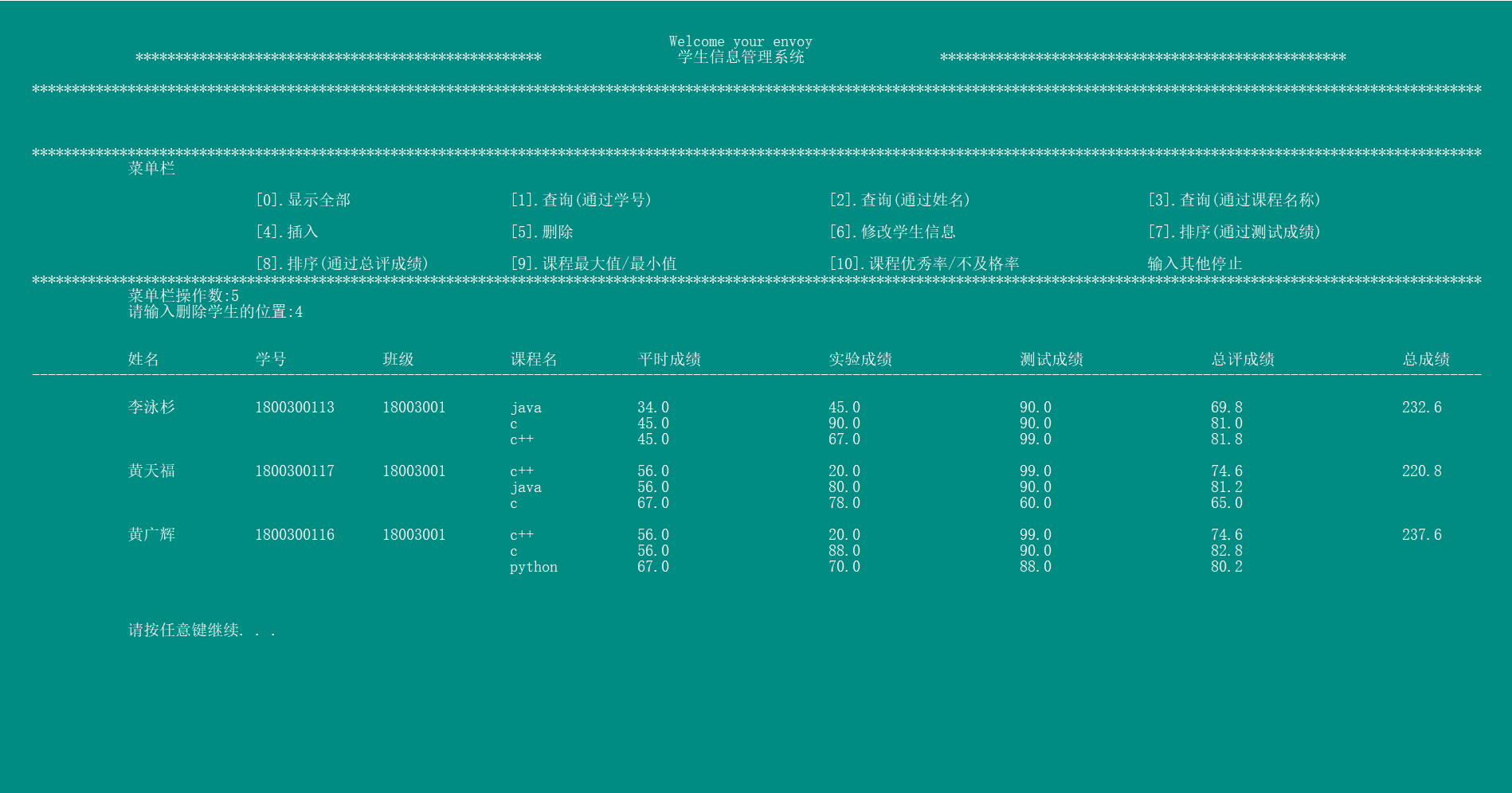


执行菜单[4]（插入）

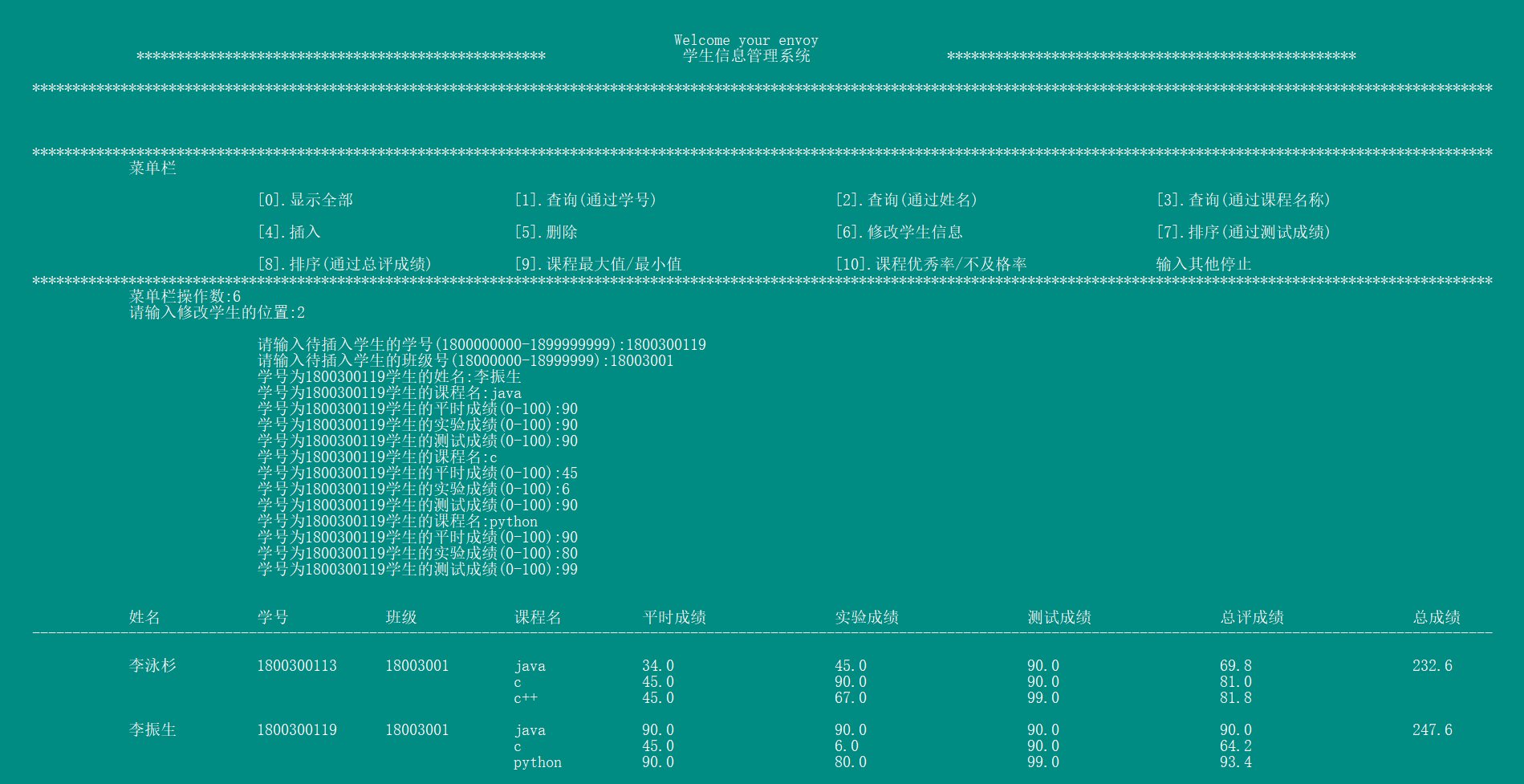


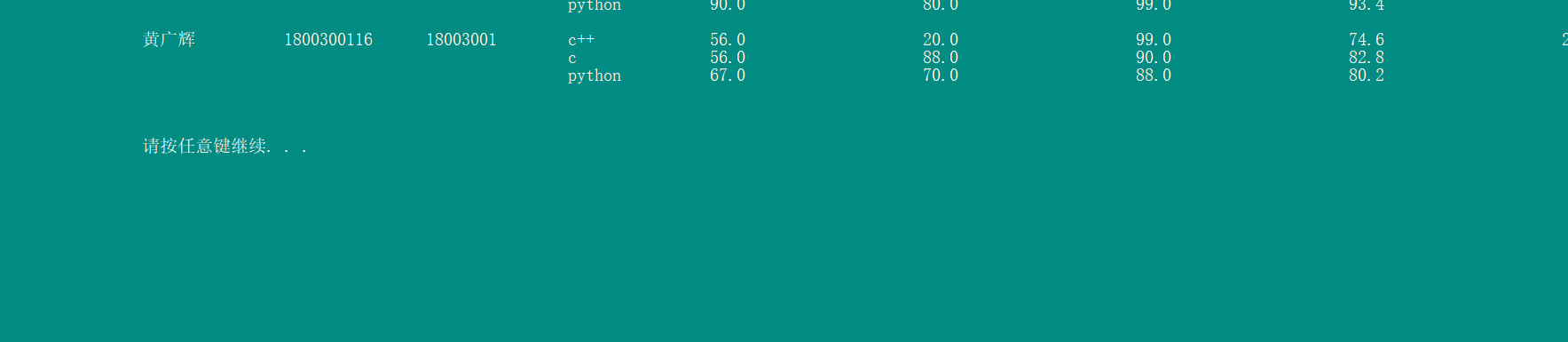


执行菜单[5]（删除）

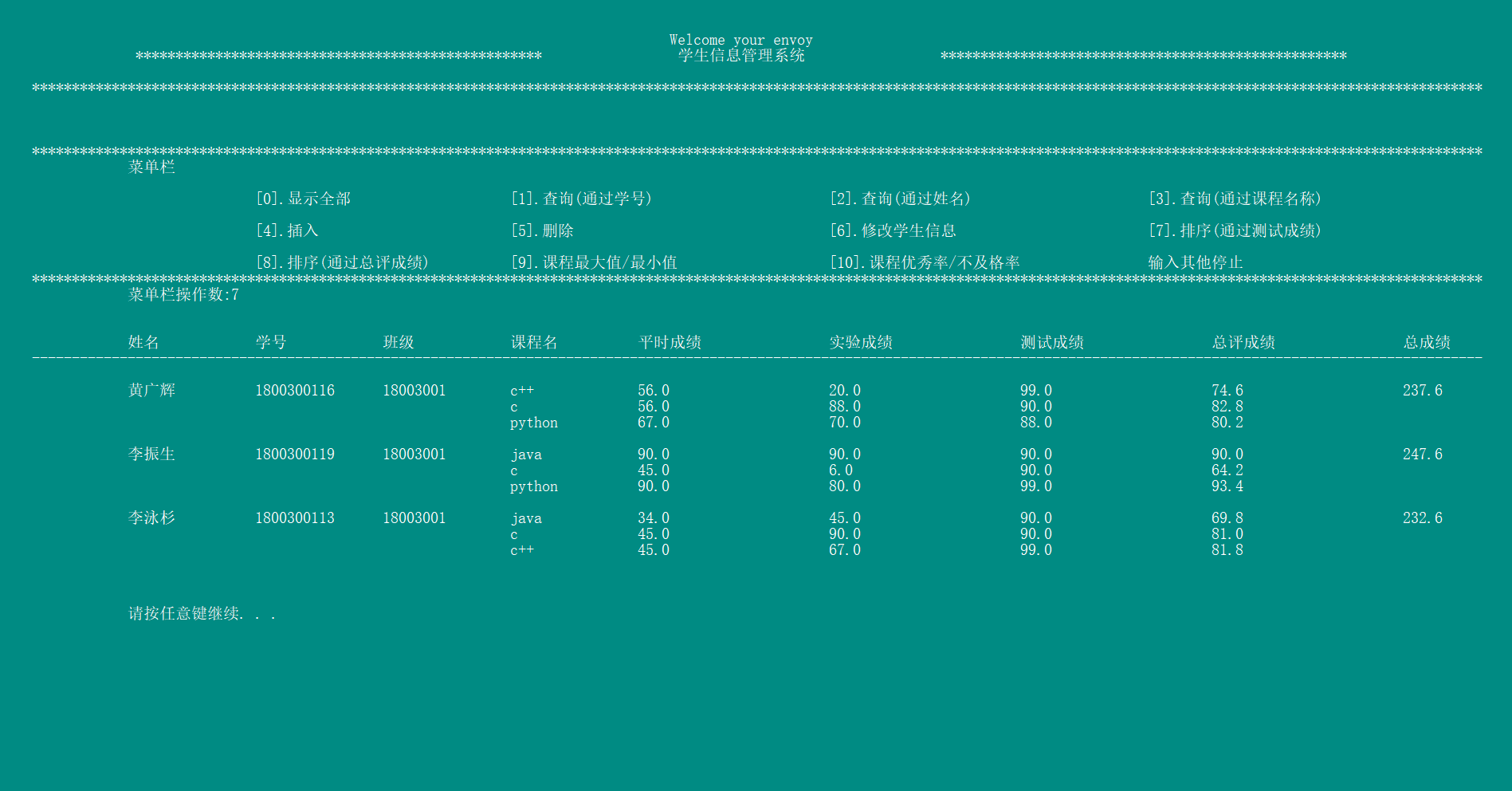


执行菜单[6]（修改学生信息）

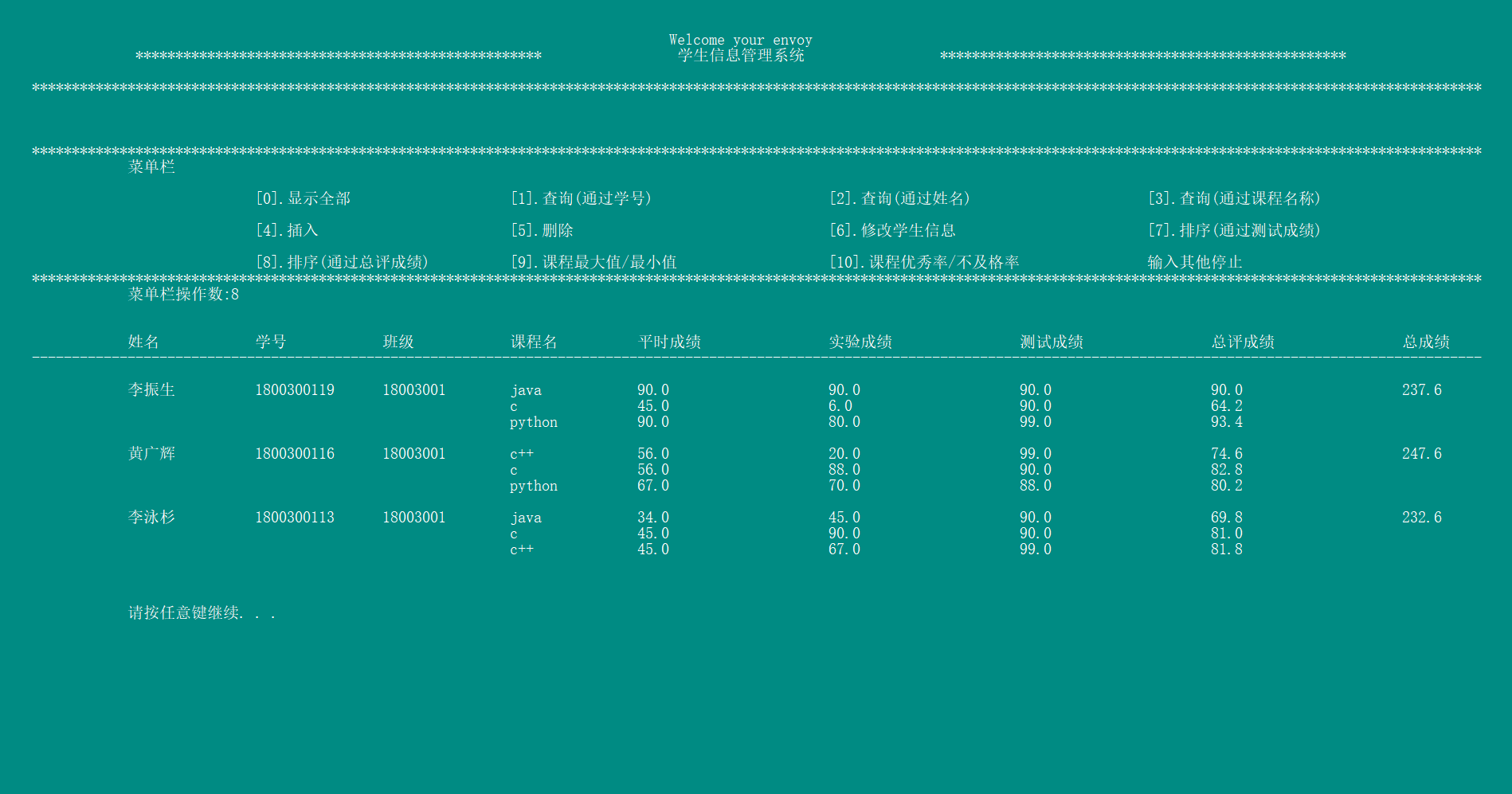




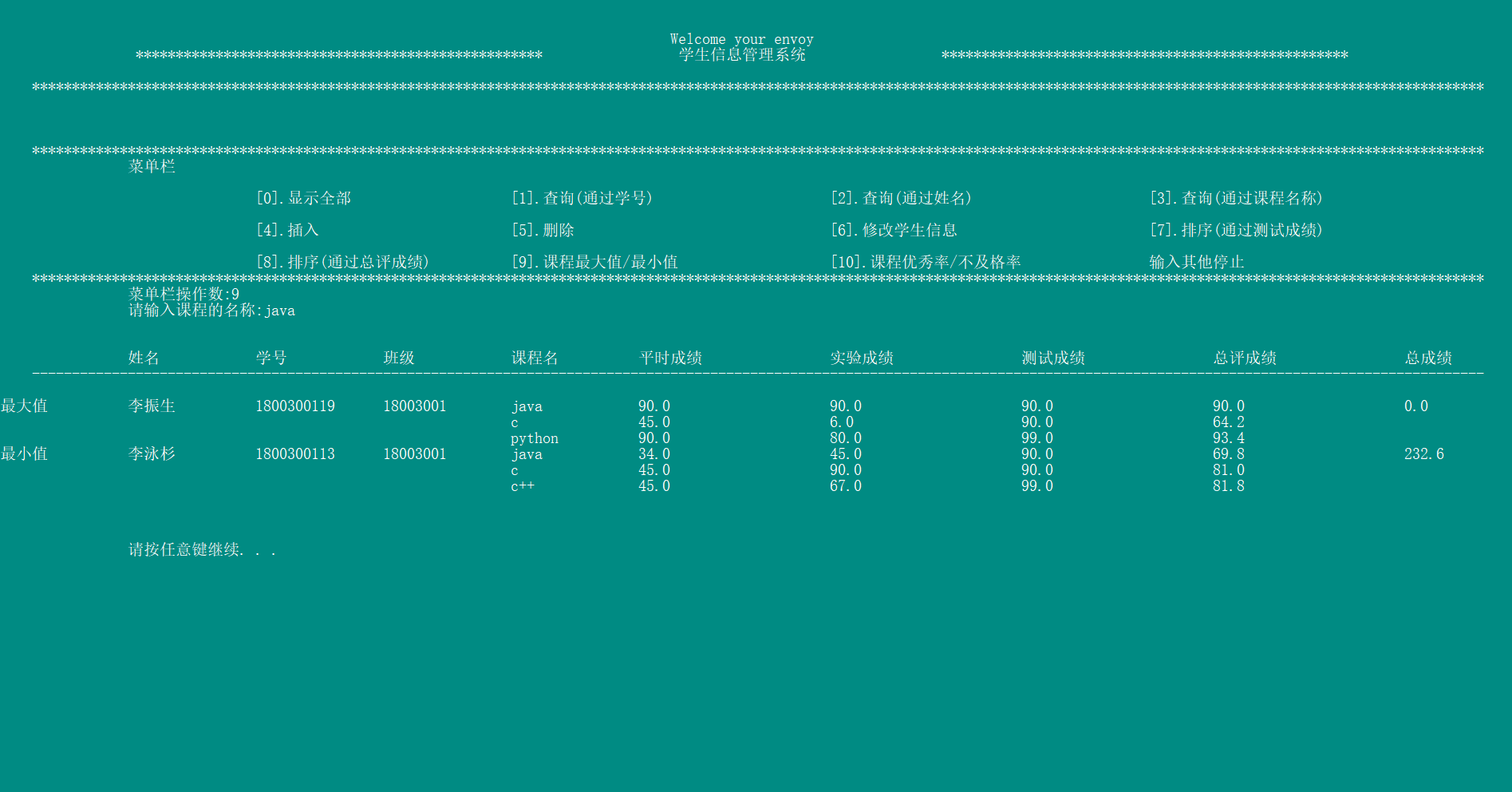
执行菜单[7]（排序(通过测试成绩)）



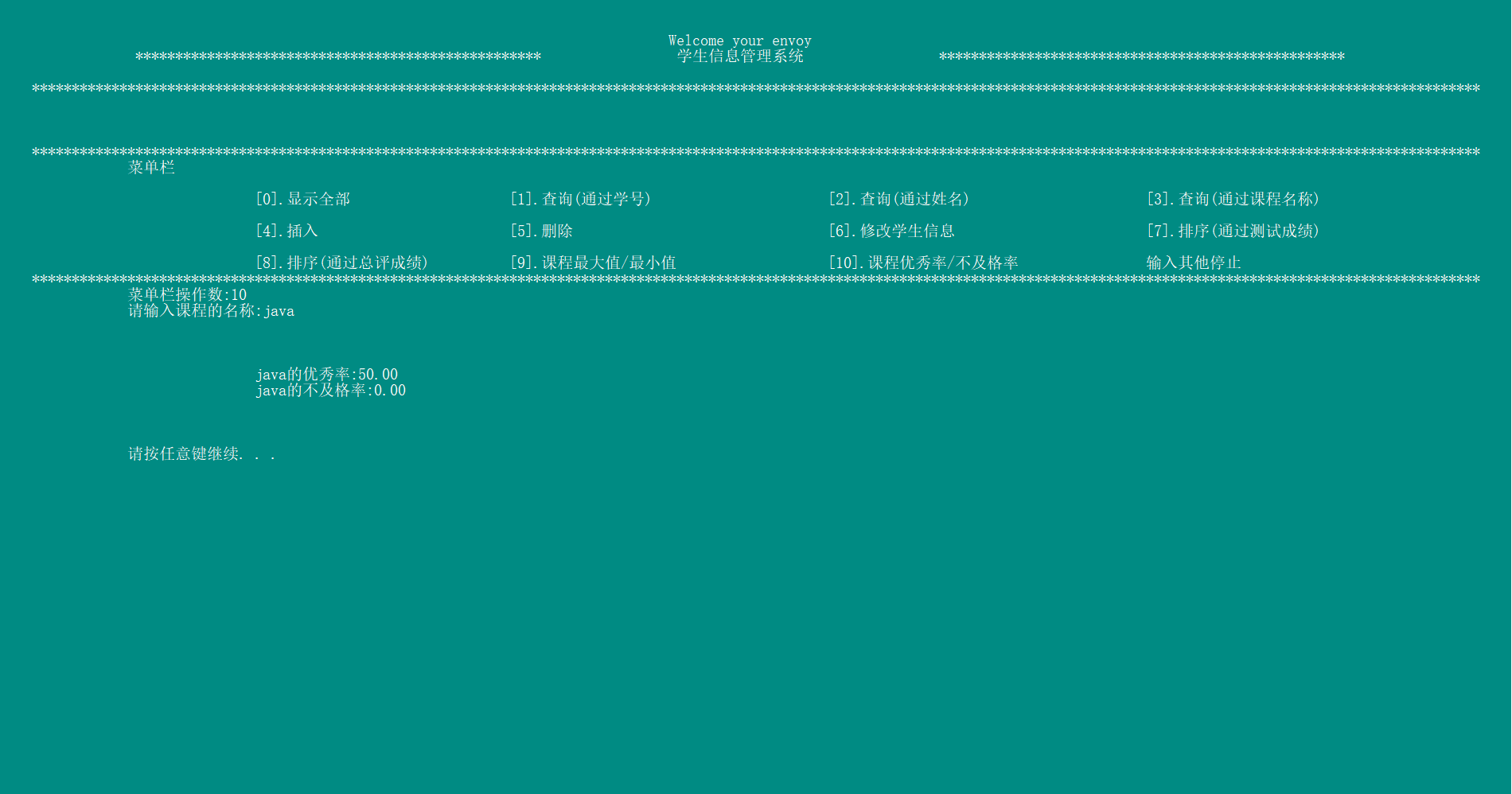
执行[8]（排序(通过总评成绩)）



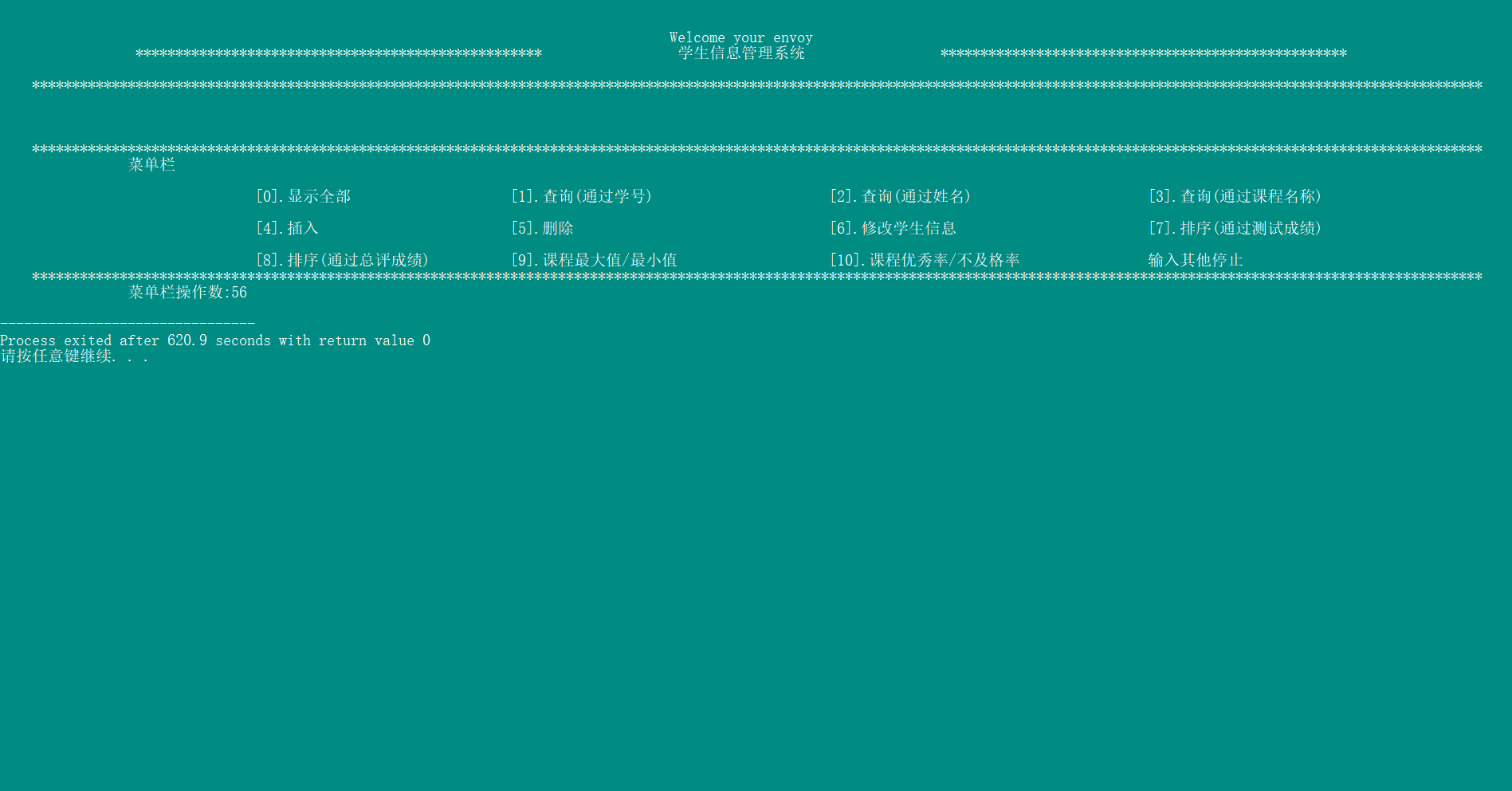
执行菜单[9]（课程最大值/最小值）



执行菜单[10]（课程优秀率/不及格率）



执行其他(输入其他停止)



代码:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct stud\_node//定义结构体

{

char num[20];

char name[20];

char className[3][20];

double regularScore[3];

double experimentalScore[3];

double testScore[3];

double totalScore[3];

double totalScoreAll;

char grade[20];

struct stud\_node \*next;

}stud[100];

void displayAll(struct stud\_node \*head)//输出全部信息的函数

{

struct stud\_node \*p;

p=(struct stud\_node\*)malloc(sizeof(struct stud\_node));

printf("\n\n\t\t姓名\t\t学号\t\t班级\t\t课程名\t\t平时成绩\t\t实验成绩\t\t测试成绩\t\t总评成绩\t\t总成绩");//输出相应的格式

printf("\n --------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n");

for(p=head;p!=NULL;p=p->next)//输出信息

{

printf("\n\t\t%-10s\t%-10s\t%-10s\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->name,p->num,p->grade,p->className[0],p->regularScore[0],p->experimentalScore[0],p->testScore[0],p->totalScore[0],p->totalScoreAll);

for(int i=1;i<3;i++)

{

printf("\t\t\t\t\t\t\t\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->className[i],p->regularScore[i],p->experimentalScore[i],p->testScore[i],p->totalScore[i]);

}

}

printf("\n\n\t\t");

}

void search1(struct stud\_node \*head,char n[])//通过学号寻找

{

int flag=1;

struct stud\_node \*p;

p=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

for(p=head;p!=NULL;p=p->next)

{

if(strcmp(p->num,n)==0)//比较输入的学号和链表中的学号是否相等

{

printf("\n\n\t\t姓名\t\t学号\t\t班级\t\t课程名\t\t平时成绩\t\t实验成绩\t\t测试成绩\t\t总评成绩\t\t总成绩");//打印相关的信息

printf("\n --------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n");

printf("\n\t\t%-10s\t%-10s\t%-10s\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->name,p->num,p->grade,p->className[0],p->regularScore[0],p->experimentalScore[0],p->testScore[0],p->totalScore[0],p->totalScoreAll);

for(int i=1;i<3;i++)

{

printf("\t\t\t\t\t\t\t\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->className[i],p->regularScore[i],p->experimentalScore[i],p->testScore[i],p->totalScore[i]);

}

flag=0;

printf("\n\n\n\t\t");

break;

}

}

//当学号不存在时

if(flag){

printf("\n\n\n\t\t\t\t您查询的学生不存在\n\n\n\t\t");

}

}

void search2(struct stud\_node \*head,char n[])//通过姓名查找

{

int flag=1;

struct stud\_node \*p;

p=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

for(p=head;p!=NULL;p=p->next)

{

if(strcmp(p->name,n)==0)//比较输入的姓名和链表中的姓名是否相等

{

printf("\n\n\t\t姓名\t\t学号\t\t班级\t\t课程名\t\t平时成绩\t\t实验成绩\t\t测试成绩\t\t总评成绩\t\t总成绩");//输出相应的格式和信息

printf("\n --------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n");

printf("\n\t\t%-10s\t%-10s\t%-10s\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->name,p->num,p->grade,p->className[0],p->regularScore[0],p->experimentalScore[0],p->testScore[0],p->totalScore[0],p->totalScoreAll);

for(int i=1;i<3;i++)

{

printf("\t\t\t\t\t\t\t\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->className[i],p->regularScore[i],p->experimentalScore[i],p->testScore[i],p->totalScore[i]);

}

flag=0;

printf("\n\n\n\t\t");

break;

}

}

//当输入的姓名不存在时

if(flag){

printf("\n\n\n\t\t\t\t您查询的学生不存在\n\n\n\t\t");

}

}

void search3(struct stud\_node \*head,char n[])//通过课程名查找

{

int flag=1,i=0;

struct stud\_node \*p;

p=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

for(p=head;p!=NULL;p=p->next)

{

for(int j=0;j<3;j++)

{

if(strcmp(p->className[j],n)==0)

{

if(i==0)

{

i++;

printf("\n\n\t\t姓名\t\t学号\t\t班级\t\t课程名\t\t平时成绩\t\t实验成绩\t\t测试成绩\t\t总评成绩\t\t总成绩");//输出相应的格式和相关信息

printf("\n --------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n");

}

printf("\n\t\t%-10s\t%-10s\t%-10s\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->name,p->num,p->grade,p->className[0],p->regularScore[0],p->experimentalScore[0],p->testScore[0],p->totalScore[0],p->totalScoreAll);

for(int i=1;i<3;i++)

{

printf("\t\t\t\t\t\t\t\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->className[i],p->regularScore[i],p->experimentalScore[i],p->testScore[i],p->totalScore[i]);

}

flag=0;

}

}

}

if(i!=0)

{

printf("\n\n\n\t\t");

}

//当不存在时

if(flag){

printf("\n\n\n\t\t\t\t您查询的学生不存在\n\n\n\t\t");

}

}

void insert(struct stud\_node\* &head,int n)//插入学生的操作

{

FILE \*fp;

if((fp=fopen("score.dat","wb"))==NULL)

{

printf("cannot open file\n");

return;

}

int score;

char num[20],name[20],className[20],grade[20];

double regularScore,experimentalScore,testScore,totalScore;

struct stud\_node \*p,\*q;

p=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

q=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

q=head;

printf("\n\t\t\t\t请输入待插入学生的学号(1800000000-1899999999):");//根据提示输入相关的信息

scanf("%s",&num);

while(!(strcmp(num,"1800000000")>=0&&strcmp(num,"1899999999")<=0))

{

printf("\t\t\t\t输入错误，请重新输入该学生的学号(1800000000-1899999999):");

scanf("%s",&num);

}

printf("\t\t\t\t请输入待插入学生的班级号(18000000-18999999):");

scanf("%s",grade);

while(!(strcmp(grade,"1800000")>=0&&strcmp(grade,"18999999")<=0))

{

printf("\t\t\t\t输入错误，请重新输入该学生的班级号(18000000-18999999):");

scanf("%s",grade);

}

printf("\t\t\t\t学号为%s学生的姓名:",num);

scanf("%s",name);

p->totalScoreAll=0;

for(int i=0;i<3;i++)

{

printf("\t\t\t\t学号为%s学生的课程名:",num);

scanf("%s",className);

printf("\t\t\t\t学号为%s学生的平时成绩(0-100):",num);

scanf("%lf",&regularScore);

while(!(regularScore>=0&&regularScore<=100)){

printf("\t\t\t\t输入错误，请重新输入该学生的平时成绩(0-100):");

scanf("%lf",&regularScore);

}

printf("\t\t\t\t学号为%s学生的实验成绩(0-100):",num);

scanf("%lf",&experimentalScore);

while(!(experimentalScore>=0&&experimentalScore<=100)){

printf("\t\t\t\t输入错误，请重新输入该学生的实验成绩(0-100):");

scanf("%lf",&experimentalScore);

}

printf("\t\t\t\t学号为%s学生的测试成绩(0-100):",num);

scanf("%lf",&testScore);

while(!(testScore>=0&&testScore<=100)){

printf("\t\t\t\t输入错误，请重新输入该学生的测试成绩(0-100):");

scanf("%lf",&testScore);

}

//将信息储存到结构体指针中

p->totalScore[i]=regularScore\*0.2+experimentalScore\*0.2+testScore\*0.6;

strcpy(p->className[i],className);

strcpy(p->grade,grade);

p->regularScore[i]=regularScore;

p->experimentalScore[i]=experimentalScore;

p->testScore[i]=testScore;

p->totalScoreAll+=p->totalScore[i];

p->next=NULL;

}

strcpy(p->num,num);

strcpy(p->name,name);

if(head==NULL)

{

head=p;

}

else

{

if(n==1)

{

p->next=q;

head=p;

}

else

{

for(int i=0;i<n-2&&q->next!=NULL;i++)

{

q=q->next;

}

if(q->next!=NULL)

{

p->next=q->next;

q->next=p;

}

else

{

q->next=p;

}

}

}

//输入相应的格式和相关学生的信息

printf("\n\n\t\t姓名\t\t学号\t\t班级\t\t课程名\t\t平时成绩\t\t实验成绩\t\t测试成绩\t\t总评成绩\t\t总成绩");

printf("\n --------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n");

for(p=head;fwrite(p,sizeof(struct stud\_node),1,fp),p!= NULL;p=p->next)

{

printf("\n\t\t%-10s\t%-10s\t%-10s\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->name,p->num,p->grade,p->className[0],p->regularScore[0],p->experimentalScore[0],p->testScore[0],p->totalScore[0],p->totalScoreAll);

for(int i=1;i<3;i++)

{

printf("\t\t\t\t\t\t\t\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->className[i],p->regularScore[i],p->experimentalScore[i],p->testScore[i],p->totalScore[i]);

}

}

fclose(fp);

printf("\n\n\n\t\t");

}

void deleteNode(struct stud\_node\* &head,int n)//删除的相关的操作

{

FILE \*fp;

if((fp=fopen("score.dat","wb"))==NULL)

{

printf("cannot open file\n");

return;

}

int score,i,flag=1;

char num[20],name[20];

struct stud\_node \*p,\*q;

p=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

q=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

p=q=head;

//查找相应的节点

if(n==1)

{

p=p->next;

head=p;

}

else

{

for(i=0;i<n-2&&q!=NULL;i++)

{

q=q->next;

}

if(i==n-2)

{

p=q->next;

q->next=p->next;

}

else

{

printf("\n\t\t\t\t您删除的学生信息不存在\n");

}

}

//输出相应的格式和相关学生的信息

printf("\n\n\t\t姓名\t\t学号\t\t班级\t\t课程名\t\t平时成绩\t\t实验成绩\t\t测试成绩\t\t总评成绩\t\t总成绩");

printf("\n --------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n");

for(p=head;fwrite(p,sizeof(struct stud\_node),1,fp),p!= NULL;p=p->next)

{

printf("\n\t\t%-10s\t%-10s\t%-10s\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->name,p->num,p->grade,p->className[0],p->regularScore[0],p->experimentalScore[0],p->testScore[0],p->totalScore[0],p->totalScoreAll);

for(int i=1;i<3;i++)

{

printf("\t\t\t\t\t\t\t\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->className[i],p->regularScore[i],p->experimentalScore[i],p->testScore[i],p->totalScore[i]);

}

}

printf("\n\n\n\t\t");

fclose(fp);

}

void gai(struct stud\_node\* head,int n)//修改学生信息的函数

{

FILE \*fp;

if((fp=fopen("score.dat","wb"))==NULL)

{

printf("cannot open file\n");

return;

}

int score;

char num[20],name[20],className[20],grade[20];

double regularScore,experimentalScore,testScore,totalScore;

struct stud\_node \*p;

p=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

p=head;

//查找相应的节点

for(int i=0;i<n-1&&p!=NULL;i++)

{

p=p->next;

}

//根据提示输入相关的信息

if(p!=NULL)

{

printf("\n\t\t\t\t请输入待插入学生的学号(1800000000-1899999999):");

scanf("%s",&num);

while(!(strcmp(num,"180000000")>=0&&strcmp(num,"1899999999")<=0))

{

printf("\t\t\t\t输入错误，请重新输入该学生的学号(1800000000-1899999999):");

scanf("%s",&num);

}

printf("\t\t\t\t请输入待插入学生的班级号(18000000-18999999):");

scanf("%s",grade);

while(!(strcmp(grade,"1800000")>=0&&strcmp(grade,"18999999")<=0))

{

printf("\t\t\t\t输入错误，请重新输入该学生的班级号(18000000-18999999):");

scanf("%s",grade);

}

printf("\t\t\t\t学号为%s学生的姓名:",num);

scanf("%s",name);

p->totalScoreAll=0;

for(int i=0;i<3;i++)

{

printf("\t\t\t\t学号为%s学生的课程名:",num);

scanf("%s",className);

printf("\t\t\t\t学号为%s学生的平时成绩(0-100):",num);

scanf("%lf",&regularScore);

while(!(regularScore>=0&&regularScore<=100)){

printf("\t\t\t\t输入错误，请重新输入该学生的平时成绩(0-100):");

scanf("%lf",&regularScore);

}

printf("\t\t\t\t学号为%s学生的实验成绩(0-100):",num);

scanf("%lf",&experimentalScore);

while(!(experimentalScore>=0&&experimentalScore<=100)){

printf("\t\t\t\t输入错误，请重新输入该学生的实验成绩(0-100):");

scanf("%lf",&experimentalScore);

}

printf("\t\t\t\t学号为%s学生的测试成绩(0-100):",num);

scanf("%lf",&testScore);

while(!(testScore>=0&&testScore<=100)){

printf("\t\t\t\t输入错误，请重新输入该学生的测试成绩(0-100):");

scanf("%lf",&testScore);

}

p->totalScore[i]=regularScore\*0.2+experimentalScore\*0.2+testScore\*0.6;

strcpy(p->grade,grade);

strcpy(p->className[i],className);

p->regularScore[i]=regularScore;

p->experimentalScore[i]=experimentalScore;

p->testScore[i]=testScore;

p->totalScoreAll+=p->totalScore[i];

}

strcpy(p->num,num);

strcpy(p->name,name);

//输出相应的格式和相关学生的信息

printf("\n\n\t\t姓名\t\t学号\t\t班级\t\t课程名\t\t平时成绩\t\t实验成绩\t\t测试成绩\t\t总评成绩\t\t总成绩");

printf("\n --------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n");

for(p=head;p!= NULL;p=p->next)

{

printf("\n\t\t%-10s\t%-10s\t%-10s\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->name,p->num,p->grade,p->className[0],p->regularScore[0],p->experimentalScore[0],p->testScore[0],p->totalScore[0],p->totalScoreAll);

for(int i=1;i<3;i++)

{

printf("\t\t\t\t\t\t\t\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->className[i],p->regularScore[i],p->experimentalScore[i],p->testScore[i],p->totalScore[i]);

}

}

printf("\n\n\n\t\t");

}

else

{

printf("\t\t\t\t您修改的学生信息不存在");

//输出相应的格式和相关学生的信息

printf("\n\n\t\t姓名\t\t学号\t\t班级\t\t课程名\t\t平时成绩\t\t实验成绩\t\t测试成绩\t\t总评成绩\t\t总成绩");

printf("\n --------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n");

for(p=head;fwrite(p,sizeof(struct stud\_node),1,fp),p!= NULL;p=p->next)

{

printf("\n\t\t%-10s\t%-10s\t%-10s\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->name,p->num,p->grade,p->className[0],p->regularScore[0],p->experimentalScore[0],p->testScore[0],p->totalScore[0],p->totalScoreAll);

for(int i=1;i<3;i++)

{

printf("\t\t\t\t\t\t\t\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->className[i],p->regularScore[i],p->experimentalScore[i],p->testScore[i],p->totalScore[i]);

}

}

printf("\n\n\n\t\t");

}

fclose(fp);

}

void testScoreSort(struct stud\_node\* &head)//根据总测试成绩排序

{

struct stud\_node \*q,\*p,\*m,\*t;

int j=1;

p=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

t=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

q=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

m=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

p=q=head;

//排序

for(;p!=NULL;p=p->next)

{

t=p;

for(q=p->next;q!=NULL;q=q->next)

{

if(t->testScore[0]+t->testScore[1]+t->testScore[2]+t->testScore[3]+t->testScore[4]<q->testScore[0]+q->testScore[1]+q->testScore[2]+q->testScore[3]+q->testScore[4])

{

t=q;

}

}

if(t!=p)

{

strcpy(m->name,t->name);

strcpy(t->name,p->name);

strcpy(p->name,m->name);

strcpy(m->num,t->num);

strcpy(t->num,p->num);

strcpy(p->num,m->num);

strcpy(m->grade,t->grade);

strcpy(t->grade,p->grade);

strcpy(p->grade,m->grade);

for(int i=0;i<3;i++)

{

m->regularScore[i]=t->regularScore[i];

t->regularScore[i]=p->regularScore[i];

p->regularScore[i]=m->regularScore[i];

m->experimentalScore[i]=t->experimentalScore[i];

t->experimentalScore[i]=p->experimentalScore[i];

p->experimentalScore[i]=m->experimentalScore[i];

m->testScore[i]=t->testScore[i];

t->testScore[i]=p->testScore[i];

p->testScore[i]=m->testScore[i];

m->totalScore[i]=t->totalScore[i];

t->totalScore[i]=p->totalScore[i];

p->totalScore[i]=m->totalScore[i];

strcpy(m->className[i],t->className[i]);

strcpy(t->className[i],p->className[i]);

strcpy(p->className[i],m->className[i]);

m->totalScoreAll=t->totalScoreAll;

t->totalScoreAll=p->totalScoreAll;

p->totalScoreAll=m->totalScoreAll;

}

}

}

//输出相应的格式和相关学生的信息

printf("\n\n\t\t姓名\t\t学号\t\t班级\t\t课程名\t\t平时成绩\t\t实验成绩\t\t测试成绩\t\t总评成绩\t\t总成绩");

printf("\n --------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n");

for(p=head;p!= NULL;p=p->next)

{

printf("\n\t\t%-10s\t%-10s\t%-10s\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->name,p->num,p->grade,p->className[0],p->regularScore[0],p->experimentalScore[0],p->testScore[0],p->totalScore[0],p->totalScoreAll);

for(int k=1;k<3;k++)

{

printf("\t\t\t\t\t\t\t\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->className[k],p->regularScore[k],p->experimentalScore[k],p->testScore[k],p->totalScore[k]);

}

j++;

}

printf("\n\n\n\t\t");

}

void totalScoreSort(struct stud\_node\* &head)//根据总成绩排序

{

int i=1;

struct stud\_node \*q,\*p,\*m,\*t;

p=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

t=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

q=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

m=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

p=q=head;

//排序

for(;p!=NULL;p=p->next)

{

t=p;

for(q=p->next;q!=NULL;q=q->next)

{

if(t->totalScoreAll<q->totalScoreAll)

{

t=q;

}

}

if(t!=p)

{

strcpy(m->name,t->name);

strcpy(t->name,p->name);

strcpy(p->name,m->name);

strcpy(m->num,t->num);

strcpy(t->num,p->num);

strcpy(p->num,m->num);

strcpy(m->grade,t->grade);

strcpy(t->grade,p->grade);

strcpy(p->grade,m->grade);

for(int j=0;j<3;j++)

{

m->regularScore[j]=t->regularScore[j];

t->regularScore[j]=p->regularScore[j];

p->regularScore[j]=m->regularScore[j];

m->experimentalScore[j]=t->experimentalScore[j];

t->experimentalScore[j]=p->experimentalScore[j];

p->experimentalScore[j]=m->experimentalScore[j];

m->testScore[j]=t->testScore[j];

t->testScore[j]=p->testScore[j];

p->testScore[j]=m->testScore[j];

strcpy(m->className[j],t->className[j]);

strcpy(t->className[j],p->className[j]);

strcpy(p->className[j],m->className[j]);

m->totalScore[j]=t->totalScore[j];

t->totalScore[j]=p->totalScore[j];

p->totalScore[j]=m->totalScore[j];

}

}

}

//输出相应的格式和相关学生的信息

printf("\n\n\t\t姓名\t\t学号\t\t班级\t\t课程名\t\t平时成绩\t\t实验成绩\t\t测试成绩\t\t总评成绩\t\t总成绩");

printf("\n --------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n");

for(p=head;p!= NULL;p=p->next)

{

printf("\n\t\t%-10s\t%-10s\t%-10s\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->name,p->num,p->grade,p->className[0],p->regularScore[0],p->experimentalScore[0],p->testScore[0],p->totalScore[0],p->totalScoreAll);

for(int k=1;k<3;k++)

{

printf("\t\t\t\t\t\t\t\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",p->className[k],p->regularScore[k],p->experimentalScore[k],p->testScore[k],p->totalScore[k]);

}

i++;

}

printf("\n\n\n\t\t");

}

void maxScore(struct stud\_node\* head,char n[])//输出相关课程最大值的函数

{

struct stud\_node \*q,\*p;

p=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

q=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

int t=-1;

p=head;

//循环遍历得到最大值

for(;p!=NULL;p=p->next)

{

for(int k=0;k<3;k++)

{

if(strcmp(p->className[k],n)==0)

{

if(t==-1)

{

strcpy(q->grade,p->grade);

strcpy(q->name,p->name);

strcpy(q->num,p->num);

for(int m=0;m<3;m++)

{

strcpy(q->className[m],p->className[m]);

q->regularScore[m]=p->regularScore[m];

q->experimentalScore[m]=p->experimentalScore[m];

q->testScore[m]=p->testScore[m];

q->totalScore[m]=p->totalScore[m];

}

t=1;

}

else

{

if(q->totalScore[k]<p->totalScore[k])

{

strcpy(q->name,p->name);

strcpy(q->num,p->num);

strcpy(q->grade,p->grade);

q->totalScoreAll=p->totalScoreAll;

for(int m=0;m<3;m++)

{

strcpy(q->className[m],p->className[m]);

q->regularScore[m]=p->regularScore[m];

q->experimentalScore[m]=p->experimentalScore[m];

q->testScore[m]=p->testScore[m];

q->totalScore[m]=p->totalScore[m];

}

}

}

}

}

}

//输出相关课程的最大值的信息

if(t==-1)

{

printf("\n\n\n\t\t最大值\t\t该课程不存在\n");

}

else

{

printf("\n\n\t\t姓名\t\t学号\t\t班级\t\t课程名\t\t平时成绩\t\t实验成绩\t\t测试成绩\t\t总评成绩\t\t总成绩");

printf("\n --------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n");

printf("\n最大值\t\t%-10s\t%-10s\t%-10s\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",q->name,q->num,q->grade,q->className[0],q->regularScore[0],q->experimentalScore[0],q->testScore[0],q->totalScore[0],q->totalScoreAll);

for(int k=1;k<3;k++)

{

printf("\t\t\t\t\t\t\t\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",q->className[k],q->regularScore[k],q->experimentalScore[k],q->testScore[k],q->totalScore[k]);

}

}

}

void minScore(struct stud\_node\* head,char n[])//输出相关课程的信息

{

struct stud\_node \*q,\*p;

p=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

q=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

int t=-1;

p=head;

//循环遍历得到最大值

for(;p!=NULL;p=p->next)

{

for(int k=0;k<3;k++)

{

if(strcmp(p->className[k],n)==0)

{

if(t==-1)

{

q->totalScoreAll=p->totalScoreAll;

strcpy(q->grade,p->grade);

strcpy(q->name,p->name);

strcpy(q->num,p->num);

for(int m=0;m<5;m++)

{

strcpy(q->className[m],p->className[m]);

q->regularScore[m]=p->regularScore[m];

q->experimentalScore[m]=p->experimentalScore[m];

q->testScore[m]=p->testScore[m];

q->totalScore[m]=p->totalScore[m];

}

t=1;

}

else

{

if(q->totalScore[k]>p->totalScore[k])

{

strcpy(q->name,p->name);

strcpy(q->num,p->num);

for(int m=0;m<5;m++)

{

strcpy(q->className[m],p->className[m]);

q->regularScore[m]=p->regularScore[m];

q->experimentalScore[m]=p->experimentalScore[m];

q->testScore[m]=p->testScore[m];

q->totalScore[m]=p->totalScore[m];

}

}

}

}

}

}

//输出相关课程的最大值的信息

if(t==-1)

{

printf("\t\t最大值\t\t该课程不存在\n\n\n\t\t");

}

else

{

printf("最小值\t\t%-10s\t%-10s\t%-10s\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",q->name,q->num,q->grade,q->className[0],q->regularScore[0],q->experimentalScore[0],q->testScore[0],q->totalScore[0],q->totalScoreAll);

for(int k=1;k<3;k++)

{

printf("\t\t\t\t\t\t\t\t%-10s\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\t\t%-10.1lf\n",q->className[k],q->regularScore[k],q->experimentalScore[k],q->testScore[k],q->totalScore[k]);

}

printf("\n\n\n\t\t");

}

}

void outStandingRate(struct stud\_node\* &head,char n[])//输出相关课程最大值的信息

{

int i=0,j=0;

double t;

struct stud\_node\* p;

p=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

p=head;

//循环遍历链表，得到相关课程优秀的学生个数

for(;p!=NULL;p=p->next)

{

for(int m=0;m<3;m++)

{

if(strcmp(p->className[m],n)==0)

{

if(p->totalScore[m]>=90)

{

i++;

}

j++;

}

}

}

//输出相关的结果

if(i==0&&j==0)

{

printf("\n\n\n\t\t\t\t该课程不存在");

}

else

{

t=100\*i/j\*1.0;

printf("\n\n\n\t\t\t\t%s的优秀率:%.2lf%\n",n,t);

}

}

void failRate(struct stud\_node\* &head,char n[])//输出相关课程的不及格率的相关信息

{

int i=0,j=0;

double t;

struct stud\_node\* p;

p=(struct stud\_node \*)malloc(sizeof(struct stud\_node));

p=head;

//循环遍历链表，累计不及格的学生个数

for(;p!=NULL;p=p->next)

{

for(int m=0;m<3;m++)

{

if(strcmp(p->className[m],n)==0)

{

if(p->totalScore[m]<=60)

{

i++;

}

j++;

}

}

}

//输出不及格率

if(i==0&&j==0)

{

printf("\n\t\t\t\t该课程不存在");

printf("\n\n\n\t\t");

}

else

{

t=100\*i/j\*1.0;

printf("\t\t\t\t%s的不及格率:%.2lf%\n",n,t);

printf("\n\n\n\t\t");

}

}

int main()

{

FILE \*fp;

if((fp=fopen("score.dat","rb"))==NULL)//打开文件

{

printf("cannot open file\n");

exit(0);

}

struct stud\_node \*head,\*tail,\*p;

int i=0,flag3=1,flag1,m;

char n[20];

int size = sizeof(struct stud\_node);

head=tail=NULL;

printf("\n\n Welcome your envoy\n");

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 学生信息管理系统 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n\n");

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n\n");

fread(&stud[i],sizeof(struct stud\_node),1,fp);

while(!feof(fp))//读取文件并打印

{

p=(struct stud\_node\*)malloc(size);

strcpy(p->num,stud[i].num);

strcpy(p->name,stud[i].name);

strcpy(p->className[0],stud[i].className[0]);

strcpy(p->className[1],stud[i].className[1]);

strcpy(p->className[2],stud[i].className[2]);

strcpy(p->grade,stud[i].grade);

p->regularScore[0]=stud[i].regularScore[0];

p->regularScore[1]=stud[i].regularScore[1];

p->regularScore[2]=stud[i].regularScore[2];

p->experimentalScore[0]=stud[i].experimentalScore[0];

p->experimentalScore[1]=stud[i].experimentalScore[1];

p->experimentalScore[2]=stud[i].experimentalScore[2];

p->testScore[0]=stud[i].testScore[0];

p->testScore[1]=stud[i].testScore[1];

p->testScore[2]=stud[i].testScore[2];

p->totalScore[0]=stud[i].totalScore[0];

p->totalScore[1]=stud[i].totalScore[1];

p->totalScore[2]=stud[i].totalScore[2];

p->totalScoreAll=stud[i].totalScoreAll;

p->next=NULL;

i++;

if(i==1)

head=tail=p;

else

{

tail->next=p;

tail=p;

}

fread(&stud[i],sizeof(struct stud\_node),1,fp);

}

//输处菜单栏

printf("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t菜单栏\n");

printf("\n\t\t\t\t[0].显示全部\t\t\t[1].查询(通过学号)\t\t\t[2].查询(通过姓名)\t\t\t[3].查询(通过课程名称)\n");

printf("\n\t\t\t\t[4].插入\t\t\t[5].删除\t\t\t [6].修改学生信息\t\t\t[7].排序(通过测试成绩)\n");

printf("\n\t\t\t\t[8].排序(通过总评成绩)\t\t[9].课程最大值/最小值\t\t\t[10].课程优秀率/不及格率\t\t输入其他停止");

printf("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t菜单栏操作数:");

scanf("%d",&flag1);

while(flag3)

{

switch(flag1)//选择相应的提示和输入的相应操作

{

case 1:

printf("\t\t请输入查找学生的学号（1800300000-1899999999）:");

scanf("%s",&n);

break;

case 2:

printf("\t\t请输入查找学生的姓名:");

scanf("%s",&n);

break;

case 3:

printf("\t\t请输入查找的课程名:");

scanf("%s",&n);

break;

case 4:

printf("\t\t请输入插入学生的位置:");

scanf("%d",&m);

break;

case 5:

printf("\t\t请输入删除学生的位置:");

scanf("%d",&m);

break;

case 6:

printf("\t\t请输入修改学生的位置:");

scanf("%d",&m);

break;

case 9:

case 10:

printf("\t\t请输入课程的名称:");

scanf("%s",&n);

break;

default:

break;

}

switch(flag1)//选择执行的函数

{

case 0:displayAll(head);break;

case 1:search1(head,n);break;

case 2:search2(head,n);break;

case 3:search3(head,n);break;

case 4:insert(head,m);break;

case 5:deleteNode(head,m);break;

case 6:gai(head,m);break;

case 7:testScoreSort(head);break;

case 8:totalScoreSort(head);break;

case 9:maxScore(head,n);minScore(head,n);break;

case 10:outStandingRate(head,n);failRate(head,n);break;

default:flag3=0;

}

if(flag3!=0)//输出菜单栏

{

system("pause");

system("cls");

printf("\n\n Welcome your envoy\n");

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 学生信息管理系统 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n\n");

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n\n");

printf("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t菜单栏\n");

printf("\n\t\t\t\t[0].显示全部\t\t\t[1].查询(通过学号)\t\t\t[2].查询(通过姓名)\t\t\t[3].查询(通过课程名称)\n");

printf("\n\t\t\t\t[4].插入\t\t\t[5].删除\t\t\t [6].修改学生信息\t\t\t[7].排序(通过测试成绩)\n");

printf("\n\t\t\t\t[8].排序(通过总评成绩)\t\t[9].课程最大值/最小值\t\t\t[10].课程优秀率/不及格率\t\t输入其他停止");

printf("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t菜单栏操作数:");

scanf("%d",&flag1);//输入所选的菜单选项

}

}

fclose(fp);

return 0;

}

**三、实验报告内容及要求**

1．按实验报告要求（样板）撰写并打印实验报告；

2．分析并画出主要程序流程图：（1）主控模块、（2）学生信息输入模块、（3）学生信息查询模块；

3. 给出源代码及输入输出结果截图；

4．实验中遇到的问题和解决问题的方法。