想要使用后端数据，请添加"class\_student.h"和"functions.h"头文件  
后端用来交付的数据会存在data命名空间当中，使用时请using namespace data，或使用data::  
后端使用的类名称为student  
下面介绍后端函数接口的使用方法

**init**

无参数，直接调用即可  
函数作用是初始化数据，从link\_list.txt文件中读取数据， 并建立链表，需在main函数的开头使用

**save**

无参数，直接调用即可  
作用是将目前程序的链表存入link\_list.txt文件中，前端界面中应当设有相应按钮，让用户可以保存数据。 若是在退出时能够提醒用户保存未保存的数据，则更好  
前端工程师也可以选择适当的时候，自动存储数据

**~~seek~~**

1参数，const char\* id\_or\_name 此函数用来查找。接收一个参数id\_or\_name。将需要查找的对象的学号或者姓名输入，函数就会自动查找所有符合条件的对象，并将这些对象的指针，按照学号顺序存储在data命名空间的seek\_res()数组中(**下标从0开始，以下不特别说明下标均从0开始**)(注意，存储的是**student对象类型的指针**)，  
符合条件的结果数量会存储在data::seek\_num整型变量中

**~~add~~**

5参数，const char\* id, const char\* name, int sex, int school, int class\_num  
1返回值 student\* 此函数用来添加学生，5个参数分别为学生的学号，姓名，性别，学院，班级号  
性别，学院已经设置好了枚举型变量，在参数表中直接填写即可，不需要加双引号  
性别：MALE, FEMALE  
学院：JI\_RUAN, DIAN\_XIN, SHU\_TONG  
举例：  
student\* ptr = add("201983290999", "马保国", MALE, JI\_RUAN, 1);  
或  
student\* ptr{add("201983290999", "马保国", MALE, JI\_RUAN, 1)};  
函数设有1个返回值，会返回指向这个新建对象的指针，注意，add函数里面不包含对学科分数的登记，登记分数可以通过这个返回的指针，用下面即将提到的modify\_score方法修改，切记要登记分数！

**modify\_score方法**

2参数，int sub, int score  
注意，**这是student类的方法**，此方法可以用来修改学生的分数，设有2个参数，分别是需要修改的科目和分数  
科目也是枚举类型：  
科目：GAO\_SHU, XIAN\_DAI, CHENG\_XU, GAI\_TONG, DIAN\_ZI, JIN\_SHI, CHANG\_WEI\_FEN, XIN\_HAO, OS, NET  
写入参数的时候也是直接写汉字就可以了，不用加双引号  
举例：  
student\* ptr{add("201983290999", "马保国", MALE, JI\_RUAN, 1)};//ptr指向你新加入的学生  
ptr->modify\_score(GAO\_SHU, 90);//这样即可登记高等数学的分数为90

**modify\_school方法**

1参数，int stu\_school  
注意，这是student类的方法，此方法可以改变学生的学院，参数填枚举类型即可

**modify\_class\_num方法**

1参数，int class\_num  
注意，这是student类的方法，此方法可以改变学生的班级，参数填班级号即可

**其他获取信息的方法**

注意，以下都是student类的方法，可以通过对象指针来调用  
get\_id 无参数，返回学号  
get\_name 无参数，返回姓名  
get\_sex 无参数，返回性别，0为男，1为女  
get\_school 无参数，返回学院  
get\_class\_num 无参数，返回班级  
get\_score 1参数，输入学科的枚举类型，返回对应学科分数  
get\_total\_score 无参数，返回总分

**~~del~~**

1参数，student\* pdel  
函数可以删除学生信息，需要向参数列表中提供该对象的指针，即可删除该学生信息

**del\_all**

无参数 直接调用即可。此函数可以删除当前程序链表中的所有数据项，但在未保存之前不会影响txt文件，前端应当提供这个函数的按钮，但是应当谨慎使用这个函数，可以提醒用户确认之后再调用

messagebox Y/N

**~~sort\_by\_total\_score~~**

2个重载 1参数，int school  
2参数，int school, int class\_num  
该函数实现学生按总分排序，两个重载分别实现了学院排序和班级排序，参数分别需要学院的枚举类型，学院的枚举类型和班级序号  
函数会将排序好的结果存储在data::seek\_res数组中(注意，**存储的是排好序的结果的指针**)，排序的总数量存储在data::seek\_num中

**~~sort\_by\_subject\_score~~**

2个重载  
2参数，int school, int subject  
3参数，int school, int class\_num, int subject  
函数实现单门科目的排序，可以按学院或者按班级，用法和sort\_by\_total\_score类似，只不过参数多了一个学科枚举类型

**~~sort\_by\_id~~**

3个重载 0参数  
1参数 int school  
2参数 int school, int class\_num  
函数实现按学号排序，无参数时按学校，1参数按学院，2参数按班级。用法依然和sort\_by\_total\_score类似

**~~subject\_pass\_rate~~**

3个重载  
1参数 int subject  
2参数 int school, int subject  
3参数 int school, int class\_num, int subject  
1返回值 double  
该函数计算通过率，1个参数仅需提供学科枚举类型，计算全校通过率，2参数需提供学院枚举类型和学科，计算学院通过率，3参数需提供学院，班级和学科，计算班级通过率  
返回一个double类型变量，为通过率

**~~subject\_average~~**

3个重载  
1参数 int subject  
2参数 int school, int subject  
3参数 int school, int class\_num, int subject  
1返回值 double  
函数计算科目平均分，用法和subject\_pass\_rate类似，返回一个double类型，是平均分

**Tigs for Test Engineer**

项目有不少的测试任务...

**Back-end Test**

首先需要你们测试的是后端的正确性，需要编写用来测试后端的大批量数据  
后端的输入不负责健壮性检查，所以请确保你们的输入格式符合后端的规范;)  
测试数据应当具有代表性和差异性，能够尽量检测出程序不完善的地方  
后端测试数据的需求量比较庞大，很难手写，所以建议你们编程来生成数据(不限语言)  
生成的测试数据需要保存在**txt文件**中  
需要生成的数据是学生信息，具体的格式为：  
分为N行，N是学生的数量，N的大小尽量大一些，但是不要超过**50000**  
每一行中需要有该学生的信息，按顺序分别为：学号，姓名，性别，学院，班级，20个科目成绩，**每一项用空格隔开**  
学号需要保证每个学生都不同，位数不超过14  
姓名使用汉字，不超过9个字，学生之间应该**有适当的重名**  
性别为MALE, FEMALE  
学院可以是：JI\_RUAN, DIAN\_XIN, SHU\_TONG  
班级应该是一个整型数字，不要太大，应保证一个班级中有适量的学生  
20个科目成绩可以是-1, -2或0到100，-2代表未修，-1代表在修，0到100代表已修，记录成绩  
应当保证一个学院的学生修的学科基本相同，不同学院学院学生修的学科有差异  
在满足上述条件的基础上，你们应当思索程序可能会对什么样的情况欠考虑，尽量编写能够发现程序漏洞的数据  
后端的测试需求基本就是这样 (//∇//)\  
示例数据：  
201983290999 王保果 MALE JI\_RUAN 10 90 90 90 -1 90 70 85 -1 55 61 -2 -2 -2 -2 -2 -2 -2 -2 -2

**Front-end Test**

测试前端程序的时候你们面对的是有图形化界面的程序  
因为程序采用了图形化编程，所以其实可供测试健壮性的点并不多，但是程序中依然会有输入框  
这时间，可供你们发挥的空间就来了ヽ(•̀ω•́ )ゝ  
告诉程序，什么叫不讲码德  
采取你们能想到的非法输入方法，测试程序的健壮性，并把结果反馈给前端工程师  
可以参考如下做法：

一个测试工程师走进一家酒吧，要了一杯啤酒；  
一个测试工程师走进一家酒吧，要了一杯咖啡；  
一个测试工程师走进一家酒吧，要了0.7杯啤酒；  
一个测试工程师走进一家酒吧，要了-1杯啤酒；  
一个测试工程师走进一家酒吧，要了2^32杯啤酒；  
一个测试工程师走进一家酒吧，要了一杯洗脚水；  
一个测试工程师走进一家酒吧，要了一杯蜥蜴；  
一个测试工程师走进一家酒吧，要了一份asdfQwer@24dg!&\*(@；  
一个测试工程师走进一家酒吧，什么也没要；  
一个测试工程师走进一家酒吧，又走出去又从窗户进来又从后门出去从下水道钻进来；  
一个测试工程师走进一家酒吧，又走出去又进来又出去又进来又出去，最后在外面把老板打了一顿；  
一个测试工程师走进一家酒吧，要了一杯烫烫烫的锟斤拷；  
一个测试工程师走进一家酒吧，要了NaN杯Null；  
一个测试工程师冲进一家酒吧，要了500T啤酒咖啡洗脚水野猫狼牙棒奶茶；  
一个测试工程师把酒吧拆了；  
一个测试工程师化装成老板走进一家酒吧，要了500杯啤酒并且不付钱；  
一万个测试工程师在酒吧门外呼啸而过；  
一个测试工程师走进一家酒吧，要了一杯啤酒';DROP TABLE 酒吧；  
测试工程师们满意地离开了酒吧。

但是仍然注意：你们最先测试的应当是正常的数据，在正常的前提下逐步提高数据非法程度  
记得反馈问题给工程师ヽ(￣д￣;)ノ=3=3=3  
嗯，关于测试的事情就这么多