هپولی مینی‌پروژش رو نزده!

در تصویر بالا چهره ناراحت هپولی رو مشاهده می کنید. هپولی قصه ما چون عید را صرف مرور سریال ها ... اهم اهم... درس هایش کرد نرسید که مینی‌پروژه درس برنامه نویسی پیشرفته‌اش را بزند. پس پیش تپلی یکی از تی ای های درس رفت و از آن خواست تا بتواند جبران کند. تپلی پیشنهاد ایجاد یک برنامه برای **محاسبه معدل** دانشجویان کلاس را به هپولی داد.

قصد داریم لیستی از دانشجویان داشته باشیم. هر دانشجو دارای اطلاعات زیر است :

- نام

- نام خانوادگی
- شماره دانشجویی
- نمره

همچنین متد های زیر را هم داراست:

۱. توابع Set و Get برای هر دانشجو
۲. تابع سازنده با ورودی های نام نام خانوادگی و نمره

نکته ۱: تابع سازنده به هر دانشجو یک شماره دانشجویی یکتا اختصاص می دهد با فرمت زیر: "3"+یک عدد+"99"

نکته ۲: بطور پیش فرض نمره هر دانشجو صفر است

۳. تابع مخرب
۴. یک تابع که تعداد دانشجویان ثبت شده تا اکنون را برگرداند.
۵. یک تابع که معدل دانشجویان ثبت شده تا کنون را برگرداند.

نکته: دوتابع ۴ و ۵ باید بتوان بدون ساخت شیئ از کلاس صدا زد.

موارد مهم:

- به فایل بندی کلاس هایتان توجه نمایید.
- در توابع سازنده و مخرب می توانید متغیر های مربوط به تعداد دانشجویان و... را مدیریت کنید.
- علت تعریف متغیر های اضافه خود را با کامنت در کد خود مشخص کنید.

دوراهی چندگانه

خب بیاین دوراهیمونو upgrade کنیم! حالا چجوری؟؟

میخایم درخت ما محدود به عدد صحیح نباشه، یعنی یه بار درخت اعداد اعشاری رو بسازیم یه بار کاراکتر و ... .

برای این کار از کلاس تمپلیت ها استفاده می کنیم! کلاس های تمپلیت به ما قابلیت ساخت کلاس ها با نوع داده های متفاوت(بر اساس پارامتری که به اون ها پاس می دیم) رو میدن. به صورت کلی کلاس های تمپلیت برای پیاده سازی کانتینر ها (Vector , ...) استفاده میشن. آبجکت های این کلاس ها هم با پاس دادن پارامتر مربوطه ساخته میشن. مثال زیر یک کلاس تمپلیته که وظیفش نگه داشتن یک عضو از هر نوع داده ست و برایش یک تابع divideBy2 تعریف شده که المنت مورد نظر رو بر دو تقسیم می کنه.

```
template <class T>
class MyTemplate {
T element;
public:
MyTemplate (T arg) {element=arg;}
T divideBy2 () {return element/2;}
};
```

همچنین می تونیم در این کلاس ها برای یک نوع داده ی خاص کلاس را شخصی سازی کنیم، که به این کار **Template Specialization** میگن. برای مثال بالا شخصی سازی یک تمپلیت برای نوع داده ی کاراکتر بهتره، پس ما به جای تابع divideBy2 یک تابع printElement برای نوع داده ی char تعریف می کنیم:

```
template <>
class MyTemplate <char> {
char element;
public:
MyTemplate (char arg) {element=arg;}
char printElement ()
{
return element;
```

```
}
};
```

حالا ازتون میخایم که سوال دوراهی جلسه قبل رو کمی تغییر بدین و با تمپلیت بزنین و بعد اجراش کنین.

احتمالا هنگام اجراش به ارور لینک برمی خورین. **علت این مشکل**، **توضیح در مورد این ارور** و **راه حل برای برطرف سازیش** رو توی یک **PDF** بنویسین و به کد upgrade شده ضمیمه کنین و بفرستین.

هپولی خسته شده :(

اول این ترم هپولی خوشحال و خندان ما از رفیق های بابش مشورت گرفت درمورد انتخاب واحد!

این دوستان خیلی باب به هپولی گفتن برو سریع درس جبرخطیو بگیر که خیلی به درد کامپیوتر میخوره و خوبه برات و هپولیم که حرف گوش کن تازه تایم انتخاب واحدشم زود بود رفت درسو گرفت!

الان از بس داره با ماتریس ها و... کار میکنه آرزوش شده که کاش خود سی پلاس پلاس یه داده ساختار ماتریسم داشت مثل وکتور و... اما خودش بلد نیست یک ماتریسی درست کنه که همه کاری بکنه تازه انواع داده های مختلف رو داشته باشه پس از شما میخواد که اینکارو براش بکنین. :)

کلاس ماتریس از موارد زیر تشکیل شده است:

۱. یک آرایه دو بعدی تمپلیت از نوع vector. (برای اطلاع بیشتر به این لینک مراجعه کنید.)
۲. تابع سازنده که یک وکتور دو بعدی به عنوان ورودی می‌گیرد.
۳. تابع سازنده پیشفرض (بدون ورودی)
۴. تابع مخرب
۵. توابع Getter که مقدار یک درایه یا کل ماتریس یا یک سطر را برمی‌گرداند.
۶. تابع Setter که دیتای متناظر با سطر و ستون یک ماتریس را در صورت وجود تغییر می‌دهد و یا شماره سطر و یک وکتور را بگیرد و مقدار آن سطر را تغییر دهد.
۷. تابع جمع و تفریق و ضرب که یک ماتریس ورودی می‌گیرد.
۸. تابع ضرب و تقسیم که یک عدد ورودی می‌گیرد و ماتریس را با آن عدد ضرب یا تقسیم می‌کند.
۹. تابع `is_equal` که یک ماتریس به عنوان ورودی می‌گیرد و برابری دوماتریس را چک می‌کند.
۱۰. تابع `is_not_equal` که برابر نبودن دوماتریس را چک می‌کند.
۱۱. تابع `matrix_cin` که مقدار دهی ماتریس را بر عهده دارد. (چگونگی اتخاذ پارامتر ها بر عهده خودتان است.)
۱۲. تابع `matrix_cout` که مارتیس را در صفحه کنسول پرینت میکند.

همچنین برای ماتریس های از نوع *char به موارد زیر توجه کنید:

- این کلاس همانند کلاس تمپلیت توابع بالا را دارد بجز موارد ۷ و ۸.
- علاوه بر توابع اصلی یک تابع جمع داریم که یک ماتریس از جنس char* به عنوان ورودی دریافت می‌کند و عملیات concat را انجام می‌دهد. برای مثال جمع درایه "a" با "b" برابر "ab" است.

نکته ۱: در توابعی که پیاده سازی میکنید به شرط هایی که ماهیت ماتریس بودن را تغییر میدهند توجه کنید و با استفاده از کتابخانه `assert.h` ورودی را فیلتر کنید.

نکته ۲: در این کلاس نیازی به پیاده سازی در چند فایل نیست.