دوراهی چندگانه (Template class)

پیش‌نیاز مسئله: دوراهی (مباحث اولیه شیئ گرایی)

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.

زمان شئ است ۲ (operator overloading)

پیش‌نیاز مسئله: زمان شیئ است ۱ (مفاهیم اولیه شیئ گرایی)

شی گرایی همچنان مسئله مهمی برای سپهر هست و جدیدا با سربارگذاری عملگر ها آشنا شده است برای همین تصمیم گرفته که به مسئله های شی گرایی قبلی سر بزند و تابع های هرکدام را که توانست با استفاده از عملگر ها پیاده سازی کند و حال به سراغ کلاس time رفته است.

در این سوال ما می خواهیم برخی از توابع کلاس time که در سوال **زمان شی است** نوشتیم را با سربار گذاری عملگر ها درست کنیم.

- می خواهیم با استفاده از عمگر >> یک شی از کلاس time را از ورودی بگیرد(برای اینکار می توانید یا فرمت مشخص خاصی مشخص کنید یا پیام مناسب دهید.)

```
friend istream & operator >> (istream &in,  Time &t){
       in >> SOMETHING;
         return in;
}
```

- می خواهیم با استفاده از عملگر << شی از کلاس time را چاپ کند و همچنین بنویسد که این ساعت در موقعی از شبانه روز است طبق تابعی که قبلا پیاده سازی کردید.

```
friend ostream & operator << (ostream &out,  Time &t){
   out << SOMETHING;
   return out;
}
```

- با استفاده از عملگر های < یا > یا =< یا => بتوان ۲ شی از این کلاس را مقایسه کرد.

```
friend bool operator<(const Time& t1, const Time& t2){
   bool b;
```

```
    //do comparison
    return b;
}
```

- میخواهیم عملیات cast کردن را با استفاده از overloading پیاده سازی کنیم، به شکلی که بتوانیم کلاس زمان را به رشته تبدیل کنیم.
- همچنین عمگر + برای جمع دو زمان که یک شیئ زمان را بعنوان ورودی می‌گیرد.

هپولی خسته تر از هفته پیش (operator overloading)

پیش‌نیاز مسئله: هپولی خسته شده (مباحث اولیه شیئ گرایی)

> بیا روراست باشیم. من خستمممممم!!

یک روز که هپولی داشت خستگی در می کرد به این فکر افتاد که اگر هر نفر بخواد اسم یک سری توابع برای ورودی گرفتن ، خروجی گرفتن ، جمع و ..... بخواهد تابع بنویسد و اسمی که دوست دارد را برایش بگذارد اینطوری اگر ما بخواهیم از اون کلاس استفاده کنیم، همش باید کد اون شخص نگاه کنیم و خب خیلی خسته کننده است برای همین رفت جستجو کرد و با سربارگذاری عملگر ها(operator overloading) آشنا شد و تصمیم گرفت که کد ماتریسی که هفته پیش زد با استفاده از این روش بهینه تر بکند.

لیست توابعی که باید عوض شوند:

- به جای تابع `matrix_cin` از عملگر `>>` استفاده کنید
- به جای تابع `matrix_cout` از عملگر `<<` استفاده کنید
- به جای تابع ضرب و تفریق و جمع یک ماتریس با ماتریس دیگر را از `*` , `-` , `+` استفاده کنید
- همچنین عملگر های `*=` ، `+=` و `-=` برای اینکه نتیجه محاسبات در خود ماتریس ذخیره کند
- به جای تابع ضرب و تقسیم یک عدد در یک ماتریس از علامت `*` , `/` استفاده کنید
- به جای تابع `is_equal` از عملگر `==` استفاده کنید
- به جای تابع `is_not_equal` عملگر `=!` استفاده کنید
- استفاده کردن از عملگر () به صورت (ستون، ردیف) برای نشان دادن داده موجود در آن خانه
- همچنین از عملگر `++` و `--` استفاده کنید، به این صورت که همه مقادیر ماتریس را به علاوه ۱ یا منهای ۱ بکند (عملگر ها در هر ۲ حالت **پسوند و پیشوند** تعریف شده باشند)

موارد دیگر:

- به جای استفاده کردن از عملگر (x,y) عملگر [y][x] را پیاده سازی کنید و روش های مختلف را در قالب یک فایل PDF ارسال کنید.

کلاس چند جمله‌ای (operator overloading)

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

هپولی تصمیم گرفته که برای اینکه حسابی operator overloading بلد بشه یک کلاس چندجمله ای درست کنه ولی خب دوست داره که بهش کمک کنید برای همین مشخصات کلاسی که می خواد درست کنه برای شما می نویسد تا به هپولی کمک کنید.

کلاس چندجمله ای در این کلاس ضرایب یک چندجمله ای را در یک وکتور نگه می داریم:

- ضرایب همگی صحیح است
- درجه همه ضرایب حداکثر ۲۰ می باشد متد های کلاس چندجمله ای:
- سازنده پیش فرضی که تمام ضرایب را در ابتدا صفر قرار می دهد
- سازنده کپی که چندجمله ای را در یک چند جمله ای دیگر کپی می کند
- عملگر تک عملوندی "-" که یک چند جمله ای را قرینه می کند
- عملگر های جمع و تفریق و ضرب برای ۲ چند جمله ای را تعریف کنید
- عملگر های == ، =< ، => ، < ، > را برای ۲ چندجمله ای تعریف کنید
- عملگر >> برای ورودی گرفتن یک چند جمله ای به صورت فرم استاندارد و نسبت داد آن به یک شی چندجمله ای
- عملگر << برای چاپ کردن چندجمله ای به صورت فرم استاندارد.

فرم استاندارد به صورت زیر می باشد:

```
...
6x^7-2x^2+x-3
19x^17+34x-78
```

نکات:

- در هنگام چاپ باید بر اساس درجه به صورت نزولی چاپ شود
- در هنگام ورودی نیازی به رعایت ترتیب نیست
- توان صفر به صورت عدد ثابت نشان داده می شود
- ضریب صفر نشان داده نمی شود
- توان یک در x^1 نشان داده نمی شود
- اگر ضریب ۱+ یا ۱- باشد، ضریب را نشان نمی دهد جز در شرایطی که توان صفر باشد

موارد امتیازی:

- استفاده از regex برای گرفتن ورودی. (برای آشنایی بیشتر این لینک و این لینک را مشاهده کنید.)
- پیاده سازی عملگر های / برای تقسیم و ٪ برای باقی مانده ۲ چندجمله ای

unique pairs (Class termplate)

صورت سوال، سوال اول پی دی اف تمرین است.

کلاس گراف جهت‌دار (Class termplate, operator overloading)

صورت سوال، سوال دو پی دی اف تمرین است.

کلاس عدد بزرگ (operator overloading)

در این مسئله میخواهیم یک کلاس تعریف کنیم که می‌تواند اعداد صحیح خیلی بزرگ (اعدادی که طول آن ها بیش از حداکثر تعریف شده در cpp باشد.) را در خود نگهداری کند.

برای ذخیره سازی این نوع داده می‌توانید ارقام را در یک بردار نگهداری کنید. (هر رقم در یک خانه ی آن بردار(همان وکتور)) برای این کلاس دو سازنده تعریف کنید یکی یک عدد صحیح می‌گیرد و دیگری رشته ای که ارقام در آن ذخیره شده اند.

این کلاس باید دارای متد های زیر باشد:

- یک سازنده با ورودی از نوع عدد صحیح
- یک سازنده با ورودی از نوع رشته

عملگرهای زیر را برای این کلاس تعریف کنید:

- (امتیازی) عملگر های + و =+ را برای این کلاس تعریف کنید.
- عملگر های == ، < را برای مقایسه دو کلاس تعریف کنید.
- (امتیازی) > و همچنین کوچکترمساوی را برای مقایسه دو کلاس تعریف کنید.
- (امتیازی) عملگر << برای ورودی گرفتن.
- عملگر >> برای چاپ کردن این عدد.

موارد مهم:

- توجه داشته باشید کلاس شما باید بتواند با ابجکت های از نوع const هم کار کند.
- به مثبت یا منفی بودن عدد ورودی توجه شود.
- هر متد دیگری که این کلاس می‌تواند نیاز داشته باشد را اضافه کنید. (برای مثلا ستر و گتر، سازنده کپی و...)
- توجه داشته باشید رعایت اصول شئ‌گرایی و encapsulation الزامی است.

- طراحی کلاس های شما باید در چند فایل در صورت نیاز باشد.