تمرین سوم

مهلت تمرین 13 خرداد

در این تمرین قرار است کلاس چند جمله ای پیاده کنید که بیشتر عملیات ریاضی را پشتیبانی میکند.

کلاس Term

این کلاس نشان دهنده یک جمله از چند جمله ای است که به صورت cx^p نشان داده میشود. c در یک جمله ضریب ($coefficient$) متغیر مجهول x و p درجه جمله یا توان ($power$) متغیر x است.

ضریب متغیر از جنس $float$ و p از جنس عدد صحیح است. برای این کلاس $operator$ های زیر را $overload$ کنید.

- `Operator +`
- `Operator -`
- `Operator *`
- `Operator /`
- `Operator +=`
- `Operator -=`
- `Operator *=`
- `Operator /=`
- `Operator =`
- `Operator >, <, <=, >=, ==,!=`
- `Operator ~`
- `Operator ++ (Post / Pre-increment)`
- `Operator -- (Post / Pre-decrement)`
- `Operator ()`
- `Operator << (overload ostream)`
- `Operator >> (overload istream)`

اپراتور ~ مشتق جمله را حساب میکند و بر روی جمله اصلی اعمال **نمیشود**.

توجه کنید که اپراتور ها باید اعمال متناسب با خود را انجام دهند و خروجی استاندارد داشته باشند. برای مثال اپراتور + نباید خروجی رفرنسی به آبجکت برگرداند و یا عملیات را بر روی خود آبجت اعمال کند.

توجه کنید که باید بتوان اپراتور های $+, *, -, /, <, >, ==, !=, \leq, \geq$ را برای اعداد اعشاری هم فراخوانی کرد. برای مثال:

Float + Term
Term + Float
Term + Term

اپراتور () ورودی عددی از جنس $float$ میگیرد و مقدار جمله را به ازای x برابر آن عدد خروجی میدهد. برای مثال:

$term = 3x\text{^}2$

$term(4) = 48$

(خروجی این تابع $float$ است).

اپراتور اورلود شده برای $ostream$ باید بتواند جمله را در آبجکتی از $ostream$ استریم کند. به مثال های زیر دقت کنید:

"x", "$2.3x$", "$x\text{^}3$", "3", "$x\text{^}6$"

اپراتور اورلود شده برای $istream$ یک رشته از آبجکت استریم ورودی میگیرد و آبجکت $Term$ را تغییر میدهد. ورودی ها به صورت زیر هستند:

"x", "$1.1x$", "$x\text{^}3$", "3", "$-x\text{^}6$"

اپراتورهای ***Pre/post-increment*** و ***Pre/post-decrement*** جمله را با عدد 1 به ترتیب جمع و تفریق میکند.

در صورتی که برای جمع یا تفریق دو جمله توان ها برابر نبود باید خطایی پرتاب شود.

برای تمامی $attribute$ های این کلاس گتر و ستر تعریف کنید

کلاس *Polynomial*

این کلاس نشان دهنده یک چند جمله ای است که به صورت $c_1x^{p_1} + c_2x^{p_2} + \ldots + c_nx^{p_n}$ است. یک چند جمله ای از جمع چندین جمله بدست می آید.

این کلاس دارای یک $vector$ از جنس $Term$ است.
برای این کلاس سازنده ای تعریف کنید که یک وکتور از جنس $Term$ به عنوان ورودی میگیرد (میتوانید از $Initializer\_list$ هم استفاده کنید).

برای این کلاس $Operator$ های زیر را $overload$ کنید:

- Operator +
- Operator -
- Operator *
- Operator +=
- Operator -=

- Operator *=
- Operator =
- Operator >, <, <=, >=, ==, !=
- Operator ~
- Operator ++ (Post / Pre-increment)
- Operator -- (Post / Pre-decrement)
- Operator ()
- Operator [] (index operator for vector)
- Operator << (overload ostream)
- Operator >> (overload istream)

اپراتور ~ مشتق چند جمله را حساب میکند و بر روی چند جمله اصلی اعمال **نمیشود**.

توجه کنید که اپراتور ها باید اعمال متناسب با خود را انجام دهند و خروجی استاندارد داشته باشند. برای مثال اپراتور + نباید خروجی رفرنسی به آبجکت برگرداند و یا عملیات را بر روی خود آبجت اعمال کند.

توجه کنید که باید بتوان اپراتور های $+,\ *, -, <, >, ==, !=, \leq, \geq$ را برای اعداد اعشاری یا $Term$ هم فراخوانی کرد.
برای مثال:

```
Polynomial + Term
Term + Polynomial
Polynomial + Polynomial
Polynomial + Float
Float + Polynomial
```

آیا میتوانید سازنده مناسبی برای کلاس $Polynomial$ و $Term$ بنویسید که **با یک بار اورلود کردن هر یکی** از اپراتورهای بالا بتواند برای تمام حالت های گفته شده خروجی درست بدهد؟ آیا میدانید چرا؟ (پاسخ این سوال اختیاری است و نمره اضافه دارد)

برای مثال تابع زیر بتواند تمامی حالت های گفته شده در بالا را پوشش دهد.

```
Polynomial operator-(const Polynomial& a, const Polynomial& b) . . .
```

وکتور جمله ها باید همواره به صورت نزولی $sort$ سورت شده باشد.

توجه کنید که نباید دو جمله در یک چند جمله ای وجود داشته باشد که توان های مساوی داشته باشند. اگر دو جمله با توان های مساوی وجود داشت باید آن ها را با هم جمع کرد و به یک جمله تبدیل کرد.

اپراتور () ورودی عددی از جنس $float$ میگیرد و مقدار چند جمله ای را به ازای x برابر آن عدد خروجی میدهد.

اپراتوری که برای $ostream$ اورلود میشود باید بتواند چند جمله ای را استریم کند. برای مثال

$x\text{^}4 - 2x\text{^}3 + 2x\text{^}2 - x + 1$

اپراتوری که برای $istream$ اورلود میشود باید بتواند یک رشته از جمع یا تفریق چندین جمله را بگیرد و به آبجکت $Polynomial$ بدهد.

اپراتور [] با گرفتن یک عدد صحیح به عنوان $Index$، جمله ای که در ایندکس مشخص شده درون وکتور موجود است را ریترن میکند. این اپراتور را یکبار $const$ اورلود کنید که کپی از آن جمله برمیگرداند و یکبار غیر $const$ تعریف کنید که رفرنسی به آن جمله برمیگرداند.

تابعی تعریف کنید که تعداد جمله های چند جمله ای را برگرداند.
تابعی تعریف کنید که درجه چند جمله ای را برگرداند.

فایل اصلی

در تابع $main$ منویی قرار دهید که شامل گزینه های زیر باشد.

Main Menu

```
1- New Polynomial
2- Load from text file
3- Load from binary file
4- Quit
```

با انتخاب گزینه اول منوی جدیدی نمایش داده شود که شامل گزینه های زیر باشد:

Polynomial Menu

```
Current Polynomial = 0

1- Add
2- Subtract
3- Multiply
4- Derivative
5- Find Degree
6- Find Value for specific x
7- Compare
8- Save to a text file
9- Save to a binary file
10-      Back to Main Menu
```

با انتخاب Add یک رشته دریافت میکند و با استفاده از آن یک چند جمله ای جدید میسازد و آن را به $Current\ Polynomial$ اضافه میکند.
با انتخاب $Subtract$ یک رشته دریافت میکند و با استفاده از آن یک چند جمله ای میسازد و آن را از $Current\ Polynomial$ کم میکند.
با انتخاب $Multiply$ یک رشته دریافت میکند و با استفاده از آن یک چند جمله ای میسازد و آن را در $Current\ Polynomial$ ضرب میکند و حاصل را در $Current\ Polynomial$ میریزد.
با انتخاب $Derivative$ مشتق $Current\ Polynomial$ را در $Current\ Polynomial$ ذخیره میکند.

توجه کنید که حاصل تمامی عملیات جمع، تفریق، ضرب و مشتق در $Current\ Polynomial$ ذخیره میشوند.

با انتخاب گزینه 5 درجه چند جمله ای چاپ میشود.

با انتخاب گزینه 6 یک عدد را به عنوان ورودی دریافت میکند و مقدار چند جمله ای را به ازای $x = input$ (برابر عدد ورودی) محاسبه و چاپ میکند.

با انتخاب گزینه 7 یک رشته دریافت میکند و با استفاده از آن یک چند جمله ای میسازد و با $Current\ Polynomial$ مقایسه میکند. برای مثال:

```
Current_Polynomial = x^2 + 3
Other_Polynomial = 100x + 10

Comparing Current_Polynomial with Other_Polynomial

Current_Polynomial > Other_Polynomial: True
Current_Polynomial >= Other_Polynomial: False
Current_Polynomial < Other_Polynomial: False
Current_Polynomial <= Other_Polynomial: False
Current_Polynomial == Other_Polynomial: False
```

با انتخاب گزینه 8 رشته ای به عنوان نام فایل دریافت میکند و چند جمله ای را در آن فایل **متنی** ذخیره میکند.

با انتخاب گزینه 9 رشته ای به عنوان نام فایل دریافت میکند و چند جمله ای را در آن فایل **باینری** ذخیره میکند.

با انتخاب گزینه 10 به منوی اصلی برنامه ($Main\ Menu$) بازمیگردیم.

در منوی اصلی با انتخاب گزینه 2 و یا 3 رشته ای به عنوان نام فایل دریافت میکند و چند جمله ای موجود در آن فایل را در یک شی از جنس $Polynomial$ به اسم $Current\ Polynomial$ قرار میدهد و منوی

$Polynomial\ Menu$ نمایش داده میشود. (گزینه 2 برای خواندن از فایل متنی و گزینه 3 برای خواندن از فایل باینری است).

به نکات زیر توجه کنید:

- توجه کنید که در بالای گزینه اول منو $Polynomial\ Menu$ باید مقدار $Current\ Polynomial$ را نشان بدهد.
- توجه کنید که در صورتی که ورودی نامعتبری در گزینه های $Polynomial\ Menu$ داده شد باید $Exception$ با پیغام مناسب ارسال شود.
- توجه کنید که اگر درفایل متنی، متن نامعتبری نوشته شده بود، هنگام خواندن چند جمله ای از فایل متنی، $Exception$ پرتاب شود.
- توجه کنید که تمامی $Exception$ های پرتاب شده باید دارای پیغام مناسبی باشند و در جای مناسبی $catch$ و مدیریت شوند یا آن پیغام را نمایش دهند.
- توجه کنید که هنگام کد نوشتن $Coding\ Style$ که در کلاس تدریس شده را رعایت کنید. $(CodingStyle.pdf)$

موفق باشید