بسم الله الرحمن الرحیم

دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

مبانی برنامه سازی به زبان C / C++

پروژه پایانی

دیفرانسیل

۳-۲ نفر

همه شما با نمایش یک عبارت ریاضی به صورت عادی آشنا هستید. رایج ترین صورت بیان یک عبارت به صورت میان وندی[1] می باشد. برای مثال عبارت زیر را در نظر بگیرید:

$$(x+y)-2*z \qquad (1)$$

در این شیوه نمایش اولیویت عملگرها با کمک پرانتز و از چپ به راست تعیین می شود. با وجودی که این عبارت ممکن است برای بسیاری از ما آشنا باشد ولی پیاده سازی آن در زبان های برنامه سازی مشکل است. واضح ترین دلیل آن هم تقدم عملگر ها بر همدیگر می باشند. برای حل این مشکل در بسیاری از برنامه های ریاضی از نمایش پیش وندی[2] یا نمایش پس وندی[3] استفاده می کنند.

برای مثال صورت پیش وندی و پس وندی عبارت ۱ به صورت زیر خواهد بود.

پیش وندی: $-+xy*2z$

پس وندی: $xy+2z*-$

هدف این پروژه این است که یک عبارت را به صورت میان وندی دریافت کرده و بتواند حاصل آنرا حساب کند و یا از آن مشتق بگیرد.

---

[1] *infix*
[2] *prefix*
[3] *postfix*

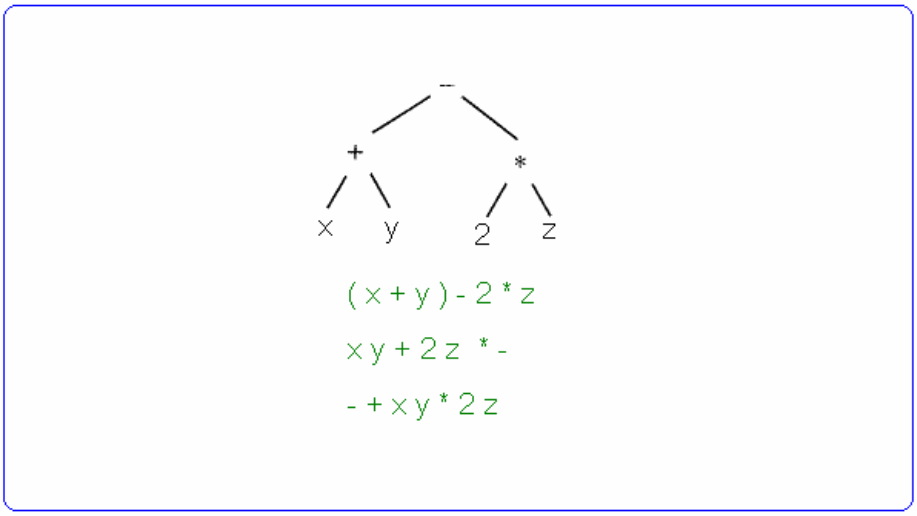
- **نمایش لهستانی[4]:**

در این شیوه بیان عبارت که عبارت ها بر اساس اولویت شان مرتب می شوند نمایش لهستانی (یا به عبارت دیگر پیش وندی یا پس وندی) نامیده می شود. محاسبه عبارت های اینچنینی آسان است چرا که کافی است عبارت ورودی را به ترتیب پیمایش کرده و باکمک یک پشته یا آرایه مقدار عبارت را حساب نماییم.

- **درخت عبارت:**

درخت عبارت یک درخت دودویی و داده ساختاری مناسب برای ذخیره سازی عبارت های ریاضی می باشد. نمونه ای از این درخت ها و عبارت های مربوطه را در شکل۲ می بینید. بدیهی است که با داشتن ریشه یک درخت و انجام عمل پیمایش inorder می توان صورت میان وندی عبارت را بدست آورد. پیمایش های preorder و postorder به ترتیب صورت پیش وندی و پس وندی عبارت را بدست می دهند.

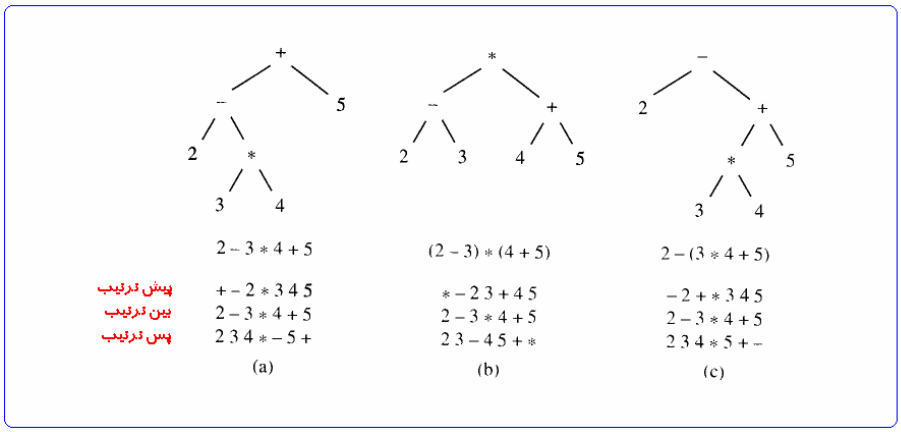
برای مثال درخت عبارت مثال ۱ به صورت زیر است:



پیمایش inorder درخت به صورت بازگشتی به صورت زیر است:

```
void inorder(node* x){
        inorder(x->left);
        cout << x->value << " " ;
        inorder(x->right);
}
```

همانطور که گفته شد با کمک این پیمایش بر روی یک درخت عبارت می توانید صورت میان وندی آن را تهیه کنید. پیمایش preOrder و postOrder هم به صورت مشابهی می باشند. ایجاد صورت کاملاً پرانتزی با داشتن درخت عبارت بدیهی است.



---

[4] *Polish Notation*

- **قوانین مشتق گیری:**

برای متغیر x می توانید از قواعد مشتق گیری زیر استفاده کنید:

$$\frac{d}{dx}k=0$$

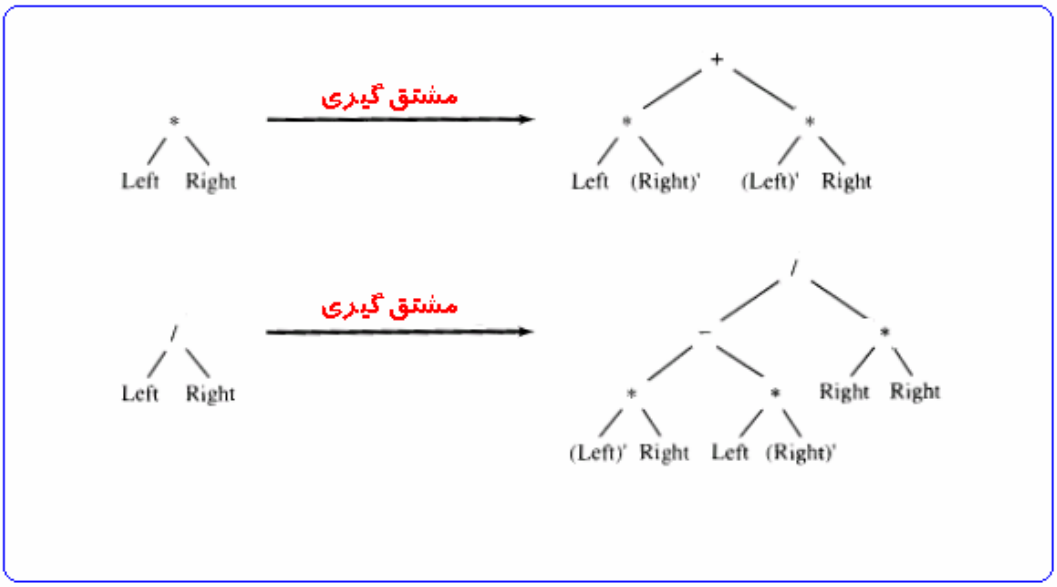
$$\frac{d}{dx}x=1$$

$$\frac{d}{dx}(F(x)\pm G(x))=\frac{d}{dx}F(x)\pm\frac{d}{dx}G(x)$$

$$\frac{d}{dx}(F(x)\times G(x))=F(x)\times\frac{d}{dx}G(x)+G(x)\times\frac{d}{dx}F(x)$$

$$\frac{d}{dx}\left(\frac{F(x)}{G(x)}\right)=\frac{G(x)\times\frac{d}{dx}F(x)-F(x)\times\frac{d}{dx}G(x)}{G(x)^2}$$

بعد از هر عمل مشتق گیری درخت عبارت شما به صورت زیر تغییر می نماید:



- **پیاده سازی درخت:**

برای پیاده سازی درخت در هر گره از دو اشاره گر به فرزند چپ و راست آن استفاده کنید. به علاوه ریشه درخت را به عنوان یک متغیر سراسری نگهداری نمایید.

```
struct node{
     node* left  ;
     node* right ;
     char  value ;
} *root ;
```

- **راهنمایی:**

توصیه می کنیم برای راحتی عمل مشتق گیری و همین طور محاسبه عبارت ابتدا عبارت را از صورت میان وندی به پیش وندی یا پس وندی تبدیل کرده و بعد درخت عبارت را از روی آن بسازید.

- **ورودی/خروجی:**

ورودی خود را از standard input بخوانید و خروجی تان را بر روی standard output بنویسید.[5]

۱. **ورودی برنامه:**

ورودی برنامه یک چند جمله ای از از متغیر x به صورت **میانوندی** می باشد. این چند جمله ای ممکن است شامل پرانتز، عمل ضرب یا عمل تقسیم یا عدد ثابت باشد. فرض کنید اعداد ثابت همه یک رقمی هستند. در نتیجه در ساختار گره استفاده از نوع داده char برای نگه داری همه اطلاعات کافی می باشد.
بعد از آن چند مقدار ورودی برای x (از نوع float) داده می شود که شما می بایست مقدار عبارت را برای آن محاسبه نمایید.

۲. **خروجی های برنامه:**

- برنامه شما در درجه اول می بایست عبارت ورودی را به صورت کاملا پرانتزی بنویسد.
- سپس باید از عبارت مورد نظر مشتق بگیرید.
- در انتها باید مقدار عبارت را به ازای برخی مقادیر ورودی برای متغیر ها محاسبه کنید.

پیاده سازی موارد زیر مشمول نمره اضافی می باشد:
نیازی به ساده سازی عبارت نیست. ولی ساده کردن آن یا اعمال اولویت ها به صورت دقیق مشمول نمره اضافی خواهد بود.
به علاوه اگر بتوانید مکانیزم مدیریت استثناء[6] را در برنامه تان (با کمک exception handling زبان c++ یا پیاده سازی توسط خودتان) را پیاده کنید به صورتی که بتواند خطاهای ساده مانند تقسیم بر صفر را تشخیص دهد نمره اضافه دریافت خواهید کرد. در صورت رخداد خطا فقط پیغام error را بر روی صفحه چاپ کرده و به کار خود ادامه دهید.
در انتها اگر علاوه بر مقدار عبارت مقدار مشتق آنرا هم حساب کرده و چاپ کنید نمره اضافی تری دریافت می کنید.

---

[5] برای این کار از *header file* های *stdio.h*(برای زبان C) و فایل *iostream* (برای زبان C++) استفاده کنید.

[6] *Exception handling*

مثال:

ورودی:

```
(x*x*x)-(x)/(2*x-3)
4.3
1.5
0
```

خروجی:

```
((((x*x)*x)-x)/((2*x)-3))
((3*x*x-1)*(2*x-3)-(2)*(x*x*x-x)/((2*x-3)*(2*x-3))
13.429821
error
0
```

- ***توجه بسیار مهم:***

برنامه شما باید در زمان تحویل کمپایل و اجرا شود و بتواند برای تست های ورودی خروجی صحیح ایجاد کند.

کد منبع پروژه خودتان را به همراه مستندات مربوط به آن را در غالب یک فایل فشرده[۷] به نام `differential` تا زمان تحویل به آدرس `ce153c@gmail.com` ارسال نمایید. فیلد subject نامه خود را برابر با *Project2-differential-(8510xxxx-8510xxxx-8510xxxx)* قرار دهید که در آن *xxxx* چهار رقم انتهایی شماره دانشجویی اعضای تیم شما می باشد.

لطفاً توجه کنید که تحویل پروژه ها به صورت حضوری می باشد.

موفق باشید.

[7] با فرمت *zip*، *rar* یا *tar*