به نام خدا

دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

پروژه پایان ترم درس مبانی کامپیوتر و برنامه‌نویسی

(ماشین حساب مهندسی)

گردآورندگان:

علی جانعلی‌زاده و محمدامین محمدی

دکتر شیری

پاییز ۹۵

ویژگی‌های پروژه

در این پروژه شما باید ماشین حسابی پیاده‌سازی کنید که علاوه بر محاسبه‌ی عبارات شامل عملگرهای توان، ضرب، تقسیم، جمع و تفریق (با ترتیب اولویت)، قابلیت‌های زیر را داشته باشد:

- محاسبه‌ی عبارات توانی که توان عدد صحیح یا کسر گویاست مانند:

2.35=2.3^5

e^x=e^x=exp(x)

- محاسبه‌ی عبارات مثلثاتی مانند:

sin(x) cos(x) tan(x) cot(x) sec(x) csc(x)

- محاسبه‌ی تابع ln(x) برای مثال:

sinh(x) cosh(x) tanh(x) coth(x)

- شناخت عبارت‌های PI و e و exp به عنوان متغیر ورودی (این عبارات غیر case-sensative، به این معنی که هم Pi و هم pI معادل یک عدد هستند).

Sin(pi/2)

Cos(e^23/Pi)

eXp(pI/e^2)

- محاسبه‌ی عبارات به صورت تودرتو تا هر چند مرحله. برای مثال:

Sin(Cos(e^Tanh(exp(p))))

- نمایش پیام خطا در صورتی که ورودی از لحاظ محاسبات ریاضی غیرمجاز است. برای مثال:

Ln(-10)

- (امتیازی) نمایش پیام خطا در صورتی که ورودی از لحاظ **syntax** ریاضی مشکل داشته باشد. برای مثال:

exp(2

Sin(+)

توجه: برای پیاده سازی توابع مثلثاتی می‌توانید از تابع‌های موجود کتابخانه‌ی math.h استفاده کنید اما اگر این تابع‌ها را نیز خودتان و با استفاده از بسط تیلور توابع پیاده سازی کنید نمره‌ی امتیازی به شما تعلق خواهد گرفت.

جزئیات پیادسازی

برای پیادسازی پروژه ابتدا باید ورودی را **tokenize** کنید و یک لیست **word** از ورودی بسازید (یک لیست پیوندی). منظور از کلمه یا **word** یک عدد یا اپراتور یا پرانتز یا یک تابع محاسباتی است. در ورودی کلمه‌ها با یک **Space** از هم جدا می‌شوند. مثال‌هایی از ورودی و لیست **tokenize** شده:

Input	Tokenized List
(32)	,(,32,),
Sin,x,	Sin x,
x,+,2,*,exp,(,y,),*,2,	x + 2 * exp (y) * 2,

سپس این لیست را باید جوری تغییر دهیم که برای کامپیوتر قابل فهم و محاسبه باشد. برای این مسئله باید لیست ورودی که در حالت Infix است را به حالت Postfix تبدیل کنیم (توضیح اینکه **Postfix** و **Infix** چه چیزهایی هستند و چگونگی تبدیل **Infix** به **Postfix** در جلسه‌ي توجیهی داده خواهد شد. همچنین لینک‌هایی برای توضیح بیشتر این عبارات در این فایل وجود دارد).

این تبدیل را با استفاده از الگوریتم Shunting-Yard انجام می‌دهیم که برای این‌کار نیاز به Stack داریم و باید آن را پیادسازی کنیم. سپس این لیست **Postfix** را با استفاده از یک تابع بازگشتی تبدیل به جواب عبارت می‌کنیم. در حین این امر اگر ورودی‌ها در حدود دامنه‌ی تابع نبودند عملیات را متوقف و خطا را نمایش می‌دهیم. دقیقاً باید نشان دهیم کجا خطا اتفاق افتاده است. مثلاً:

→ Ln (2 * Sin (-2*pi))

Error: Invalid Negative Argument for Ln

→ Sinh ((2 ^ pi) * pi) / Ln (Cos (4 * pi) ^ exp (-e))

Error: Divided by Zero

→ Tan (pi / 2)

Error: Invalid Argument (out of Tan function domain)

→ -87.5 ^ (1 / 4)

Error: Invalid Negative Argument for Radical

آخرین مهلت آپلود پروژه ۲۹ دی ماه می‌باشد و در تاریخ ۳۰ دی ماه تحویل حضوری پروژه انجام می‌شود.

موفق باشید.