OOP大作业：计算图

一.第二阶段开始之前

每组应该收到4份其他组的代码。组的编号已被打乱，请大家自觉维护不要讨论打分相关的问题。

每组应该首先阅读其他4个组的代码，联系第二阶段需求，分析代码是否易于进一步开发。你需要对这些代码进行排序，并简要分析每份代码的优劣性。排名不得并列，对每组代码的评论至少50字。你的排序及评价会由助教评估是否合理，占你大作业总得分的5%。别人对你的评价占你大作业总得分的10%。

你继续开发使用的代码应当是你排序中的第一位。你也可以选择直接使用自己的代码，这时第二阶段得分会受到**分数x0.8**的惩罚。但相应地，你可以节省看代码的时间，通过完成更多的功能提高自己的得分。

二.第二阶段需求

在以下的开发过程中，你需要尽量保持第一阶段其他组的接口不被修改（可以增加）。在第二阶段开发结束后，第一阶段的测试程序应当不需改动，直接能够正确运行。（因为往往在真实项目中，修改接口会牵一发而动全身，容易浪费大量时间）。如果实在需要改动，会按照改动幅度和合理性酌情扣分。最后你需要提交原来的main.cpp和完成第二阶段的main.cpp，我们会再次测试第一阶段的数据。

基础需求（占25%，如果只完成基础需求，大作业总分上限为85%）：

1. 你需要提交第一阶段的main.cpp，并保证仍能通过第一阶段的测试，**占5%**。注意你拿到的代码可能仍有bug，你应该尝试修复你使用的这份代码中可能存在的bug。
2. 实现以下功能，**共占5%**：

   (1) 实现用于调试的Assert运算。当计算图构建完成后，计算的过程对用户是不可见的，这时候调试往往会产生困难。这个运算符输入1个变量，输出始终为0，但是却能够在变量小于等于0时这个条件时报错。这项功能使得你能够用Assert(x)来保证**x>0**功能。

   (2) 实现捆绑Bind运算。只有Assert的输出被计算时，才会报错。但Assert的输出没有什么意义。Bind运算使得一个变量求值时能顺便求另一个变量的值，但不影响计算结果。形式化地说，这个运算符输入2个变量，输出第1个变量的值，但会对第2个变量求值。

   - 输入样例：

```
4
x P
y P
z C 3.0
t C 2.0
5
a = x + y
b = x + t
d = b - y
c = ASSERT d
res = BIND a c
2
EVAL res 2 x 1.0 y 2.0
EVAL res 2 x 1.0 y 4.0
```

- 输出样例：

```
3.0000
ERROR: Assertion failed
```

3. 实现链式求导功能。计算图最大的一个功能就是自动求导，你需要实现一个grad函数。当对一个变量L求导时，调用L.grad()，可以得到对所有中间结点的偏导数，偏导数仍然是一个变量，但请注意复杂度，我们对一个结点求导，需要在O（N）时间内得到所有中间的导数变量。

   你可以假设只有+*两种运算。（**占10%**。若实现所有运算的自动求导为拓展功能，另外加分）。

   保证测试一次eval里不涉及2次求导，保证不涉及高阶导（若实现可另外加分）。

   - 样例1：
     - 输入样例：

     ```
     3
     x P
     a C 3
     b C 2
     7
     y = x * a
     z = x * b
     L = y + z
     g = GRAD L
     gx = g AT x
     gy = g AT y
     t = gx + gy
     3
     EVAL gx 1 x 3.0
     EVAL gy 1 y 3.0
     EVAL t 1 x 3.0
     ```

     - 输出样例1：

     ```
     5.0000
     1.0000
     6.0000
     ```

     提示：

     $$\begin{aligned}\frac{\partial L}{\partial x} &= \frac{\partial L}{\partial y}\frac{\partial y}{\partial x} + \frac{\partial L}{\partial z}\frac{\partial z}{\partial x} \\ &= 1 * Constant(3) + 1 * Constant(2) \\ &= 5\end{aligned}$$

     计算图模型为 $x \to y/z \to L$

     我们可以反向进行计算，按照 $L \to y/z \to x$

     - 求出 $\frac{\partial L}{\partial L} = 1$
     - 求出 $\frac{\partial L}{\partial y}$，因为 $L = x + y$，所以答案为 1

        - 求出 $\frac{\partial L}{\partial z}$，因为 $L = x + y$，所以答案为 1
        - 从 y 回到 x，$\frac{\partial L}{\partial y}\frac{\partial y}{\partial x} = 1 \cdot Constant(3) = 3$
        - 从 z 回到 x，$\frac{\partial L}{\partial z}\frac{\partial z}{\partial x} = 1 \cdot Constant(2) = 2$
        - 总结所有到 x 的导数，$3 + 2 = 5$
    - 样例2:
        - 输入样例:

```
3
x P
y P
z C 2
5
t = x * x
L = t * y
g = GRAD L
gt = g AT t
gx = g AT x
gy = g AT y
3
EVAL gt 2 x 3.0 y 2.0
EVAL gx 2 x 3.0 y 2.0
EVAL gy 2 x 3.0 y 2.0
```

        - 输出样例:

```
2.0000
12.0000
9.0000
```

提示:

$$\begin{aligned}\frac{\partial L}{\partial x} &= \frac{\partial L}{\partial t}(\frac{\partial t}{\partial x} + \frac{\partial t}{\partial x}) \\ &= 2 \cdot (x + x) \\ &= 12\end{aligned}$$

计算图模型为 $x \rightarrow t/y \rightarrow L$

我们可以反向进行计算，按照 $L \rightarrow t/y \rightarrow x$

        - 求出 $\frac{L}{L} = 1$
        - 求出 $\frac{\partial L}{\partial y}$，因为 $L = t * y$，所以答案为 $\frac{L}{y} = t = 9$
        - 求出 $\frac{\partial L}{\partial t}$，因为 $L = t * y$，所以答案为 $\frac{L}{t} = y = 2$
        - 从 t 回到 x （第一操作数），因为 $t = x * x$，$\frac{\partial L}{\partial t}\frac{\partial t}{\partial x} = y * x = 6$
        - 从 t 回到 x （第二操作数），因为 $t = x * x$，$\frac{\partial L}{\partial t}\frac{\partial t}{\partial x} = y * x = 6$
        - 总结所有到 x 的导数 12
    - 样例3:
        - 输入样例:

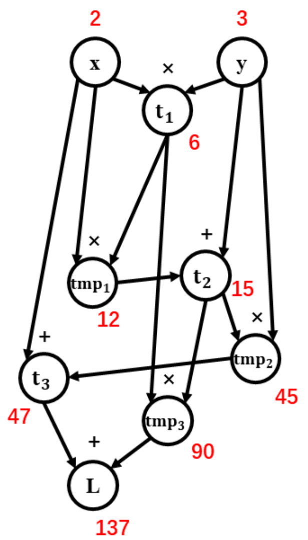
```
3
```

```
x P
y P
11
t1 = x * y
tmp1 = x * t1
t2 = tmp1 + y
tmp2 = y * t2
t3 = tmp2 + x
tmp3 = t1 * t2
L = tmp3 + t3
g = GRAD L
gx = g AT x
gy = g AT y
res = gx + gy
1
EVAL res 2 x 2.0 y 3.0
```

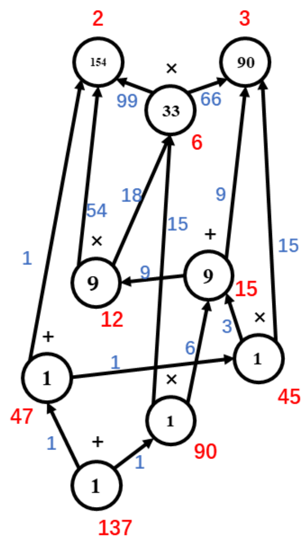
- 输出样例:

```
244.0000
```

提示：反向传播进行计算。+运算两侧的变量导数为1，*运算两侧的导数为另一个变量，最后将到同一变量的导数加起来。下面有个计算的示意图:



正向计算



反向计算

4. 实现牛顿迭代法解方程。请参考牛顿迭代法。利用计算图的自动求导功能，求 n 次方程 $a_nx^n+a_{n-1}x^{n-1}+\cdots+a_2x^2+a_1x+a_0=0$ 的根。你需要输出每一步迭代x的值。 **（占5%）**
   - 输入样例:

     第一行为需要求解的方程个数 m，接下来 m 行为 m 个方程，每行第一个数为方程的的最高次数 n， 接下来 $n+1$ 个数分别为 $a_n,a_{n-1},\cdots,a_0$，再接下来一个数为迭代的初始值

```
2
2 2.0 1.0 0.0 -2.0
5 1.0 0.0 3.0 2.0 1.0 1.0 -1.0
```

- 输出样例：

  对于每个方程，输出 x 在每一次迭代的值（保留四位小数），最高只需迭代 5次

  ```
  -1.1429 -0.7314 -0.5556 -0.5051 -0.5001
  -0.8182 -0.7366 -0.7213 -0.7209 -0.7209
  ```

拓展需求

1. 实现给Variable赋值的Assign运算。这和第一阶段的set函数不同，Assign运算是在运算图中的一个操作，和Print等运算是一样的。该函数输入两个参数，第一个是Variable，第二个是变量。该运算的输出为第二个变量，运行该运算的后果是Variable的值变为第二个变量的值。

   注意这个运算可能会因为求值顺序的不同影响结果。比如运行以下伪代码：

   ```
   x 为Variable 初始化为1
   y = x + 1
   res = y + Assign(x,2)
   ```

   此时，若先算让 $x = 2$，则结果为5；先算 y，则结果为4。为了解决这个问题，我们要求，Assign的赋值不会在本次运算时起效。即使我们先算Assign，$y = x + 1$中的x值仍然为1。在一次eval中对同一个Variable多次赋值是未定义的，也就是说你不用考虑这种情况。（最多+10%）

   - 输入样例：

     ```
     2
     x V 1.0
     y V 2.0
     5
     a = ASSIGN x y
     b = a * y
     c = ASSIGN y b
     d = c * x
     res = d * y
     3
     EVAL res 0
     EVAL x 0
     EVAL y 0
     ```

   - 输出样例：

     ```
     8.0000
     2.0000
     4.0000
     ```

2. 实现完整的自动求导功能，你需要为所有可以求导的运算符添加求导方法，使grad()函数能够在有除+*运算符以外的运算符中也能完成自动求导。需要支持的运算符有-，/，sin，exp，log。Print、Assert、Bind、比较运算符虽然不能求导，但也需要为其选定合适的处理方式，便于用户使用。（根据实现情况，最多+10%）

3. 实现多维矩阵Tensor的运算图，需要支持Placeholder、Constant、Parameter等，并支持我们已经完成的各种运算，注意矩阵运算需要匹配大小，不匹配时应该检查是否能broadcast或者报错。若只实现基础运算（基础需求中的所有运算，最多+5%），实现broadcast和各类reshape、concat等改变矩阵大小的操作(最多+5%)，实现矩阵的自动求导（最多+10%）。上述操作可以参考TensorFlow的接口。
4. 运算图的session和存取功能。Session的意思即为会话，在TensorFlow中，同一个运算图的Variable在不同session中的值可以是不同的。因为所有的parameter的值都是和session绑定的，即在session中才能进行eval操作（为了兼容，如果eval不带参数可以认为是在一个默认的session中进行的）。你需要实现session，实现session与parameter值的绑定 (最多+10%)；并能够从文件存储和读取session中的parameter的值（最多+5%）。
5. 实现梯度下降求优化最小二乘法。对于一个函数f(x)=ax+b，最小二乘法需要优化f(x)和真实的y的均方误差Loss。这里将a,b看做参数，求均方误差的最小值有一种梯度下降(gradient descent)的方法。即每次按照Loss对a、b的导数，将a和b向f(x)较小的地方移动一点距离。请自己构造一组数据，列出最小二乘法需要优化的均方误差，使用梯度下降方法优化，并与最小二乘法的公式计算得到的结果进行比对。你需要自己编写测试程序，展示你的结果。(对于自变量只有1个的情况，最多+10%；对于自变量有多个的情况，最多 +20%)
6. 基于神经网络的手写数字识别（或其他简单任务）。现在计算图常用于神经网络的学习之中，请查阅资料，了解并实现自己的MLP多层感知机模型。简单地说，MLP网络实际上就是一个向量x，经过一个线性变换变为Mx，再经过一次非线性变换f(Mx) （非线性变换经常采用sigmoid或者relu函数）。重复这个操作多次，最后获得结果y，与真实结果进行比较，产生一个误差。请你列出需要优化的误差公式，使用梯度下降进行优化。你需要自己编写测试程序，对结果进行分析。（和5不叠加，最多+30%）
7. 其他你觉得有意义的改进。

   注意：对于多数拓展需求，你需要自己编写测试程序，保证展示你的结果。

三.项目限制

只能使用C++完成，禁止使用第三方库（不包括STL或编译器自带的库）。

四.提交要求

第12周周日(5.19)：你对其他组的排序及评价

第17周周日(6.23)：

1. 提交你编写的计算图库，请遵守OOP的设计规范。
2. 给出示例代码，保证能够运行第一阶段的测试程序(编译生成main1)，第二阶段的测试程序（编译生成main2），牛顿迭代法程序（main3），以及你完成的其他功能的测试程序。
3. 给出一份实验报告，展示你们最终完成的功能，测试程序运行的结果。并分析实现的好和需要改进的地方。
4. 在readme.md或readme.txt中写明程序的运行环境。（你的库在跨平台上最好能在跨平台下运行，并且依赖项尽量少，否则我们将很难运行你的代码。）
5. 给出一个说明文档（markdown或word），写清计算图库的结构、封装、接口。（字数不限，目标是能让别人在最短时间内明白项目结构，太短太长都不太好。）