

CS205 C/ C++ Program Design - Assignment 1

Please implement a calculator which can multiply two integers.

Requirements:

1. When you run the program as follows:

```
$. /mul
Please input two integers
2 3
```

It will output 6.

2. If you input some non-integer numbers, the program can tell the user that the input is wrong.

```
./mul
Please input two integers
a 2
```

3. If you input some big integers as follows

```
./mul
Please input two integers
1234567890 1234567890
```

What will happen? Please describe some possible solutions.

4. Some others which can improve the program.

Rules:

1. Please submit your assignment report before its deadline. After the deadline (even 1 second), **0 score!**
(For students who register this course before Sep. 6, they should submit their assignment reports before 23:59 on Sep. 13. For the rest students they should submit their assignment reports in one week after they register.)
2. If you only implement **requirement 1, the upper boundary of your score is 80**. For a better score, you should implement the rest requirements. Your score will also depend on the quality of your source code and your report. Your report should be easy to understand and describe your work well, especially the highlights of your work.
3. Please pay more attention to your **code style**. After all this is not ACM-ICPC contest. You have enough time to write code with both correct result and good code style. You will get deduction if your code style is terrible. You can read Google C++ Style Guide (<http://google.github.io/styleguide/cppguide.html>) or some other guide for code style.

Report Template:

CS205 C/ C++ Program Design

Assignment 1

Name: 刘旭坤, SID: 11912823

Part 1. Source Code

```
/*
 * @Author: satan
 * @Date: 2020-09-12 23:43:28
 * @LastEditTime: 2020-09-13 00:53:35
 * @LastEditors: satan
 * @Description: A*B And A+B Problem
 * @FilePath: /C++/mul.cpp
 */
#include <bits/stdc++.h>
using namespace std;
class Oper
{
public:
    vector<int> Num;
    vector<char> Name;
    int power;
    bool isPositive;
    string Error;
    string Ans;
    /**
     * @description: init
     * @param {}
     * @return {}
     */
    Oper()
    {
        isPositive = true;
        power = 1;
    }
    /**
     * @description: Reduce The Formula
     * @param {Oper Oper}
     */
}
```

```

    * @return {}
    */
void Connect(Oper a, Oper b)
{
    int L;
    this->Ans += a.to_String();
    if (b.isPositive)
    {
        this->Ans += '+';
        this->Ans += b.to_String();
    }
    else
    {
        this->Ans += b.to_String();
    }
    if (this->Ans[0] == '0')
    {
        this->Ans = this->Ans.substr(2);
    }
    L = this->Ans.length() - 1;
    if (this->Ans[L] == '0' && (this->Ans[L - 1] == '-'
' || this->Ans[L - 1] == '+'))
    {
        this->Ans = this->Ans.substr(0, L - 1);
    }
}
/**
 * @description: Process negative numbers
 * @param {int}
 * @return {}
 */
void reProcess(int L)
{
    for (int i = 0; i <= L; i++)
    {
        this->Num[i] = -this->Num[i];
    }
    for (int i = 0; i < L; i++)
    {
        if (this->Num[i] < 0)
        {
            this->Num[i] += 10;
            this->Num[i + 1] -= 1;
        }
    }
}

```

```

    }
    for (int i = 0; i <= L; i++)
    {
        this->Num[i] = -this->Num[i];
    }
}

/**
 * @description: Change an Oper to string
 * @param {}
 * @return {string}
 */
string to_String()
{
    string ans;
    int L = this->Num.size() - 1;
    while (this->Num[L] == 0 && L >= 1)
    {
        L--;
    }
    if (L == 0)
    {
        if (this->Num[0] == 0)
        {
            ans += '0';
            return ans;
        }
        else if (this->Num[0] == 1 || this->Num[0] == -1)
        {
            if (this->Name.size() != 0)
            {
                L = this->Name.size();
                if (this->isPositive == false)
                {
                    ans += '-';
                }
                for (int i = 0; i < L; i++)
                {
                    ans += this->Name[i];
                }
                if (this->power != 1 && this->power != 0)
                {
                    ans += '^';
                    ans += to_string(this->power);
                }
            }
        }
    }
}

```

```

        return ans;
    }
}

if (this->isPositive == false)
{
    ans += '-';
}
for (int i = L; i >= 0; i--)
{
    ans += this->isPositive ? to_string(this->Num[i]) : to_string(-this->Num[i]);
}
L = this->Name.size();
for (int i = 0; i < L; i++)
{
    ans += this->Name[i];
}
if (this->power != 1 && this->power != 0)
{
    ans += '^';
    ans += to_string(this->power);
}
return ans;
}

/**
 * @description: Output An Oper
 * @param {}
 * @return {}
 */
void out()
{
    if (this->Ans.length() > 0)
    {
        cout << this->Ans << endl;
        if (this->Error.length() > 0)
        {
            cout << this->Error << endl;
            return;
        }
        return;
    }
    cout << to_String() << endl;
    return;
}

```

```

}
/**
 * @description: Input An Oper
 * @param {}
 * @return {}
 */
void in()
{
    char tmp = getchar();
    while (tmp == ' ' || tmp == '\n' || tmp == '\r')
    {
        tmp = getchar();
    }
    if (tmp == '-')
    {
        this->isPositive = false;
        tmp = getchar();
    }
    while (tmp >= '0' && tmp <= '9')
    {
        this->Num.push_back(this->isPositive ? tmp - '0' : -
(tmp - '0'));
        tmp = getchar();
        if (tmp == -1)
            break;
    }
    while (tmp != ' ' && tmp != '\n' && tmp != '^' && tmp != '\r')
    {
        if (tmp == -1)
            break;
        this->Name.push_back(tmp);
        tmp = getchar();
    }
    if (tmp == '^')
    {
        this->power = 0;
        tmp = getchar();
        bool op = true;
        int Tmp = 1;
        if (tmp == '-')
        {
            tmp = getchar();
            op = false;
        }
    }
}

```

```

        while (tmp >= '0' && tmp <= '9')
        {
            if (tmp == -1)
                break;
            this->power *= 10;
            this->power += tmp - '0';
            tmp = getchar();
        }
        if (!op)
        {
            this->power = -this->power;
        }
    }
    if (this->Num.size() == 0)
    {
        if (this->isPositive)
            this->Num.push_back(1);
        else
            this->Num.push_back(-1);
    }
    if (this->Num[0] == 0)
        this->isPositive = true;
    if (this->power == 0)
    {
        this->Name.clear();
    }
    reverse(this->Num.begin(), this->Num.end());
}

/**
 * @description: A + B (Don't Have Non-integer Numbers)
 * @param {Oper}
 * @return {Oper}
 */
Oper Plus(const Oper &b)
{
    Oper c;
    int L = max(this->Num.size(), b.Num.size());
    c.Num.resize(L + 1);
    this->Num.resize(L + 1);
    for (int i = 0; i < L; i++)
    {
        if (i < b.Num.size())
        {
            c.Num[i] += this->Num[i] + b.Num[i];

```

```

    }
    else
    {
        c.Num[i] += this->Num[i] + 0;
    }

    c.Num[i + 1] += c.Num[i] / 10;
    c.Num[i] %= 10;
    if (c.Num[i] < 0)
    {
        c.Num[i] += 10;
        c.Num[i + 1] -= 1;
    }
}

while (L >= 1 && c.Num[L] == 0)
{
    L--;
}
if (c.Num[L] < 0)
{
    c.isPositive = false;
    c.reProcess(L);
}
L = b.Name.size();
for (int i = 0; i < L; i++)
{
    c.Name.push_back(b.Name[i]);
}
c.power = b.power;
return c;
}

/**
 * @description: Comper Non-integer Part
 * @param {Oper}
 * @return {bool}
 */
bool JudgeName(const Oper &b)
{
    if (this->Name.size() + b.Name.size() == 0)
    {
        return true;
    }
    else
    {

```



```

        if (this->Name.size() != b.Name.size())
        {
            return false;
        }
        else
        {
            int L = this->Name.size();
            for (int i = 0; i < L; i++)
            {
                if (this->Name[i] != b.Name[i])
                {
                    return false;
                }
            }
            return true;
        }
    }
}

/**
 * @description: Overload And Classified discussion
 * @param {Oper}
 * @return {Oper}
 */
Oper operator+(const Oper &b)
{
    Oper c;
    int l = max(this->Num.size(), b.Num.size());
    if ((!JudgeName(b)) || this->power != b.power)
    {
        c.Error = "WARNING: You Have Input Some Non-integer Numbers,This Formula Can Not Be Reduced!";
        if (this->isPositive && b.isPositive)
        {
            c.Connect(*this, b);
        }
        else if (!(this->isPositive | b.isPositive))
        {
            c.Connect(*this, b);
        }
        else
        {
            if (this->isPositive)
            {
                c.Connect(*this, b);
            }
        }
    }
}

```

```

        }
        else
        {
            c.Connect(b, *this);
        }
    }
    return c;
}
else
{
    return c = this->Plus(b);
}
}
/**
 * @description: Overload multiplication
 * @param {Oper}
 * @return {Oper}
 */
Oper operator*(const Oper &b)
{
    Oper c;
    int L = max(this->Num.size(), b.Num.size()), l1 = this->Num.size(), l2 = b.Num.size();
    c.Num.resize(2 * L + 1);
    c.isPositive = (!(this->isPositive ^ b.isPositive));
    for (int i = 0; i < l2; i++)
    {
        for (int j = 0; j < l1; j++)
        {
            c.Num[i + j] += b.Num[i] * this->Num[j];
        }
    }
    for (int i = 0; i < 2 * L; i++)
    {
        c.Num[i + 1] += c.Num[i] / 10;
        c.Num[i] %= 10;
    }
    if (JudgeName(b) == true && b.Name.size() > 0)
    {
        L = this->Name.size();
        for (int i = 0; i < L; i++)
        {
            c.Name.push_back(this->Name[i]);
        }
    }
}

```

```

        c.power = this->power + b.power;
    }
    else
    {
        L = this->Name.size();
        for (int i = 0; i < L; i++)
        {
            c.Name.push_back(this->Name[i]);
        }
        if (this->power != 1)
        {
            c.Name.push_back('^');
            for (int i = 0; i < to_string(this->power).size(); i++)
            {
                c.Name.push_back(to_string(this->power)[i]);
            }
        }
        L = b.Name.size();
        for (int i = 0; i < L; i++)
        {
            c.Name.push_back(b.Name[i]);
        }
        if (b.power != 1)
        {
            c.Name.push_back('^');
            for (int i = 0; i < to_string(b.power).size(); i++)
            {
                c.Name.push_back(to_string(b.power)[i]);
            }
        }
    }
    if (c.power == 0)
    {
        c.Name.clear();
    }
    return c;
}

};

int main()
{
    puts("Please input two integers");
    Oper A, B, C;
    A.in();
    B.in();

```

```

    C = A * B;
    cout << "A * B =";
    C.out();
    A = A + B;
    cout << "A + B =";
    A.out();
    return 0;
}

```

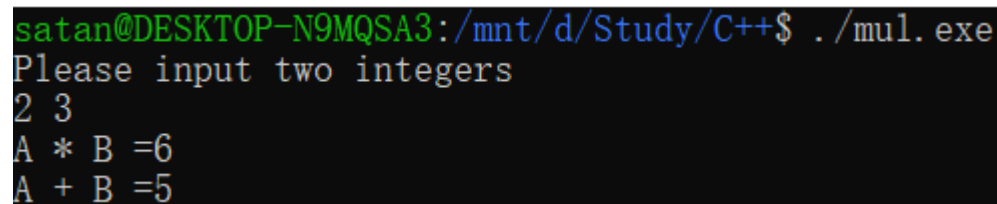
Part 2. Result & Verification

In this part, you should present the result of your program by listing the output of test cases and optionally add a screen-shot of the result.

Test case #1:

2 3

Screen-short for case #1:



```

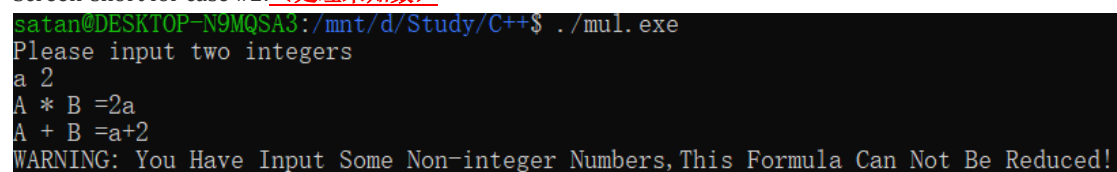
satan@DESKTOP-N9MQSA3:/mnt/d/Study/C++$ ./mul.exe
Please input two integers
2 3
A * B =6
A + B =5

```

Test case #2:

a 2

Screen-short for case #2: (处理未知数)



```

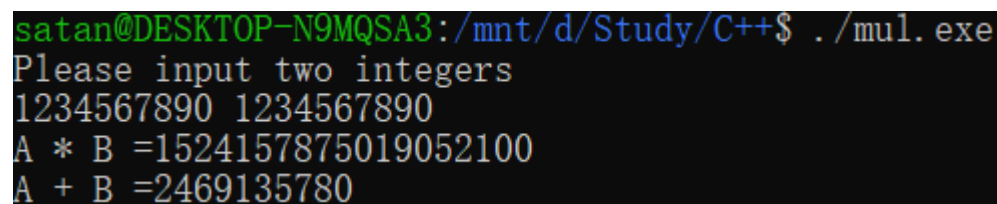
satan@DESKTOP-N9MQSA3:/mnt/d/Study/C++$ ./mul.exe
Please input two integers
a 2
A * B =2a
A + B =a+2
WARNING: You Have Input Some Non-integer Numbers, This Formula Can Not Be Reduced!

```

Test case #3:

1234567890 1234567890

Screen-short for case #3: (大数运算)



```

satan@DESKTOP-N9MQSA3:/mnt/d/Study/C++$ ./mul.exe
Please input two integers
1234567890 1234567890
A * B =1524157875019052100
A + B =2469135780

```

Test case #4:

-9a¹¹⁰ 10a⁻¹⁰⁹

Screen-short for case #4: (乘方运算)

Test case #5:

-1 a

Screen-short for case #5: (负数运算与化简)

```
satan@DESKTOP-N9MQSA3:/mnt/d/Study/C++$ ./mul.exe
Please input two integers
-1 a
A * B =-a
A + B =a-1
WARNING: You Have Input Some Non-integer Numbers,This Formula Can Not Be Reduced!
```

-1 a

```
satan@DESKTOP-N9MQSA3:/mnt/d/Study/C++$ ./mul.exe
Please input two integers
-1 a
A * B ==a
A + B ==a-1
WARNING: You Have Input Some Non-integer Numbers, This Formula Can Not Be Reduced!
```

$$-10a^{-1}b^{10}$$

```
satan@DESKTOP-N9MQSA3:/mnt/d/Study/C++$ ./mul.exe
Please input two integers
-10a^-1 b^10
A * B = -10a^-1b^10
A + B = b^10-10a^-1
WARNING: You Have Input Some Non-integer Numbers,This Formula Can Not Be Reduced!
```

a 0

```
satan@DESKTOP-N9MQSA3:/mnt/d/Study/C++$ ./mul.exe
Please input two integers
a 0
A * B =0
A + B =a
WARNING: You Have Input Some Non-integer Numbers,This Formula Can Not Be Reduced!
```

[illegible]

```

root@DESKTOP-7066243:~/c++/Study/C++$ ./mi.exe
Please input two integers
1000000000
1
1000000000
1

```

$$-10a^2 - 15a^2$$

```
satan@DESKTOP-N9MQSA3:/mnt/d/Study/C++$ ./mul.exe
Please input two integers
-10a^2 -15a^2
A * B =150a^4
A + B =-25a^2
```

$$a^{-1}a$$

Screen-short for case #10: (幂为负数的特殊情况)

```
satan@DESKTOP-N9MQSA3:/mnt/d/Study/C++$ ./mul.exe
Please input two integers
a^-1 a
A * B =1
A + B =a^-1+a
WARNING: You Have Input Some Non-integer Numbers,This Formula Can Not Be Reduced!
```

Part 3. Difficulties & Solutions, or others

1. 难点：对于非整数的化简与计算。
解决：对未知数的处理分成多种情况讨论。
2. 难点：乘方的引入，使得考虑的特殊情况（如：0 次方等）进一步增加。
解决：在运算和输出过程中增加判断，强制更改部分结果。
3. 难点：支持负数运算。
解决：增加符号位，对加法部分进行再次处理。
4. 难点：隐性 bug：在 Linux, Windows 与 MacOS 下个别符号不同会引起读入错误。
解决：增加'\r'的判断，将其也作为终止字符。
5. 难点：对于 $x+0/0+x$ 类型的化简。
解决：将表达式化为 string 后匹配处理。
6. 难点：不确定输入数字大小。
解决：高精度运算+vector 保存。