C/C++ Pro 1 – 设计一个乘法计算器

姓名：林洁芳

学号：12011543

第一部分：问题思路和分析

一、问题分析

1、 本次Pro要求做一个能够计算乘法的计算器，首先考虑的就是整数的乘法，但由于整数的数值可以超过INT_MAX 和 LONG_MAX，所以考虑用自己写一个计算大数乘法的方法来代替C++自带的乘法计算方法；

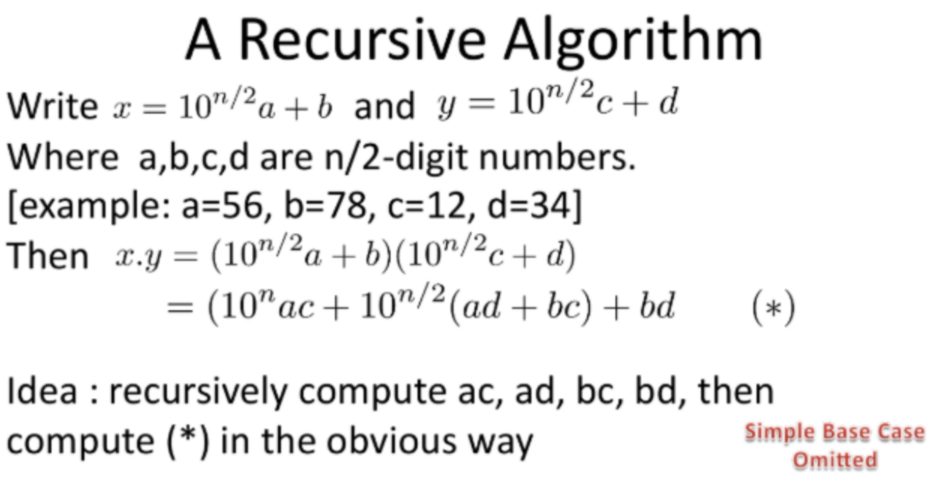
2、 考虑到从命令行获取数据，即要与用户交互，所以要考虑许多不理想的情况，例如：用户输入的是非数字的字母和字符，用户错误输入信息需要提醒更正再重新计算，用户需要计算超大数字的乘法，用户需要计算负数乘法和用户需要计算小数乘法等等。

二、问题思路

1、首先实现整数计算，考虑到乘法的竖式计算，可以考虑利用二维数组来模拟数值的竖式计算，从而得到正确且不会丢失精度的结果；乘法竖式例子如下：

$$\begin{array}{ccccc}
 & & 1 & 2 & 3 \\
 & \times & 1 & 1 & 0 \\
\hline
 & & 0 & 0 & 0 \\
 & 1 & 2 & 3 & \\
1 & 2 & 3 & & \\
\hline
1 & 3 & 5 & 3 & 0
\end{array}$$

但是考虑到在超大数计算的时候，O（n^2）的复杂度优化度太低了，于是参考Karatsuba乘法的思想，采用分治算法，将复杂度降为O($n^{log_2 3}$)，在此基础上，还可以通过增加拆分的个数，从而将复杂度再降低，但是优化的程度没有本质区别，仍然是O（n^logn)的复杂度，因此本次project采用拆分个数为2的算法（Karatsuba)，来对O（n^2)的算法进行优化，Karatsuba原理如下：（针对这个原理利用递归，并自己定义字符串的加减法，就可以实现复杂度的优化)；



A Recursive Algorithm

Write $x = 10^{n/2}a + b$ and $y = 10^{n/2}c + d$

Where a,b,c,d are n/2-digit numbers.

[example: a=56, b=78, c=12, d=34]

Then $x.y = (10^{n/2}a + b)(10^{n/2}c + d)$

$= (10^{n}ac + 10^{n/2}(ad + bc) + bd \qquad (*)$

Idea : recursively compute ac, ad, bc, bd, then compute (*) in the obvious way

Simple Base Case Omitted

在实现这个方法后，竖式乘法的方法被用于计算3位数及3位数以内的乘法计算；

2、其次考虑用户的输入错误，可以分为一下情况：

①首先，当用户错误输入非数字且非字母的非法数字或数学变量名（这里我们认为由数字和字母组成的参数名合法），考虑提示用户输入有误，请重新输入；

②在①基础上确保在用户正确输入后才进行后续程序进行计算操作，错误的输入不会得到计算结果；

③其次考虑用户输入数字和字母同时存在或只存在字母的情况，由于认为该参数名合法，于是会给用户“是否继续计算的提示”，并通过读取用户的输入，即在Pro中，我们设置为用户输入“y”即表示要用此变量名进行操作，并反馈以结果，输入“n”即表示用户希望重新输入数据，其余的输入都会给出错误提示；

④接着考虑用户可能有正负数乘法运算的需求，于是在读取信息时将负号读取后记录，并反映到结果之中；

⑤最后考虑小数乘法，将整数扩展为有理数的计算。

3、 最后，本次Pro使用到的libraries 有：

```
#include <iostream>
#include <cstdint>
#include <vector>
#include <stack>
#include <string>
#include <cstring>
#include <sstream>
```

第二部分：源码

一、主方法体：

```
#include <iostream>
#include <cstdint>
#include <vector>
#include <stack>
#include <string>
#include <sstream>

using namespace std;

string karatsubaMul(string n, string m, int zenum);

string intMul(const int a[], const int b[], int lenA, int lenB, string signDot);

int judgeCal(string c, string d);

bool judgeLawful(string c, string d);

string judSignDot(string c, string d);

bool calculation(string a, string b);

string myPlus(string a, string b, string sign);

int main(int argc, char **argv) {
    string a, b;
    //获取命令行参数
    if (argc > 1) {
        a = argv[1];
        b = argv[2];
    } else {
        //从操作行获取参数
        cout << "Please input two numbers: \n";
        cin >> a;
        cin >> b;
    }

    bool calRes = calculation(a, b);

    if (!calRes) {
        while (!judgeLawful(a, b)) {
            cout << "Please input again: \n";
            cin >> a;
            cin >> b;
            if (calculation(a, b)) {
                exit(0);
            }
        }
        int judCalAns = judgeCal(a, b);
        while (judCalAns == 2) {
            cout << "Please input again: \n";
            cin >> a;
            cin >> b;
            if (calculation(a, b))
                exit(0);
            else
                judCalAns = judgeCal(a, b);
        }
        if (judCalAns == 1) {
            cout << "Calculation successful! \n";
            cout << "The answer is: " << a << " * " << b << endl;
        }
        if (judCalAns == 3) {
            string answer = karatsubaMul(a, b, 0);
            int zeroNum = 0;
            int firLen = answer.length();
            for (int i = 0; i < answer.length(); ++i) {
                if (answer[i] == '0')
                    zeroNum++;
                else break;
            }

            cout << "Calculation successful! \n";
            cout << "The answer is:" << answer.substr(zeroNum, firLen) << endl;
        }
    }
}
```

一、其他方法体：

```
//此方法用于计算两个数相乘，并将结果以字符串的形式返回
string intMul(const int a[], const int b[], int lenA, int lenB, string signDot)
{
    string sig, dt;
    istringstream is(signDot);
    is >> sig >> dt;
    if (lenA == 1 && a[0] == 0)
        return "0";
    if (lenB == 1 && b[0] == 0)
        return "0";
    string res;
    int matrix[lenB][lenA + lenB];
    int result[lenA + lenB];
    for (int i = 0; i < lenB + lenA; i++) {
        result[i] = 0;
    }
    for (int i = 0; i < lenB; ++i) {
        for (int j = 0; j < lenB + lenA; ++j) {
            matrix[i][j] = 0;
        }
    }
    for (int i = 0; i < lenB; i++) {
        for (int j = 0; j < lenA; j++) {
            int temp = matrix[i][j + i] + b[i] * a[j];
            matrix[i][j + i] = temp % 10;
            matrix[i][j + i + 1] = (temp - temp % 10) / 10;
        }
    }
    for (int i = 0; i < lenB + lenA; i++) {
        int tem = result[i];
        for (int j = 0; j < lenB; j++) {
            tem += matrix[j][i];
        }
        result[i] = tem % 10;
        if (i + 1 < lenA + lenB)
            result[i + 1] = (tem - result[i]) / 10;
    }

    stack<int> temp;
    for (int i = 0; i < lenA + lenB; ++i) {
        temp.push(result[i]);
    }
    while (!temp.empty()) {
        res += to_string(temp.top());
        temp.pop();
    }

    if (dt != "0")
        res = res.insert(res.length() - atoi(dt.c_str()), 1, '.');
    long firInt = 0;
    long firDot = 0;
    long lastDot = 0;
    long lastInt = 0;
    for (int i = 0; i < res.length(); ++i) {
        if (res[i] != '0') {
            if (res[i] == '.')
                firDot = i;
            else
                firInt = i;
            break;
        }
    }
    if (firDot != 0)
        res = res.substr(firDot - 1, res.length());
    else if (firInt != 0)
        res = res.substr(firInt, res.length());

    for (int i = 0; i < res.length(); ++i) {
        if (res[i] == '.') {
            lastDot = i;
            continue;
        }
        if (lastDot != 0 && res[i] != '0')
            lastInt = i;
    }
    if (lastInt > lastDot)
        res = res.substr(0, lastInt + 1);
    else if (lastInt < lastDot)
        res = res.substr(0, lastDot);
    if (sig == "-")
        res = sig + res;
    return res;
}
//此方法判断输入是否合法
bool judgeLawful(string c, string d) {

    if ((long) c.length() > INT32_MAX || (long) c.length() < INT32_MIN) {
        cout << "Your input is too long to calculate!\n";
        return false;
    }
    if ((long) d.length() > INT32_MAX || (long) d.length() < INT32_MIN) {
        cout << "Your input is too long to calculate!\n";
        return false;
    }

    for (int i = 0; i < c.length(); ++i) {
        if (c[i] < 48 || (c[i] > 57 && c[i] < 65) || (c[i] > 90 && c[i] < 97) ||
            c[i] > 122) {
            cout << "Your input contains non-numerals or non-letters and it is
invalid!\n";
            return false;
        }
    }
    for (int i = 0; i < d.length(); ++i) {
        if (d[i] < 48 || (d[i] > 57 && d[i] < 65) || (d[i] > 90 && d[i] < 97) ||
            d[i] > 122) {
            cout << "Your input contains non-numerals or non-letters and it is
invalid!\n";
            return false;
        }
    }
    return true;
}
//此方法判断在输入合法的情况下，是否能够相乘，即判断计算的合法性
int judgeCal(string c, string d) {
    int ans = 0;
    string judChar;
    for (int i = 0; i < c.length(); ++i) {
        if (ans == 1 || ans == 2) {
            break;
        }
        if ((c[i] >= 97 && c[i] <= 122) || (c[i] >= 65 && c[i] <= 90)) {
            cout << "Your input contains non-numbers!\n";
            cout << "Do you still want to calculate?\n";
            cout << "Please input y if you want and input n if you not.\n";
            cin >> judChar;
            while (judChar != "y" && judChar != "n") {
                cout << "Your input is wrong! \n";
                cin >> judChar;
            }
            if (judChar == "y") {
                ans = 1;
            } else {
                ans = 2;
            }
        }
    }
    for (int i = 0; i < d.length(); ++i) {
        if ((d[i] >= 97 && d[i] <= 122) || (d[i] >= 65 && d[i] <= 90)) {
            if (ans == 0) {
                cout << "Your input contains non-numbers!\n ";
                cout << "Do you still want to calculate?\n";
                cout << "Please input y if you want and input n if you not.\n";
                cin >> judChar;
                while (judChar != "y" && judChar != "n") {
                    cout << "Your input is wrong!\n";
                    cin >> judChar;
                }
                if (judChar == "y") {
                    ans = 1;
                } else {
                    ans = 2;
                }
            }
        }
        if (ans == 1 || ans == 2)
            break;
    }
    if (ans != 2 && ans != 1)
        ans = 3;
    return ans;
}
//此方法用于正负号和小数点位数的计算
string judSignDot(string c, string d) {
    int dotC = 0;
    int dotD = 0;
    int positionC = 0, positionD = 0;
    string ret;

    if (c[0] != '-' && (c[0] < 48 || c[0] > 57))
        return "false";
    if (d[0] != '-' && (d[0] < 48 || d[0] > 57))
        return "false";

    for (int i = 1; i < c.length(); ++i) {
        if (c[i] == '.') {
            dotC++;
        }
        if (dotC == 1 && c[i] == '.') {
            positionC = c.length() - 1 - i;
            continue;
        }
        if (dotC > 1) {
            return "false";
        }
```

```
            if (c[i] < 48 || c[i] > 57) {
                return "false";
            }
        }
        for (int i = 1; i < d.length(); ++i) {
            if (d[i] == '.') {
                dotD++;
            }
            if (dotD == 1 && d[i] == '.') {
                positionD = d.length() - 1 - i;
                continue;
            }
            if (dotD > 1) {
                return "false";
            }
            if (d[i] < 48 || d[i] > 57) {
                return "false";
            }
        }
        if ((c[0] == '-' && d[0] == '-') || (c[0] != '-' && d[0] != '-')) {
            ret = "+ " + to_string(positionD + positionC);
        } else if ((c[0] != '-' && d[0] == '-') || (c[0] == '-' && d[0] != '-'))
            ret = "- " + to_string(positionD + positionC);

        return ret;
    }
    bool calculation(string a, string b) {
        if (judSignDot(a, b) != "false") {
            int countA = 0;
            int countB = 0;
            if (a[0] == '-')
                countA++;
            for (int i = 0; i < a.length(); ++i) {
                if (a[i] == '.')
                    countA++;
            }
            if (b[0] == '-')
                countB++;
            for (int i = 0; i < b.length(); ++i) {
                if (b[i] == '.')
                    countB++;
            }
            string usA, usB;

            for (int i = a.length() - 1; i >= 0; --i) {
                if (a[a.length() - i - 1] >= 48 && a[a.length() - i - 1] <= 57) {
                    usA += to_string(a[a.length() - i - 1] - '0');
                }
            }

            for (int i = b.length() - 1; i >= 0; --i) {
                if (b[b.length() - i - 1] >= 48 && b[b.length() - i - 1] <= 57) {
                    usB += to_string(b[b.length() - i - 1] - '0');
                }
            }

            int ze1 = 0, ze2 = 0;
            for (int i = 0; i < usA.length(); ++i) {
```

```
            if (ze1 == i && usA[i] == '0')
                ze1++;
            else
                break;
        }
        for (int i = 0; i < usB.length(); ++i) {
            if (ze2 == i && usB[i] == '0')
                ze2++;
            else
                break;
        }
        string res = karatsubaMul(usA, usB, ze1 + ze2);
        string sigDo = judSignDot(a, b);
        string sig, dt;
        istringstream is(sigDo);
        is >> sig >> dt;

        if (dt != "0")
            res = res.insert(res.length() - atoi(dt.c_str()), 1, '.');

        long firInt = 0;
        long firDot = 0;
        long lastDot = 0;
        long lastInt = 0;
        for (int i = 0; i < res.length(); ++i) {
            if (res[i] != '0') {
                if (res[i] == '.')
                    firDot = i;
                else
                    firInt = i;
                break;
            }
        }
        if (firDot != 0)
            res = res.substr(firDot - 1, res.length());
        else if (firInt != 0)
            res = res.substr(firInt, res.length());

        for (int i = 0; i < res.length(); ++i) {
            if (res[i] == '.') {
                lastDot = i;
                continue;
            }
            if (lastDot != 0 && res[i] != '0')
                lastInt = i;
        }
        if (lastInt > lastDot)
            res = res.substr(0, lastInt + 1);
        else if (lastInt < lastDot)
            res = res.substr(0, lastDot);
        if (sig == "-")
            res = sig + res;
        cout << "Calculation successful! \n";
        cout << "The answer is:" << res << endl;
        return true;
    }
    return false;
}
```

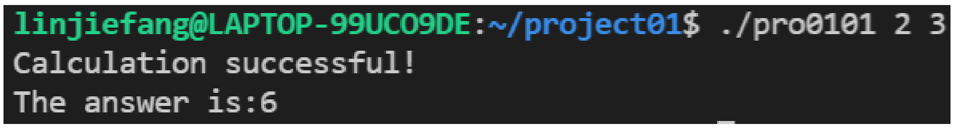
```
//karatsuba算法实现
string karatsubaMul(string n, string m, int zenum) {

    //递归分制的终止条件
    if (n.length() < 4 || m.length() < 4) {
        int numN[n.length()], numM[m.length()];

        int indN = n.length() - 1;
        int indM = m.length() - 1;

        for (int i = n.length() - 1; i >= 0; --i) {
            numN[indN--] = n[n.length() - i - 1] - '0';
        }

        for (int i = m.length() - 1; i >= 0; --i) {
            numM[indM--] = m[m.length() - i - 1] - '0';
        }
        return intMul(numN, numM, n.length(), m.length(), "+ 0");
    }

    // 计算拆分长度
    int sizeN = n.length();
    int sizeM = m.length();
    int half;
    if (sizeN >= sizeM)
        half = (sizeM) / 2;
    else half = (sizeN) / 2;

    //分为a, b, c, d
    string a = n.substr(0, sizeN - half);
    string b = n.substr(sizeN - half, sizeN);
    string c = m.substr(0, sizeM - half);
    string d = m.substr(sizeM - half, sizeM);

    //使用递归
    string p2 = karatsubaMul(a, c, 0);
    string p0 = karatsubaMul(b, d, 0);
    string p1 = myPlus(karatsubaMul(myPlus(a, b, "+"), myPlus(c, d, "+"), 0),
myPlus(p0, p2, "+"), "-");

    for (int i = 0; i < half * 2; ++i) {
        p2 += "0";
    }
    for (int i = 0; i < half; ++i) {
        p1 += "0";
    }
    string mypl = myPlus(myPlus(p1, p2, "+"), p0, "+");

    for (int i = 0; i < zenum; i++) {
        mypl = "0" + mypl;
    }
    return mypl;
}
//辅助方法，实现字符串加减法
string myPlus(string a, string b, string sign) {
    int maxLen = 0;
    if (a.length() >= b.length())
        maxLen = a.length();
```

```cpp
    else
        maxLen = b.length();

    int inA[maxLen];
    int inB[maxLen];
    int ina = a.length() - 1;
    int inb = b.length() - 1;
    for (int i = maxLen - 1; i >= 0; --i) {
        if (ina >= 0) {
            inA[i] = a[ina--] - '0';
        } else
            inA[i] = 0;

        if (inb >= 0) {
            inB[i] = b[inb--] - '0';
        } else
            inB[i] = 0;
    }

    ina = maxLen - 1;
    inb = maxLen - 1;
    int rs[maxLen + 1];
    for (int i = 0; i < maxLen + 1; ++i) {
        rs[i] = 0;
    }
    if (sign == "+") {
        for (int i = maxLen; i >= 1; --i) {
            int temp = rs[i] + inA[ina--] + inB[inb--];
            rs[i] = temp % 10;
            rs[i - 1] = temp / 10;
        }
    } else {
        for (int i = maxLen; i >= 1; --i) {
            if (inA[ina--] + rs[i] >= inB[inb--])
                rs[i] += inA[ina + 1] - inB[inb + 1];
            else {
                rs[i - 1]--;
                rs[i] += inA[ina + 1] + 10 - inB[inb + 1];
            }
        }

    }
    string reu;
    int numb = 0;
    for (int i = 0; i < maxLen + 1; ++i) {
        if (i == numb && rs[i] == 0 && numb != maxLen) {
            numb++;
            continue;
        }

        reu += to_string(rs[i]);
    }
    return reu;
}
```

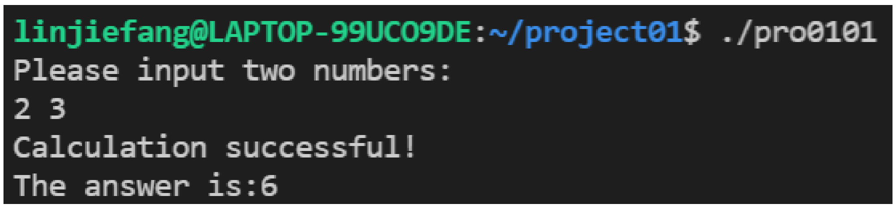
第三部分：结果展示

1、 正常的整数计算（包括正负数、小数计算和大数计算），结果如下：

① 两种传参形式：

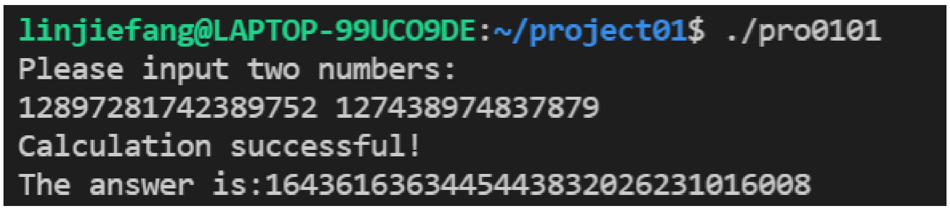


```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101 2 3
Calculation successful!
The answer is:6
```

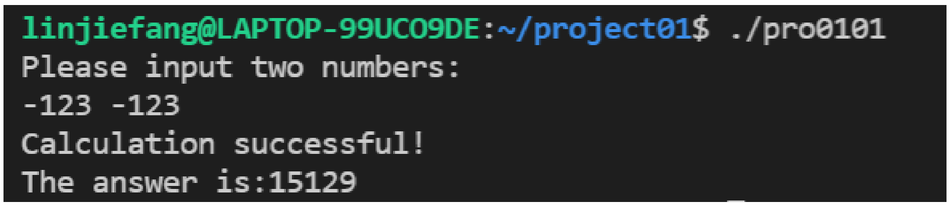


```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
2 3
Calculation successful!
The answer is:6
```

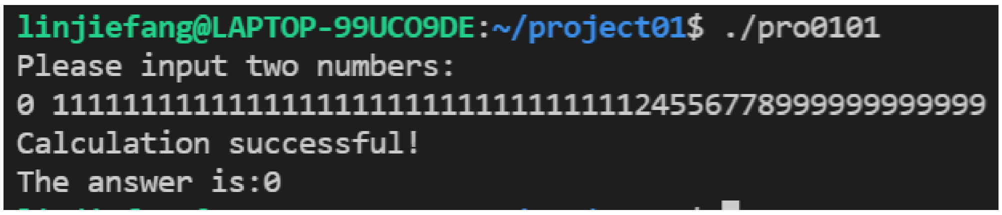
② 较大整数（包含正负数）相乘和超大整数相乘（可以计算小于但接近INT_MAX/2 长度的数字相乘，长度局限于数组最大长度一般为INT_MAX）：



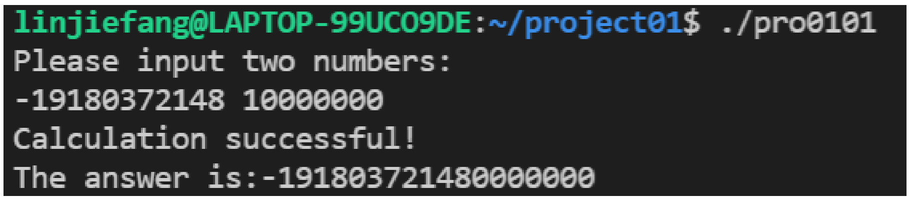
```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
128972817423897552 127438974837879
Calculation successful!
The answer is:16436163634454438320262310160008
```



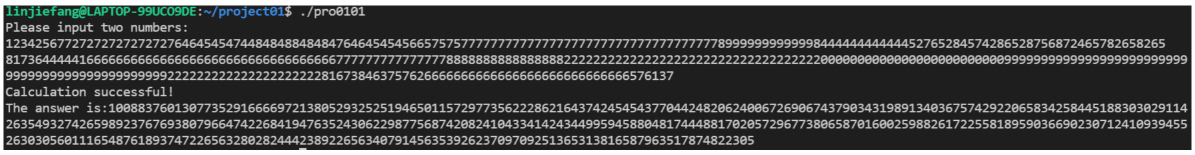
```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
-123 -123
Calculation successful!
The answer is:15129
```



```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
0 111111111111111111111111111111111111245567789999999999999
Calculation successful!
The answer is:0
```

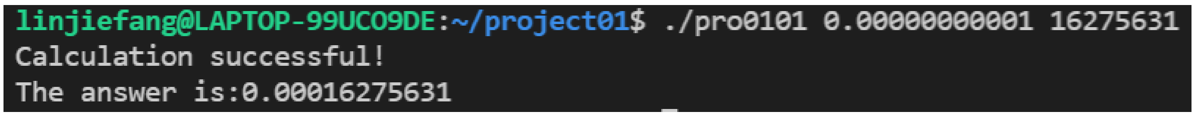


```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
-19180372148 10000000
Calculation successful!
The answer is:-191803721480000000
```

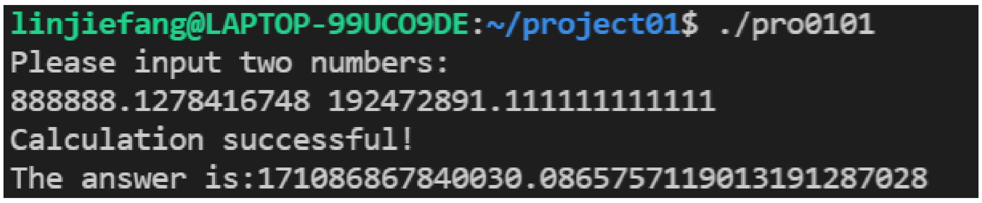


```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
12342567727272727272727276464545474484848848484764645454566575757777777777777777777777777777777777899999999999999844444444444452765284574286528756872465782658265
81736444441666666666666666666666666666666666667777777777777777888888888888888822222222222222222222222222222222222222200000000000000000000000000000099999999999999999999999999
99999999999999999999922222222222222222222222281673846375762666666666666666666666666666576137
Calculation successful!
The answer is:100883760130773529166669721380529325251946501157297735622286216437424545437704424820624006726906743790343198913403675742922065834258445188303029114
26354932742659892376769380796647422684194763524306229877568742082410433414243449959458804817444881702057296773806587016002598826172255818959036690230712410939455
263030560111654876189374722656328028244423892265634079145635392623709709251365313816587963517874822305
```

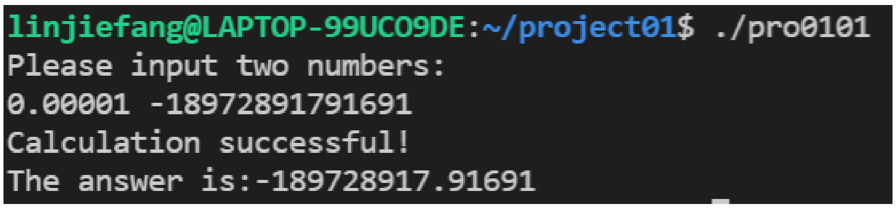
③ 小数相乘（考虑保留有效数字，即当结果中有无效0，都将其省略，例如1000*0.001=1.000，省略后面三个0）：



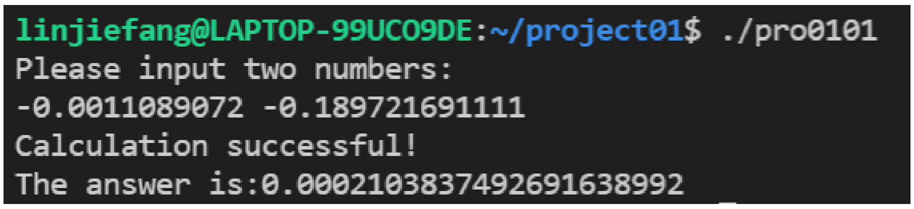
```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101 0.00000000001 16275631
Calculation successful!
The answer is:0.00016275631
```



```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
888888.1278416748 192472891.111111111111
Calculation successful!
The answer is:171086867840030.0865757119013191287028
```



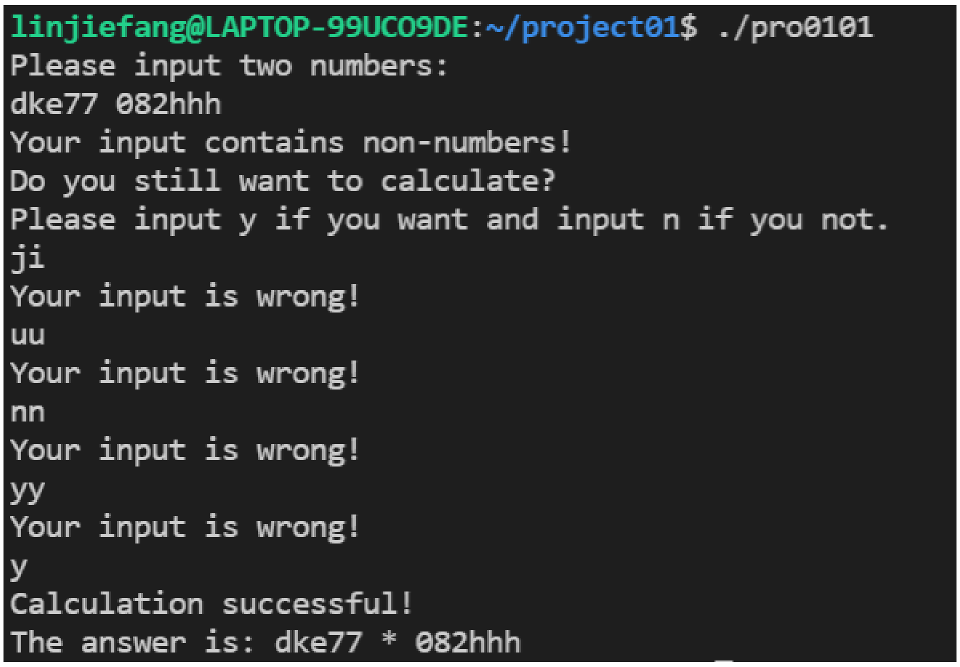
```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
0.00001 -18972891791691
Calculation successful!
The answer is:-189728917.91691
```



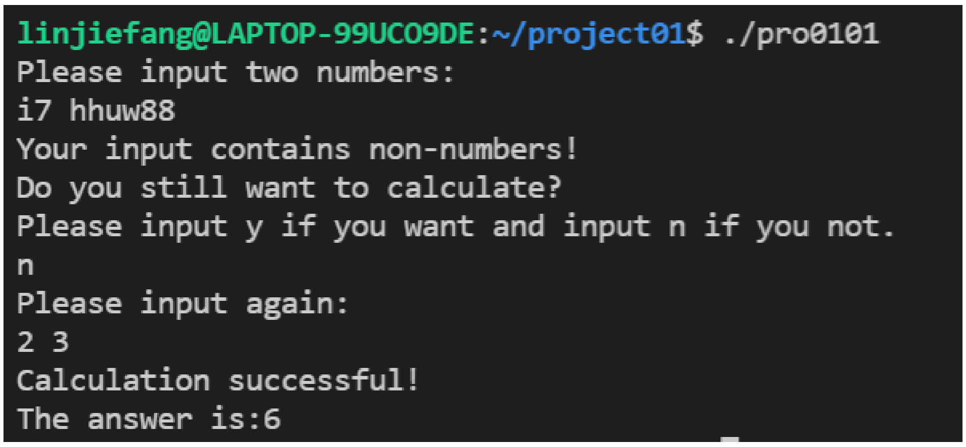
```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
-0.0011089072 -0.189721691111
Calculation successful!
The answer is:0.0002103837492691638992
```

2、 字母及非法式子

① 字母相乘（仅含字母或字母和数字，认为该变量名合法）：

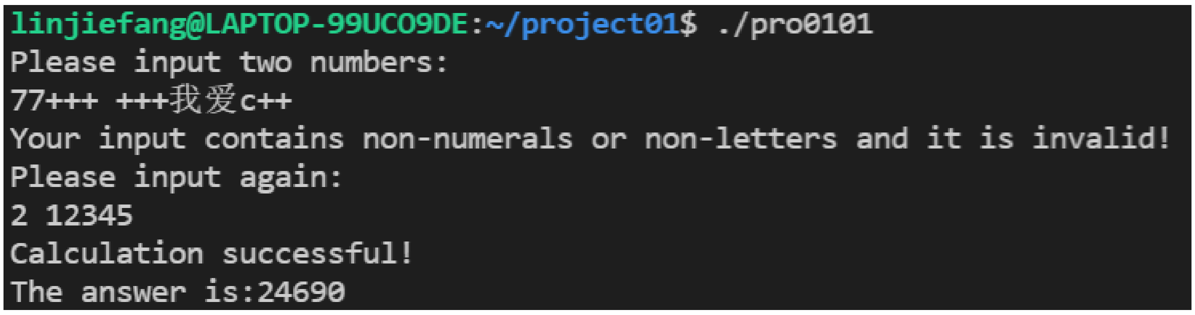


```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
dke77 082hhh
Your input contains non-numbers!
Do you still want to calculate?
Please input y if you want and input n if you not.
ji
Your input is wrong!
uu
Your input is wrong!
nn
Your input is wrong!
yy
Your input is wrong!
y
Calculation successful!
The answer is: dke77 * 082hhh
```

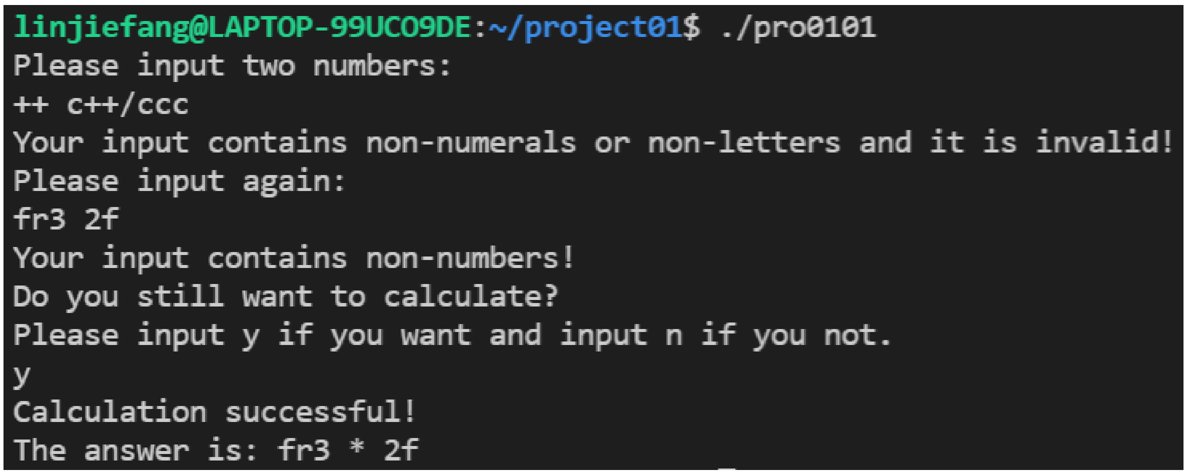


```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
i7 hhuw88
Your input contains non-numbers!
Do you still want to calculate?
Please input y if you want and input n if you not.
n
Please input again:
2 3
Calculation successful!
The answer is:6
```

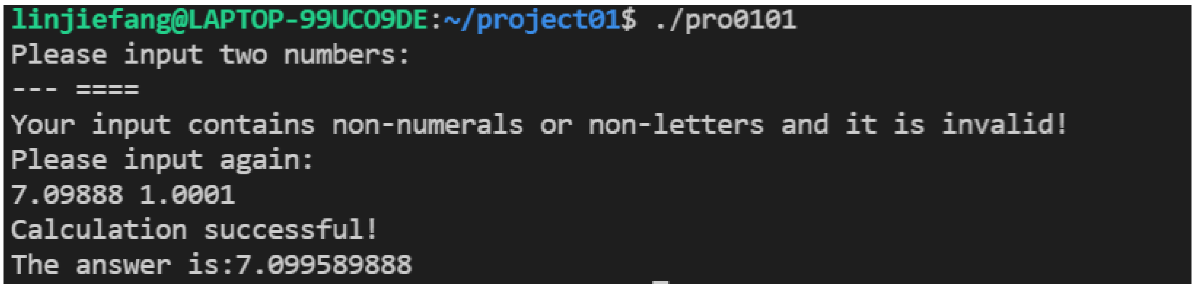
② 不合法输入（重新输入后可以进行以上所有计算）：



```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
77+++ +++我爱c++
Your input contains non-numerals or non-letters and it is invalid!
Please input again:
2 12345
Calculation successful!
The answer is:24690
```



```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
++ c++/ccc
Your input contains non-numerals or non-letters and it is invalid!
Please input again:
fr3 2f
Your input contains non-numbers!
Do you still want to calculate?
Please input y if you want and input n if you not.
y
Calculation successful!
The answer is: fr3 * 2f
```



```
linjiefang@LAPTOP-99UCO9DE:~/project01$ ./pro0101
Please input two numbers:
--- ====
Your input contains non-numerals or non-letters and it is invalid!
Please input again:
7.09888 1.0001
Calculation successful!
The answer is:7.099589888
```

第四部分：难点和方案

1、在实现乘法计算器时，考虑超过INT_MAX 或者 LONG_MAX ，所以利用竖式计算的思想和二维数组的手段，自己完成了乘法方法体的代码；

在竖式计算思想的启发下，通过查找资料，发现复杂度可以进一步优化，即利用Karatsuba算法，将复杂度降低到O($n^{log_2 3}$)，但是此算法是基于一个大数考虑，即其考虑范围仍然在LONG_MAX的范围之内，因此，需要自己实现字符串的加减法是解决这个问题的关键，因此自己实现了一个方法用来计算字符串的加减法。还可以考虑通过增加拆分，让复杂度降到O（n^log3(5))等等，但是优化空间较为局限；

2、考虑用户不按常规出牌，输入的内容有可能合法有可能非法，只要两个参数中存在一个不满足有理数的特性，就要及时给予用户反馈，因此要考虑多种情况，写多种函数来判断输入的合理性，该过程较复杂，考虑到非字符的出现有可能是“-”（负号）或者“.”（小数点），因此并非所有非字符的内容一旦出现即不合法，这给判断条件带来了很大的不方便，以及小数点的加入，和删去多余的0（包括10.000 以及00100.0001000 等类似的多余），都是不小的过程量，这些判断都为了防止程序出错，无法正常反馈答案；

3、在实现乘法计算器后，考虑复杂度的优化，利用普通的乘法竖式计算的思想所得到的计算方式是O（n^2），于是查找资料中发现，大数相乘目前最优化的算法是快速傅里叶变换，它通过引进复数和矩阵计算的思想，可以将复杂度优化到O（nlogn），但是由于此公式的数学思想复杂，短时间内没法很好的理解，希望下次有机会可以用上快速傅里叶变换（FFT)。