CS205 C/C++ Programming - Project Report 1

Name:张闻城

SID:12010324

Part 1 - Analysis

1. **Read input from command line arguments:**

   ```
   int main(int argc, char** argv);
   ```

   argc means the number of arguments. **argv[0]** is your program's name. You can get the two integers by **argv[1]** and **argv[2]** in the form of char array.

2. **Implement of multiplication between big integers:**

   The requirement is to design a calculator that can multiply two integers. This is actually a quite easy problem. What makes it difficult is that there are no data types that can store unlimited data. So when the multipliers are extremely large, it will cause data overflow. So I solve the problem in this way:

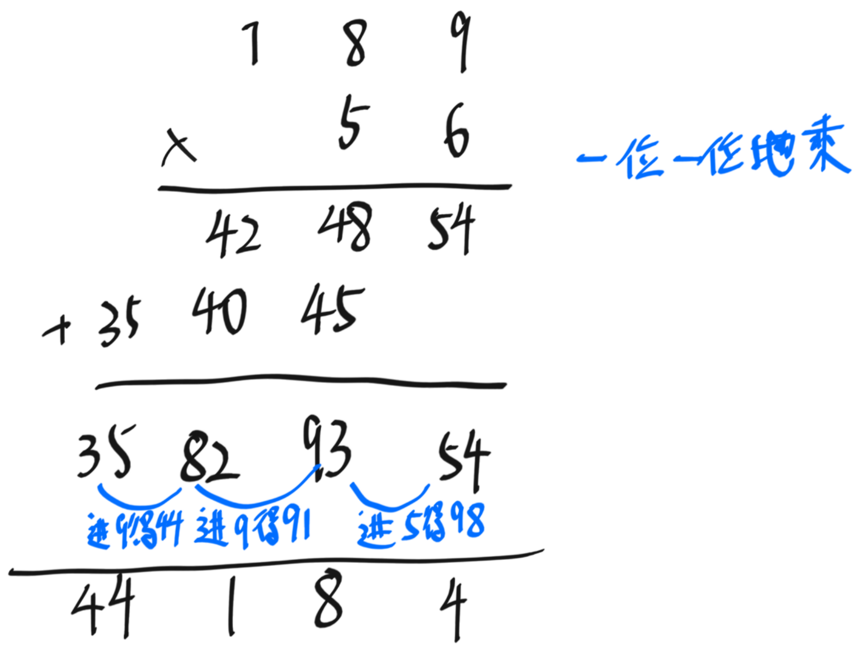
   There are two integers, **a** and **b** (assuming that **a** is lager than **b**)

   - Multiply the digit in ones place of the smaller integer(assuming the smaller one is **b**) with

     every digit of **a**, and put every result into an vector **res**, seperately.
   - Then multiply the digit in tens place of **b** with every digit of **a**, and add every result to the corrresponding position in the vector **res**(digit in ones place adds to digit ones place, digit in

     tens place adds to digit in tens place).
   - Similarly, repeat the step with digits in hundreds place, thousands place, ... of **b**
   - Finally, if the number stored in **res** is more than 9, then you need to divide the number with 10 and get the remaider, and then add the remainder to the number in the next place. Repeat it one place by one place. Then you can get the result.

- **eg.** 789 * 56 = ?



3. **Addition between two big integers:**

   This is quite similar with the mulplication but without the step of multiplication. But it can not complete the addition between positive integer and negative integer so far.

4. **Multiplication between real integers(not just positive integers):**

   To handle the sign problem of the result, I remove the negative sign of two integers if has firstly. Then do the simple multiplication of positive integers. At last, I insert the negative sign at the beginning of the result if needed.

5. **Remove the zero at the head of two integers automatically:**

   If user input an integers like **"00123"** and **"0000321"**, the program will automatically remove the unneccessary zero at the head of the integer.

6. **Idempotent operation:**

   Complete this task by invoking **mul()** several times. However, it can't do the idempotent operations whose index is negative(such as 2^{-2}).

7. **What if I want to exit?**

   If you input "quit", then the **main()** function will invoke the **exit()** function to finiish the program.

Part 2 - Code

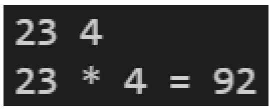
```
#include <iostream>
#include <string>
#include <cstring>
#include <vector>
#include <algorithm>
using namespace std;

string mul(string str1, string str2);

string add(string str1, string str2);

```

```cpp
string power(string base, string index);

void mulPrint(string str1, string str2);

void addPrint(string str1, string str2);

void powPrint(string base, string indexStr);

string removeZero(string s);

int main(int argc, char **argv)
{
    bool flag = false;

    string str1;
    string str2;
    if (argc > 1)
    {
        str1 = argv[1];
        str2 = argv[2];
        goto FLAG;
    }

    while (true)
    {
        cout << "Please input two integers" << endl;
        cin >> str1;
        if (str1.compare("quit") == 0)
        {
            exit(1000);
        }
        cin >> str2;

    FLAG:

        bool isNum = true;
        for (int i = 1; i < str1.length(); i++)
        {
            if (!isdigit(str1[i]) && (str1[0] == '-' || isdigit(str1[0])))
            {
                isNum = false;
                break;
            }
        }

        //检查输入是否合法
        for (int i = 1; i < str2.length(); i++)
        {
            if (!isdigit(str2[i]) && (str2[0] == '-' || isdigit(str2[0])))
            {
                isNum = false;
                break;
            }
        }
        if (!isNum)
        {
            cout << "Invalid input. Try again." << endl;
            continue;
        }

        //去掉前面无意义的0
        str1 = removeZero(str1);
        str2 = removeZero(str2);

        //进行乘，加，指数运算并打印出结果(加法只支持正整数加法)
        mulPrint(str1, str2);
        addPrint(str1, str2);
        powPrint(str1, str2);
    }
}

string mul(string str1, string str2)
{
    //先除去头部的负号(如果有的话)，以便于后续运算
    if (str1[0] == '-')
    {
        str1 = str1.erase(0, 1);
    }
    if (str2[0] == '-')
    {
        str2 = str2.erase(0, 1);
    }

    vector<int> res(str1.length() + str2.length() + 2, 0);
    //先按位一位一位的乘，并将对应位上的数求和并存储到res中(倒着存储)
    for (int i = 0; i < str1.length(); i++)
    {
        for (int j = 0; j < str2.length(); j++)
        {
            res[i + j] += (str2[str2.length() - j - 1] - '0') *
(str1[str1.length() - i - 1] - '0');
        }
    }
    //进位
    for (int i = 0; i < str1.length() + str2.length(); i++)
    {
        int digit = res[i] % 10;
        int carry = res[i] / 10;
        res[i] = digit;
        res[i + 1] += carry;
    }

    bool null = false;
    string s = "";
    //将结果拼接在一起
    for (int i = str1.length() + str2.length() - 1; i >= 0; i--)
    {
        if (res[i] != 0 && res[i + 1] == 0)
        {
            null = true;
        }
        if (null)
        {
            s.append(to_string(res[i]));
        }
    }
    return s;
}

void mulPrint(string str1, string str2)
{
    bool minus_num1 = false;
    bool minus_num2 = false;
    if (str1[0] == '-')
    {
        minus_num1 = true;
    }
    if (str2[0] == '-')
    {
        minus_num2 = true;
    }
    cout << str1
         << " * "
         << str2
         << " = "
         << ((minus_num1 ^ minus_num2) ? "-" : "");
    cout << (str1.length() > str2.length() ? mul(str2, str1) : mul(str1, str2)) << endl;
}

string add(string str1, string str2)
{
    int length = (str1.length() > str2.length() ? str1.length() : str2.length());
    reverse(str1.begin(), str1.end());
    reverse(str2.begin(), str2.end());
    vector<int> res(length + 2, 0);
    for (int i = 0; i < str2.length(); i++)
    {
        if (i >= str1.length())
        {
            res[i] = str2[i] - '0';
        }
        else
        {
            res[i] = (str1[i] - '0') + (str2[i] - '0');
        }
    }

    for (int i = 0; i < res.size(); i++)
    {
        int digit = res[i] % 10;
        int carry = res[i] / 10;
        res[i] = digit;
        res[i + 1] += carry;
    }

    bool null = false;
    string s = "";
    for (int i = res.size() - 1; i >= 0; i--)
    {
        if (res[i] != 0 && res[i + 1] == 0)
        {
            null = true;
        }
        if (null)
        {
            s.append(to_string(res[i]));
        }
    }
    return s;
}

void addPrint(string str1, string str2)
{
    cout << str1 << " + " << str2
         << " = "
         << (str1.length() > str2.length() ? add(str2, str1) : add(str1, str2))
         << endl;
}

string power(string base, string indexStr)
{
    string res = base;
    int index = stoi(indexStr);
    for (int i = 0; i < index - 1; i++)
    {
        res = mul(res, base);
    }
    return res;
}

void powPrint(string base, string indexStr)
{
    bool minus = false;
    cout << base << " ^ " << indexStr << " = ";
    if (base[0] == '-')
    {
        base = base.erase(0, 1);
        minus = true;
    }
    int index = stoi(indexStr);
    if (minus && index % 2 != 0)
    {
        cout << "-";
    }
    cout << power(base, indexStr) << endl;
}

string removeZero(string s)
{
    int zeroNum = 0;
    for (int i = 0; i < s.length(); i++)
    {
        if (s[i] == '-' && i == 0)
        {
            continue;
        }
        if (s[i] == '0')
        {
            zeroNum++;
        }
        if (s[i] != '0')
        {
            break;
        }
    }
    return s[0] == '-' ? s.erase(1, zeroNum) : s.erase(0, zeroNum);
}
```

Part 3 - Result & Verification

Test case #1:

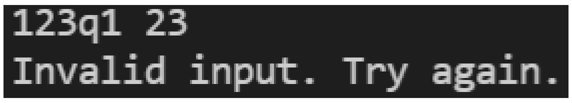
```
Input:23 4
Output:23 * 4 = 92
```



```
23 4
23 * 4 = 92
```

Test case #2:

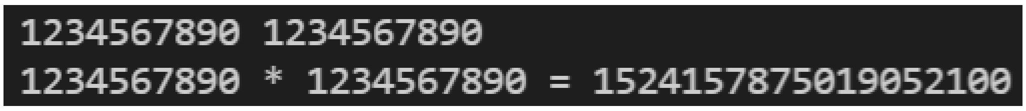
```
Input:123q1 23
Output:Invalid input. Try again
```



```
123q1 23
Invalid input. Try again.
```

Test case #3:

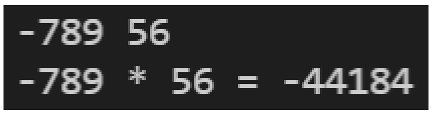
```
Input case #3:1234567890 1234567890
Output:1234567890 * 1234567890 = 1524157875019052100
```



```
1234567890 1234567890
1234567890 * 1234567890 = 1524157875019052100
```

Test case #4:

```
Input:-789 56
Output:-789 * 56 = -44184
```



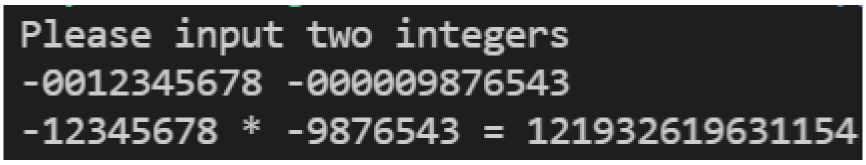
```
-789 56
-789 * 56 = -44184
```

Test case #5:

```
Input:-789 -56
Output:-789 * -56 = 44184
```

Test case #6:

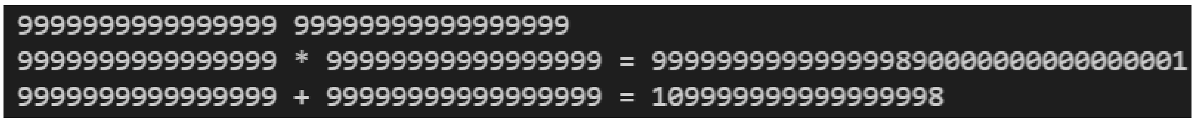
```
Input:-0012345678 -000009876543
OUtput:-12345678 * -9876543 = 121932619631154
```



```
Please input two integers
-0012345678 -000009876543
-12345678 * -9876543 = 121932619631154
```

Test case #7:

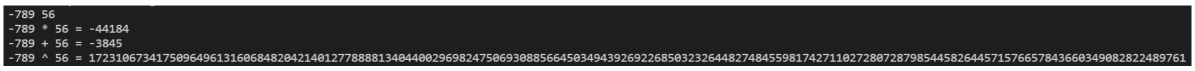
```
Input:9999999999999999 99999999999999999
Output:9999999999999999 + 99999999999999999 = 109999999999999998
```



```
9999999999999999 99999999999999999
9999999999999999 * 99999999999999999 = 999999999999999890000000000000001
9999999999999999 + 99999999999999999 = 109999999999999998
```

Test case #8:

```
Input:-789 56
Output:-789 ^ 56 =
17231067341750964961316068482042140127788881340440029698247506930885664503494
39269226850323264482748455981742711027280728798544582644571576657843660349082
822489761
```



```
-789 56
-789 * 56 = -44184
-789 + 56 = -3845
-789 ^ 56 = 1723106734175096496131606848204214012778888134044002969824750693088566450349439269226850323264482748455981742711027280728798544582644571576657843660349082822489761
```

Test case #9:

```
Input:quit
Output:(The program exits)
```

*Note: The results above are all correct after checking in **Mathematica**.*