**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.

1. 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

1. If you input some big integers as follows

./mul

Please input two integers

1234567890 1234567890

What will happen? Please describe some possible solutions.

1. 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.)

1. If you only implement requirement 1, the upper boundary of your score is 80. For a better score, you should implement the rest requirements. You 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.
2. 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**:巫晓, **SID**: 11912803

## Part 1. Source Code

#include <iostream>

#include <cstdio>

#include <cstring>

using namespace std;

bool checkInput(string s, int n);

int a[100001] = { 0 };

int b[100001] = { 0 };

int c[200003] = { 0 };

int main()

{

int flag = 0; // 0+ 1-

string s1, s2;

printf("Please input two integers\n");

cin >> s1 >> s2;

if (s1[0] == '-')

{

flag = flag ^ 1;

s1.erase(0, 1);

}

else if (s1[0] == '+')

{

s1.erase(0, 1);

}

if (s2[0] == '-')

{

flag = flag ^ 1;

s2.erase(0, 1);

}

else if (s2[0] == '+')

{

s2.erase(0, 1);

}

int m = s1.length() + s2.length() - 1;

if (m > 200000)

{

printf("Too big to calculate!\n");

return 0;

}

if (checkInput(s1, 1) && checkInput(s2, 2))

{

for (int i = 0; i < s1.length(); i++)

{

a[i] = s1[s1.length() - 1 - i] - '0';

}

for (int i = 0; i < s2.length(); i++)

{

b[i] = s2[s2.length() - 1 - i] - '0';

}

for (int i = 0; i < s1.length(); i++)

{

for (int j = 0; j < s2.length(); j++)

{

c[i + j] += a[i] \* b[j];

c[i + j + 1] += c[i + j] / 10;

c[i + j] = c[i + j] % 10;

}

}

for (int i = 0; i < m; i++)

{

c[i + 1] += c[i] / 10;

c[i] = c[i] % 10;

}

if (flag == 1)

{

printf("-");

}

for (int i = m; i >= 0; i--)

{

printf("%d", c[i]);

}

}

return 0;

}

bool checkInput(string s, int n)

{

for (int i = 0; i < s.length(); i++)

{

if (('0' <= s[i])&&(s[i] <= '9'))

{

continue;

}

else

{

printf("The s%d isn't a integer, please check it.\n", n);

return false;

}

}

return true;

}

## 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:

-9999 +9999

-99980001

Screen-short for case #1:

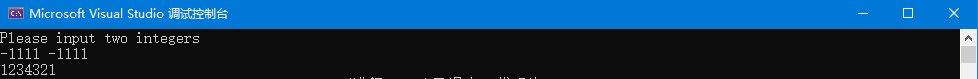


Test case #2:

-1111 -1111

1234321

Screen-short for case #2:



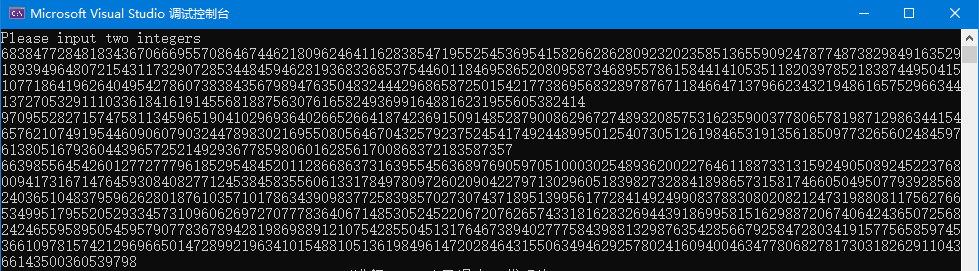
Test case #3:

A=6838477284818343670666955708646744621809624641162838547195525453695415826628628092320235851365590924787748738298491635291893949648072154311732907285344845946281936833685375446011846958652080958734689557861584414105351182039785218387449504151077186419626404954278607383843567989476350483244429686587250154217738695683289787671184664713796623432194861657529663441372705329111033618416191455681887563076165824936991648816231955605382414

B=9709552827157475811345965190410296936402665266418742369150914852879008629672748932085753162359003778065781987129863441546576210749195446090607903244789830216955080564670432579237524541749244899501254073051261984653191356185097732656024845976138051679360443965725214929367785980601628561700868372183587357

Result= 66398556454260127727779618529548452011286686373163955456368976905970510003025489362002276461188733131592490508924522376800941731671476459308408277124538458355606133178497809726020904227971302960518398273288418986573158174660504950779392856824036510483795962628018761035710178634390983772583985702730743718951399561772841492499083788308020821247319880811756276653499517955205293345731096062697270777836406714853052452206720762657433181628326944391869958151629887206740642436507256824246559589505459579077836789428198698891210754285504513176467389402777584398813298763542856679258472803419157756585974536610978157421296966501472899219634101548810513619849614720284643155063494629257802416094004634778068278173031826291104366143500360539798

Screen-short for case #3:



## Part 3. Difficulties & Solutions, or others

思路：1.两数相乘，首先想到的是输入合不合法，于是我使用字符串读入数据并且加入了checkInput语句来检测。

2.考虑到“正负号”，我现在check前检查并截取了，用flag存储最终符号，注意一点，“+-”在数字中间是违法的。

3.当检测到不合法输入时会提醒。

4.当输入合法时，我们考虑进行计算，这里使用高精度乘法。

5.直接使用字符串计算会有比较多的逻辑问题，所以我舍去了过大的case（200000以上）并进行提醒需要更改代码。

6.使用高精度数组时，要注意结果的位数是[n1+n2-1,n1+n2]（n为对应数的位数）