

CS205 C/ C++ Program Design

Assignment 1

Name: 方唯可 **SID:** 22010055 交换生大二

Part 1. Source Code

```
//mul.cpp
//Name: Weike Fang (22010055)
// Credit to www.cnblogs.com/tonglingliangyong/p/3908463.html for
checking c-input datatype
#include <iostream>
#include <iomanip>
#include <string>
#include <string.h>
#include <algorithm>

using namespace std;

int main()
{
    cout << "Integer Multiplier!" << endl;
    cout << "Please input two integers separated with a space." <<
endl;
    long long int a, b;
    while (1)
    {
        cin >> a >> b;
        if (cin.fail()) //or !cin
        {
            cout << "The input should be 2 integers separated by a
space. Enter Again" << endl;
            cin.clear();
            cin.sync();
        }
        else
        {
            long long int product = a * b;
            cout << "How would you like your answer be presented?
Decimal or Scientific." << endl;
            string format;
```

```

while (1)
{
    cin >> format;
    transform(format.begin(), format.end(),
    format.begin(), ::tolower); //convert to lowercase
    string d = "decimal";
    string s = "scientific";
    if (format.compare(d) == 0)
    {
        cout << "The product of " << a
        << " and " << b << " is ";
        cout << dec << product << endl;
        break;
    }
    else if (format.compare(s) == 0)
    {
        cout << "The product of " << a << " and "
        << b << " is ";
        cout << scientific << double(product) << endl;
        break;
    }
    else
    {
        cout << "The input should be either Scientific or
        Decimal. Enter Again" << endl;
        cin.clear();
        cin.sync();
    }
}

break;
}
}
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.

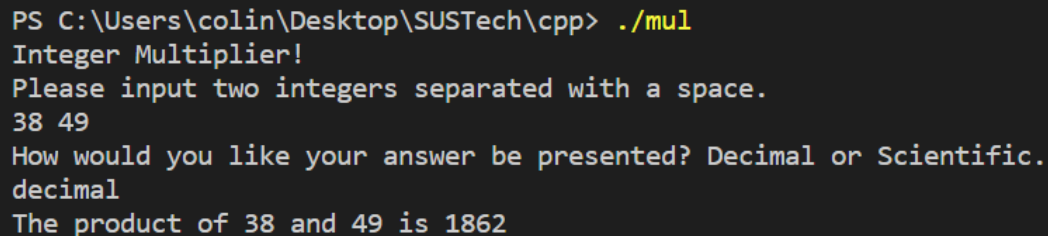
Test case #1:

Multiply two small integers, and present the answer in decimal.

Integer Multiplier!

```
Please input two integers separated with a space.  
38 49  
How would you like your answer be presented? Decimal or Scientific.  
decimal  
The product of 38 and 49 is 1862
```

Screen-short for case #1:



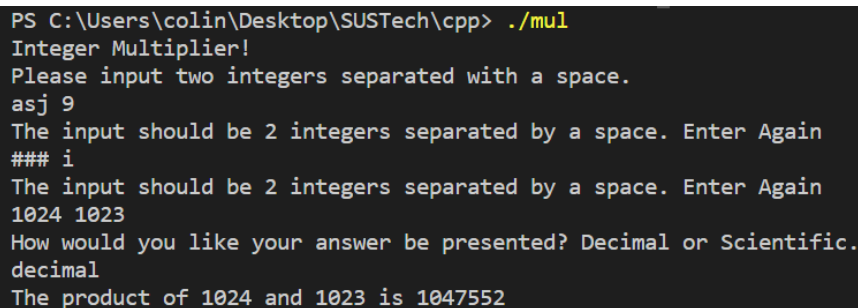
```
PS C:\Users\colin\Desktop\SUSTech\cpp> ./mul  
Integer Multiplier!  
Please input two integers separated with a space.  
38 49  
How would you like your answer be presented? Decimal or Scientific.  
decimal  
The product of 38 and 49 is 1862
```

Test case #2:

When the user input other characters, such as letters, the program will prompt another line to ask for another valid input.

```
Integer Multiplier!  
Please input two integers separated with a space.  
asj 9  
The input should be 2 integers separated by a space. Enter Again  
### i  
The input should be 2 integers separated by a space. Enter Again  
1024 1023  
How would you like your answer be presented? Decimal or Scientific.  
decimal  
The product of 1024 and 1023 is 1047552
```

Screen-short for case #2:



```
PS C:\Users\colin\Desktop\SUSTech\cpp> ./mul  
Integer Multiplier!  
Please input two integers separated with a space.  
asj 9  
The input should be 2 integers separated by a space. Enter Again  
### i  
The input should be 2 integers separated by a space. Enter Again  
1024 1023  
How would you like your answer be presented? Decimal or Scientific.  
decimal  
The product of 1024 and 1023 is 1047552
```

Test case #3:

Multiply two big integers (~9-10digits), the program will have the correct result.

```
Integer Multiplier!  
Please input two integers separated with a space.  
1234567890 1234567890  
How would you like your answer be presented? Decimal or Scientific.  
Decimal  
The product of 1234567890 and 1234567890 is 1524157875019052100
```

Screen-short for case #3:

```

PS C:\Users\colin\Desktop\SUSTech\cpp> ./mul
Integer Multiplier!
Please input two integers separated with a space.
1234567890 1234567890
How would you like your answer be presented? Decimal or Scientific.
Decimal
The product of 1234567890 and 1234567890 is 1524157875019052100

```

Test case #4:

Multiply big number and present the answer with scientific notation. The program will specify two options, Decimal or Scientific. If not, the program will ask for valid input. Upper- or lower- case doesn't matter.

```

Integer Multiplier!
Please input two integers separated with a space.
123456789 1234567890
How would you like your answer be presented? Decimal or Scientific.
b
The input should be either Scientific or Decimal. Enter Again
ScienTiFic
The product of 123456789 and 1234567890 is 1.524158e+17

```

Screen-short for case #4:

```

PS C:\Users\colin\Desktop\SUSTech\cpp> ./mul
Integer Multiplier!
Please input two integers separated with a space.
123456789 1234567890
How would you like your answer be presented? Decimal or Scientific.
b
The input should be either Scientific or Decimal. Enter Again
ScienTiFic
The product of 123456789 and 1234567890 is 1.524158e+17

```

Part 3. Difficulties & Solutions, or others

I completed the first task rather quickly, as it only requires extracting the input of two (small) integers and then output the product of them. The first difficulty that I encountered was how to deal with the wrong input data type. The function that I wished to have was to ask again if the wrong input type is extracted. After a decent amount of research online, I found a blog that catered to my need (<https://www.cnblogs.com/tonglingliangyong/p/3908463.html>). It discusses the use of `!cin`, which flags the error when the input data type does not match our pre-defined variables. With the simple if-else statement, the program can deliver different outputs in the 2 cases. Also, the blog teaches how to use `cin.clear()` and `cin.sync()` in a while loop, which helps us reset the error marking and clear the input stream, respectively. The mechanism is straightforward: if the received input is correct, then the program will break the loop and continue the operation; if not, it will clear the error mark and input stream and prompt another input until it's valid.

The next difficulty that I encountered was how to multiply two big integers without exceeding the boundary. Apparently, the standard integer type `int` was not enough, which is only 4 bytes (=32

bits) and cannot exceed the limit of $2^{31}-1$ if signed *int* is used. After carefully going through the documentation (en.cppreference.com/w/cpp/language/types), I found the *long long* integer type allows for 64 bits of storage, whose limit is about $2^{63}-1$ with sign included. Since we would like to present the product of two integers, the maximal integer allowed would be the square root of 2^{63} (~9- to 10-digit number). Such a boundary is good enough for the issue raised in task 3, the product of 1234567890 with itself. However, if larger integers are inputted, the product will exceed the boundary, and thus the answer will be invalid. I have also looked for a solution for quite a long time but found that most of the tutorials include the use of character lists, with extensive use of pointer and loop, which were unfamiliar to me. I will return to this assignment and try to refine my code when I figure them out.

Another function that I wished to have was to present the result of multiplication in scientific notation. When the output integer has many digits, scientific notation gives a sense of how big the number is so that the user doesn't have to count the number of digits manually. However, Professor Yu didn't cover the topic in the first class, so it was difficult for me at that time. After skimming the C++ documentation, I found we can directly specify the `cout` format/base (in my case, either decimal or scientific).

While I was writing the code asking for "Decimal or Scientific," I found that the user may input a string/word in upper- or lower- case. To deal with such an issue, I went through the C++ algorithm library (header) and found the `transform` function that can convert a string to its lower case. Such a method helps me avoid input issues related to small/capital letters.

Some problems remain in my code, for which I don't have a concrete solution now. In addition to the problem of huge integers, there is another issue about floating numbers. If the user enters a floating-point number as the second integer, the program may not be able to detect the wrong input type. For example, if the user input `12 12.3` in the command line, the program will misinterpret the information as `12 12` and thus output `144`. The `.3` will remain in the input stream. I will keep working on the problem and hopefully will solve it in my next assignment.