CS205 C/ C++ Programming – Project 1

Name: 凌硕（Ling Shuo）
SID：11912409

Part 1 – Analysis

The problem is computing the product of two numbers. But consider that the numbers can be large numbers (which beyond the range of int, even of long long), decimals and numbers given by scientific counting method, we can solve the problem by the following steps:

Step 1 – Computing the product of two positive large integers
In this step, we compute the product of two positive integers which may beyond the range of int, or even the range of long long. Suppose the two numbers are *a* and *b*. We store *a* in the array (of int) from the last bit forward, and do the same thing to *b*, the result is *a*[ ] and *b*[ ]. For example, if *a*=123, we can get the array *a*[3] with *a*[0]=3, *a*[1]=2 and *a*[2]=1. Suppose the product of *a* and *b* is *c*, we construct a new array *c*[ ] which is wanted to store the each bits of the result. We have the equation:

$$c[k] = \sum_{i+j=k} a[i] * b[j].$$

Note that *c*[*k*] may large than 9 for some *k*, we need carry-over to ensure that each *c*[*k*] is a one-digit number. At last, we can output the result with the helping of *c*[*k*] (note that the order of *c*[*k*] and the order of the digits of the real result are inverse, that is, if the real result is 123, *c*[*k*] will be with *c*[*0*]=3, *c*[*1*]=2 and *c*[*2*]=1).

Step 2 – Dealing with decimals
To compute the products of two numbers which contain decimals, we can interpret the decimal with an integer and an exponent of ten. For example,

$$3.14 = 314 * 10^{-2},$$

then we denote 314 as $314 * 10^{-2}$. in this case, we can compute the product of integers using the method in **step 1** and compute the exponent of ten. For example,

$$3.14 * 2.3 = (314 * 10^{-2}) * (23 * 10^{-1}) = (314 * 23) * 10^{(-2+1)}.$$

Step 3 – Dealing with numbers given by scientific counting method
We interpret this kind of numbers to the formular in **step 2**, and we can do the multiplication. For example,

$$3.14e4 * 2e - 3 = (314 * 10^{2}) * (2 * 10^{-3}) = (314 * 2) * \left(10^{2+(-3)}\right).$$

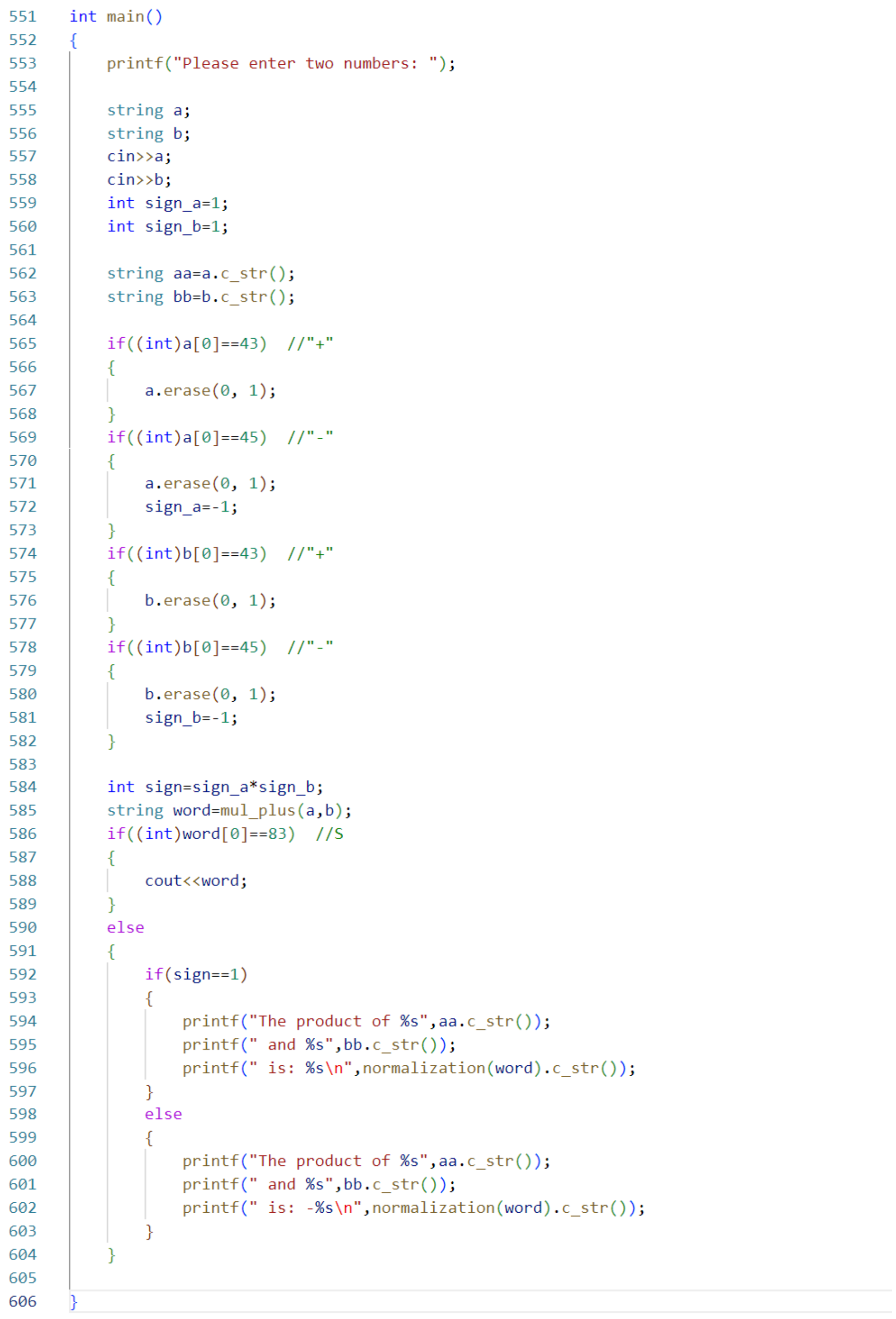
Step 4 – Computing the product of positive or negative numbers
We just need to count the numbers of the negative signs, then we can conclude the sign of the result. That is, we can tell the sign of the result firstly and do the multiplication of two positive numbers by **step 1** ~ **step 3**, then combine them.

Remark:

1. Integers, decimals and numbers given by scientific counting method are regarded as legal input; other input are illegal.
2. In my code, the result given by my method may be chaotic and don't have the uniform format. So I construct a function to transform the result to the format of standard scientific counting method. For example, result of my multiplication may occur numbers like $300e1$, then I will output $3e3$.

Part 2 – Code

```
string mul(string a,string b)  //compute the product of two positive numbers (may be large numbers)
{
    int data_a[a.size()];  //store the first large number
    int data_b[b.size()];  //store the second large number
    int data_c[a.size()+b.size()];  //store the result

    for(int i = 0; i < a.size(); i++)
    {
        data_a[a.size()-i-1] = a[i]-'0';
    }

    for(int i = 0; i < b.size(); i++)
    {
        data_b[b.size()-i-1] = b[i]-'0';
    }

    for(int i=0;i<a.size()+b.size();i++)
    {
        data_c[i]=0;
    }

    for(int i=0;i<a.size();i++)
    {
        for(int j=0;j<b.size();j++)
        {
            data_c[i+j]=data_c[i+j]+data_a[i]*data_b[j];
        }
    }

    for(int i=0;i<a.size()+b.size();i++)
    {
        if(data_c[i]>9)
        {
            data_c[i+1]=data_c[i+1]+data_c[i]/10;
            data_c[i]=data_c[i]%10;
        }
    }

    string result="";
    int flag=0;
    for(int i=a.size()+b.size()-1;i>=0;i--)
    {
        if(data_c[i]!=0)
        {
            flag=i;
            break;
        }
    }
    for(int i=flag;i>=0;i--)
    {
        result=result+to_string(data_c[i]);
    }

    return result;
}
```



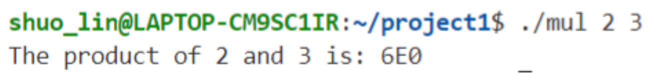
```
551 int main()
552 {
553     printf("Please enter two numbers: ");
554 
555     string a;
556     string b;
557     cin>>a;
558     cin>>b;
559     int sign_a=1;
560     int sign_b=1;
561 
562     string aa=a.c_str();
563     string bb=b.c_str();
564 
565     if((int)a[0]==43)  //"+"
566     {
567         a.erase(0, 1);
568     }
569     if((int)a[0]==45)  //"-"
570     {
571         a.erase(0, 1);
572         sign_a=-1;
573     }
574     if((int)b[0]==43)  //"+"
575     {
576         b.erase(0, 1);
577     }
578     if((int)b[0]==45)  //"-"
579     {
580         b.erase(0, 1);
581         sign_b=-1;
582     }
583 
584     int sign=sign_a*sign_b;
585     string word=mul_plus(a,b);
586     if((int)word[0]==83)  //S
587     {
588         cout<<word;
589     }
590     else
591     {
592         if(sign==1)
593         {
594             printf("The product of %s",aa.c_str());
595             printf(" and %s",bb.c_str());
596             printf(" is: %s\n",normalization(word).c_str());
597         }
598         else
599         {
600             printf("The product of %s",aa.c_str());
601             printf(" and %s",bb.c_str());
602             printf(" is: -%s\n",normalization(word).c_str());
603         }
604     }
605 
606 }
```

Remark:

1. “mul_plus” is a function computing the multiplication if input is legal while returning "Sorry! The input cannot be interpret as numbers!\n" if the input is illegal.
2. “normalization” is a function which can turn a number into standard scientific counting form.
3. In line 586, if a or b is illegal, “word”="Sorry! The input cannot be interpret as numbers!\n", in this case, the main function will output “word” directly.

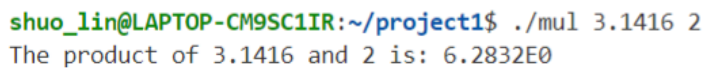
Part 3 - Result & Verification

Case 1:



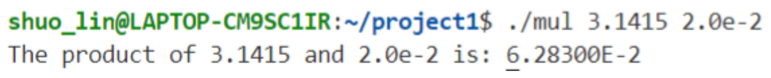
```
shuo_lin@LAPTOP-CM9SC1IR:~/project1$ ./mul 2 3
The product of 2 and 3 is: 6E0
```

Case 2:



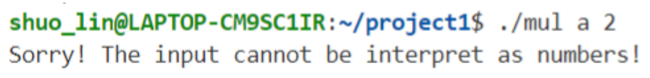
```
shuo_lin@LAPTOP-CM9SC1IR:~/project1$ ./mul 3.1416 2
The product of 3.1416 and 2 is: 6.2832E0
```

Case 3:



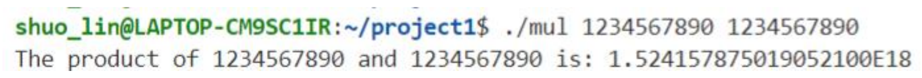
```
shuo_lin@LAPTOP-CM9SC1IR:~/project1$ ./mul 3.1415 2.0e-2
The product of 3.1415 and 2.0e-2 is: 6.28300E-2
```

Case 4:



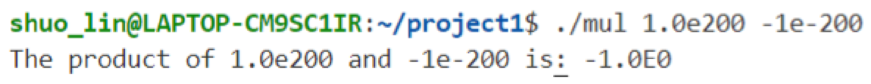
```
shuo_lin@LAPTOP-CM9SC1IR:~/project1$ ./mul a 2
Sorry! The input cannot be interpret as numbers!
```

Case 5:



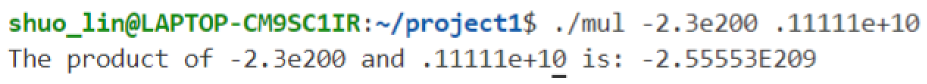
```
shuo_lin@LAPTOP-CM9SC1IR:~/project1$ ./mul 1234567890 1234567890
The product of 1234567890 and 1234567890 is: 1.524157875019052100E18
```

Case 6:



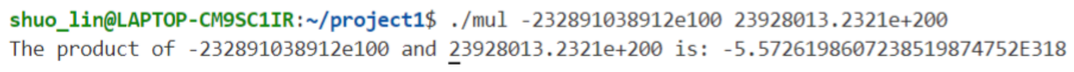
```
shuo_lin@LAPTOP-CM9SC1IR:~/project1$ ./mul 1.0e200 -1e-200
The product of 1.0e200 and -1e-200 is: -1.0E0
```

Case 7:



```
shuo_lin@LAPTOP-CM9SC1IR:~/project1$ ./mul -2.3e200 .11111e+10
The product of -2.3e200 and .11111e+10 is: -2.55553E209
```

Case 8:



```
shuo_lin@LAPTOP-CM9SC1IR:~/project1$ ./mul -232891038912e100 23928013.2321e+200
The product of -232891038912e100 and 23928013.2321e+200 is: -5.5726198607238519874752E318
```

Part 4 - Difficulties & Solutions

1. Multiplication of large numbers are not trivial and I solve it by computing the digits of the result respectively.
2. Dealing with the multiplication of decimals of scientific numbers is not easy and I solve it by turning it into a problem of integer multiplication and calculating exponents of ten.
3. There are some functions in my code to identify whether a number is an integer, a decimal, or a scientific number.
4. The output in my program uses a uniform and common format.