Project Title: Inheritance and Combination of the class Graph

1. Requirement Analysis

This project will accomplish a special string class *FunnyString,* the operator for the object of this class is defined as follow:

- Addition: the result for string A plus string B is to connect the string B to the tail of the string A, if there is the same letter they encounter, the two letters will be neutralized, and so on. For example:
  abc + cbe   result: ae
- Subtraction: the result for string A minus string B is to search the first letter of the string B in the string A, if there is the same letter in the string A, delete it from the string A, and so on. For example:
  abc - cde   result: ab
- Multiplication: the result for string A times string B is to insert the first letter in the string B after the first letter in the string A, and insert the second letter in the string B into after the second letter in the string A, and so on. For example:
  abc * def   result: adbecf

  Other operator definitions have no difference with common String operations.

  Please read Ch. 9 in the textbook by yourself and accomplish basic functions for the class *FunnyString* and overload the operators +，-，*，=, +=, -=, <<, >>. And then write a driver program (see p. 170 in the textbook). One running result for the driver program is as follows:

  Please input s1: abcd

  Please input s2: dabf

  s1 + s2 is abcabf

  s1 - s2 is c

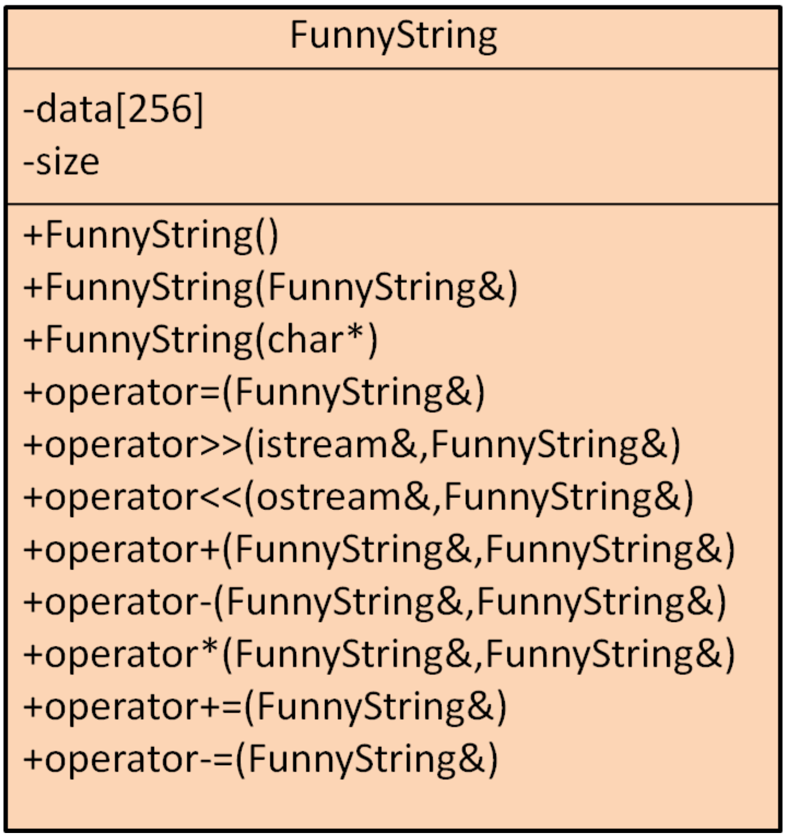
  s1+= s2 is abcabf

  s1 -= s2 is c

  s1 * s2 is adbacbdf

Any output other than this one is considered wrong.

2. Design Description


FunnyString
-data[256]
-size
+FunnyString()
+FunnyString(FunnyString&)
+FunnyString(char*)
+operator=(FunnyString&)
+operator>>(istream&,FunnyString&)
+operator<<(ostream&,FunnyString&)
+operator+(FunnyString&,FunnyString&)
+operator-(FunnyString&,FunnyString&)
+operator*(FunnyString&,FunnyString&)
+operator+=(FunnyString&)
+operator-=(FunnyString&)

3. Debug Analysis

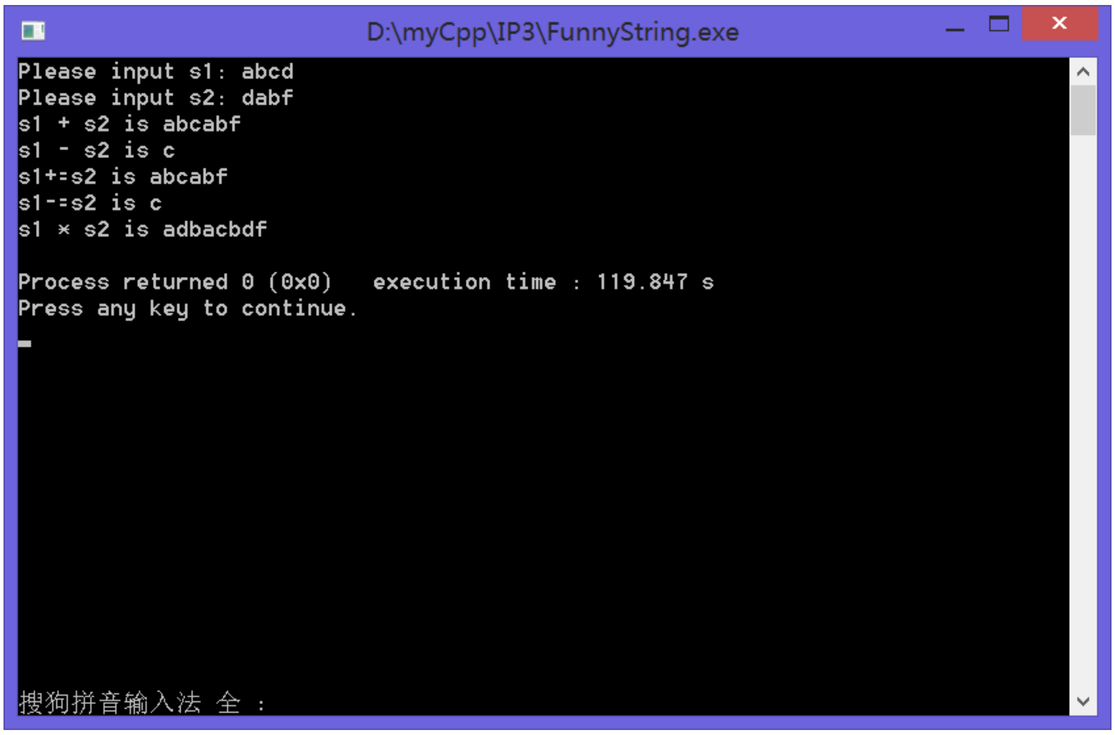
1. For the robustness and correctness of the program, troublesome conditions should be carefully taken into consideration. For instance, while overloading the operator *, if the total length of two FunnyStrings which are to be multiplied is larger than 256, the result ought to be adjusted into a size of 256.

2. When a const char* pointer is assigned to a char* pointer, typecast is required.

3. What a certain function returns also deserves considerable attention to prevent memory leakage.

4. Test Result

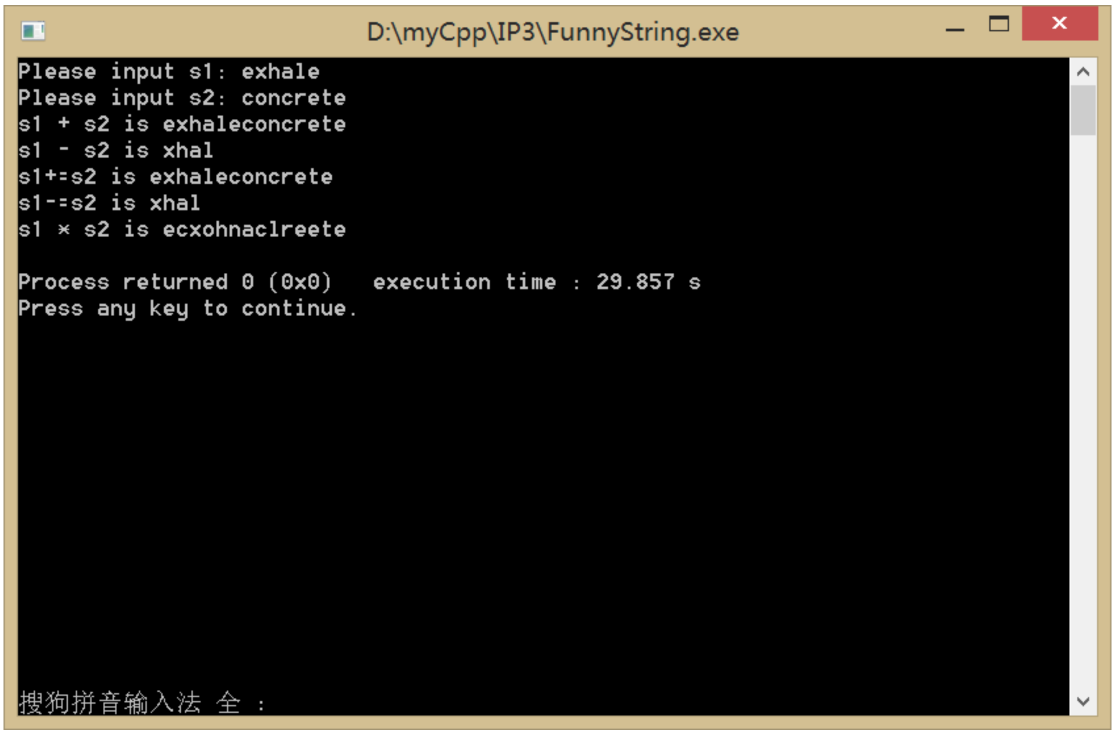
a)



```
Please input s1: abcd
Please input s2: dabf
s1 + s2 is abcabf
s1 - s2 is c
s1+=s2 is abcabf
s1-=s2 is c
s1 * s2 is adbacbdf

Process returned 0 (0x0)   execution time : 119.847 s
Press any key to continue.
```

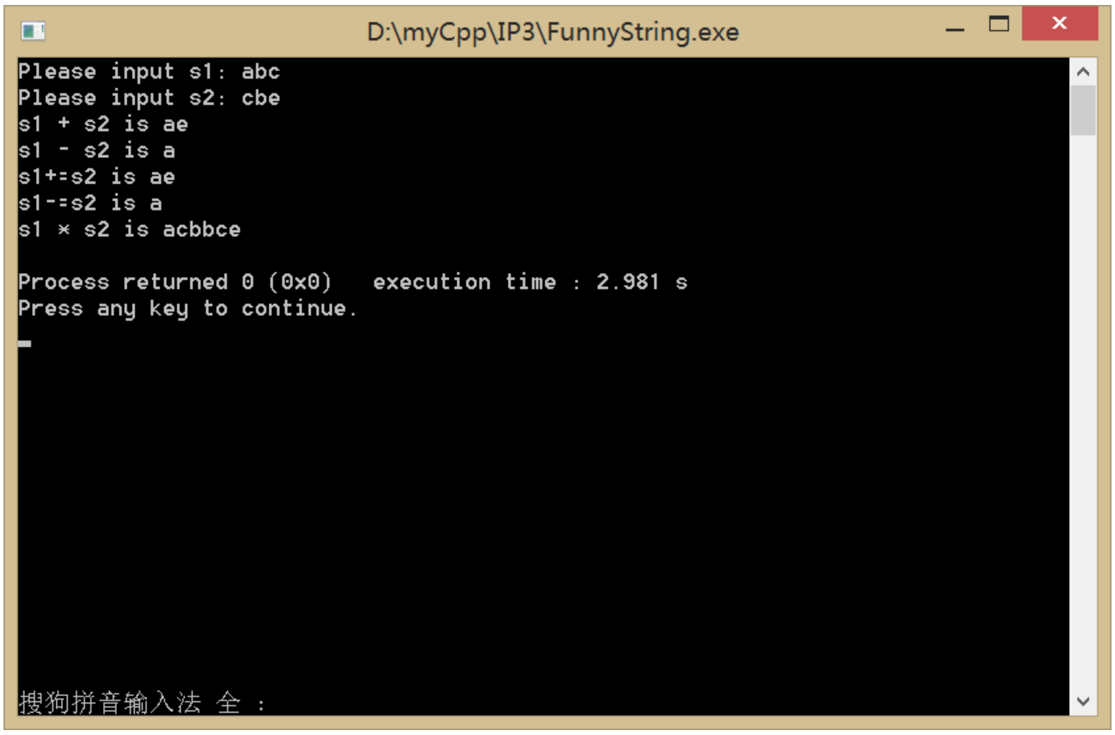
b)



```
Please input s1: exhale
Please input s2: concrete
s1 + s2 is exhaleconcrete
s1 - s2 is xhal
s1+=s2 is exhaleconcrete
s1-=s2 is xhal
s1 * s2 is ecxohnaclreete

Process returned 0 (0x0)   execution time : 29.857 s
Press any key to continue.
```

c)



```
Please input s1: abc
Please input s2: cbe
s1 + s2 is ae
s1 - s2 is a
s1+=s2 is ae
s1-=s2 is a
s1 * s2 is acbbce

Process returned 0 (0x0)   execution time : 2.981 s
Press any key to continue.
```

5. Appendix

All the files are packed together with this report in the compressed files. However, the code as well as annotation is still provided in this report in case.

[FunnyString.cpp]

```
//File name:FunnyString.cpp
//This program accomplishes and tests the application of the class FunnyString.
#include <iostream>    //For console input and output.
using namespace std;
class FunnyString
{
private:
        char data[256];
        int size;
public:
        FunnyString(){size=0;}    //Initialize an FunnyString object
        FunnyString(FunnyString& fs);    //Initialize the FunnyString object with another
one
        FunnyString(const char* str);    //Initialize the FunnyString object with a given
string
        FunnyString& operator=(const FunnyString& fs);    //Overloading operator =
        friend istream& operator>>(istream& in,FunnyString& fs);    //Overloading
operator >>
```

```
    friend ostream& operator<<(ostream& out,const FunnyString& fs);
//Overloading operator <<
    friend FunnyString operator+(const FunnyString& f1,const FunnyString& f2);
//Overloading operator +
    friend FunnyString operator-(const FunnyString& f1,const FunnyString& f2);
//Overloading operator -
    friend FunnyString operator*(const FunnyString& f1,const FunnyString& f2);
//Overloading operator *
    FunnyString& operator+=(const FunnyString& fs);    //Overloading operator +=
    FunnyString& operator-=(const FunnyString& fs);    //Overloading operator -=
};
FunnyString::FunnyString(FunnyString& fs)
{
    size=fs.size;
    for (int i=0;i<size;++i) data[i]=fs.data[i];
}    //Initialize the FunnyString object with another one
FunnyString::FunnyString(const char* str)
{
    size=0;
    while (*str!='\0')
    {
        data[size]=*str;
        ++size;
        ++str;
    }
}    //Initialize the FunnyString object with a given string
FunnyString operator+(const FunnyString& f1,const FunnyString& f2)
{
    char *p1=(char*)&(f1.data[f1.size-1]),*p2=(char*)f2.data;
    int lim=(f1.size>f2.size)?f2.size:f1.size,i=0;
    while ((*p1==*p2)&&(i<lim))
    {
        --p1;
        ++p2;
        ++i;
    }
    FunnyString tmp;
    tmp.size=f1.size+f2.size-2*i;
```

```
	for (lim=0;lim<f1.size-i;++lim) tmp.data[lim]=f1.data[lim];
	for (lim=0;lim<f2.size-i;++lim) tmp.data[lim+f1.size-i]=f2.data[lim+i];
	return tmp;
}    //Overload operator +
//Addition: the result for string A plus string B is to connect the string B to the tail of the string A,
//if there is the same letter they encounter, the two letters will be neutralized, and so on.
FunnyString operator-(const FunnyString& f1,const FunnyString& f2)
{
	bool flag[256];
	int i,t=0;
	for (i=0;i<256;++i) flag[i]=false;
	for (i=0;i<f2.size;++i) flag[(int)f2.data[i]]=true;
	char res[256];
	for (i=0;i<f1.size;++i) if (!flag[(int)f1.data[i]])
	{
		res[t]=f1.data[i];
		++t;
	}
	FunnyString tmp;
	tmp.size=t;
	for (i=0;i<t;++i) tmp.data[i]=res[i];
	return tmp;
}    //Overload operator -
//Subtraction: the result for string A minus string B is to search the first letter of the string B in the string A,
//if there is the same letter in the string A, delete it from the string A, and so on.
FunnyString operator*(const FunnyString& f1,const FunnyString& f2)
{
	int lim=(f1.size>f2.size)?f2.size:f1.size,i;
	if (lim>128) lim=128;
	FunnyString tmp;
	for (i=0;i<lim;++i)
	{
		tmp.data[2*i]=f1.data[i];
		tmp.data[2*i+1]=f2.data[i];
	}
```

```
        for                                   (i=0;((lim+i<f1.size)&&(2*lim+i<256));++i)
tmp.data[2*lim+i]=f1.data[lim+i];
        for                                   (i=0;((lim+i<f2.size)&&(2*lim+i<256));++i)
tmp.data[2*lim+i]=f2.data[lim+i];
        tmp.size=(f1.size+f2.size>256)?256:(f1.size+f2.size);
        return tmp;
}    //Overload operator *
//Multiplication: the result for string A times string B is to insert the first letter in the
string B after the first letter in the string A,
//and insert the second letter in the string B into after the second letter in the string A,
and so on.
FunnyString& FunnyString::operator+=(const FunnyString& fs)
{
        *this=*this+fs;
        return *this;
}    //Overload operator +=
FunnyString& FunnyString::operator-=(const FunnyString& fs)
{
        *this=*this-fs;
        return *this;
}    //Overload operator -=
FunnyString& FunnyString::operator=(const FunnyString& fs)
{
        if (this==&fs) return *this;
        size=fs.size;
        for (int i=0;i<size;++i) data[i]=fs.data[i];
}    //Overload operator =
istream& operator>>(istream& in,FunnyString& fs)
{
        char s[257];
        in>>s;
        fs=FunnyString(s);
}    //Overload operator >>
ostream& operator<<(ostream& out,const FunnyString& fs)
{
        for (int i=0;i<fs.size;++i) out<<fs.data[i];
        return out;
}    //Overload operator <<
```

```
int main()
{
        FunnyString s1,s2;
        cout<<"Please input s1: ";
        cin>>s1;
        cout<<"Please input s2: ";
        cin>>s2;
        cout<<"s1 + s2 is "<<s1+s2<<endl;
        cout<<"s1 - s2 is "<<s1-s2<<endl;
        FunnyString t1=s1,t2=s1;
        t1+=s2;
        t2-=s2;
        cout<<"s1+=s2 is "<<t1<<endl;
        cout<<"s1-=s2 is "<<t2<<endl;
        cout<<"s1 * s2 is "<<s1*s2<<endl;
}
```