HOMEWORK 2

- Write a code about member functions(operators) which implement String class (String.h).

```cpp
#include <iostream>
using std::ostream;
using std::istream;

class String {

friend ostream &operator<<( ostream &, const String & );
friend istream &operator>>( istream &, String & );

public:
String( const char * = "" ); // conversion/default constructor
String( const String & ); // copy constructor
~String(); // destructor

const String &operator=( const String & ); // assignment operator
const String &operator+=( const String & ); // concatenation operator

bool operator!() const; // is String empty?
bool operator==( const String & ) const; // test s1 == s2
bool operator<( const String & ) const; // test s1 < s2

// test s1 != s2
bool operator!=( const String &right ) const ;
// test s1 > s2
```
bool operator>( const String &right ) const;
// test s1 <= s2
bool operator<=( const String &right ) const;
// test s1 >= s2
bool operator>=( const String &right ) const;
char &operator[]( int ); // subscript operator (modifiable lvalue) char operator[]( int ) const;
// subscript operator (rvalue)
String operator()( int, int = 0 ) const; // return a substring int getLength() const; // return string length

private:
int length; // string length (not counting null terminator)
char *sPtr; // pointer to start of pointer-based string

void setString( const char * ); // utility function

}; // end class String

String operator()( int, int = 0 ) const; // return a substring int getLength() const; // return string length

private:
int length; // string length (not counting null terminator)
char *sPtr; // pointer to start of pointer-based string

void setString( const char * ); // utility function

}; // end class String
```cpp
#include <iostream>
using std::cout;
using std::endl;
using std::boolalpha;

#include "String.h"

int main() {

String s1( "happy" );
String s2( " birthday" ); String s3;

// test overloaded equality and relational operators
cout << "s1 is \"happy\" \"; s2 is \"W"\" \"s3 is \"W\"\" \"s2 \< s1 \"W\"; s2 \< s1 \"W\"\" \"s3 \< s1 \"W\";
<< boolalpha << "The results of comparing s2 and s1:" << "Wns2 \= s1 yields \" ( s2 \= s1 )
<< "Wns2 > s1 yields \" ( s2 > s1 )
<< "Wns2 < s1 yields \" ( s2 < s1 )
<< "Wns2 >= s1 yields \" ( s2 >= s1 )
<< "Wns2 <= s1 yields \" ( s2 <= s1 );
// test overloaded String empty (!) operator
cout << "Testing !s3:" << endl;

if ( !s3 ) {
  cout << "s3 is empty; assigning s1 to s3;" << endl; s3 = s1; // test overloaded assignment
  cout << "s3 is \"W\"\" << s3 << \"W\"\"; } // end if

// test overloaded String concatenation operator
```
cout << "WnWns1 += s2 yields s1 = ";
s1 += s2; // test overloaded concatenation cout << s1;

// test conversion constructor
cout << "WnWns1 += W" to youW" yields" << endl; s1 += " to you"; // test conversion constructor cout << "s1 = " << s1 << "WnWn";

// test overloaded function call operator () for substring
cout << "The substring of s1 starting atWn"
<< "location 0 for 14 characters, s1(0, 14), is:Wn" << s1(0, 14) << "WnWn";

// test substring "to-end-of-String" option
cout << "The substring of s1 starting atWn"
<< "location 15, s1(15), is: "
<< s1(15) << "WnWn";

// test copy constructor
String *s4Ptr = new String(s1);
cout << "Wn*s4Ptr = " << *s4Ptr << "WnWn";

// test assignment (=) operator with self-assignment
cout << "assigning *s4Ptr to *s4Ptr" << endl;

*s4Ptr = *s4Ptr; // test overloaded assignment cout << "*s4Ptr = " << *s4Ptr << endl;

// test destructor
delete s4Ptr;

// test using subscript operator to create a modifiable lvalue
s1[ 0 ] = 'H';
s1[ 6 ] = 'B';
cout << "Wns1 after s1[0] = 'H' and s1[6] = 'B' is: "
     << s1 << "WnWn";

// test subscript out of range
cout << "Attempt to assign 'd' to s1[30] yields:" << endl;
s1[ 30 ] = 'd'; // ERROR: subscript out of range
return 0;

} // end main
Submission – Compressed file : source code and report

Mail title: [COMP-HW2]student id_name
Compressed file name: HW2_student id_name.zip(tar)
Email : will be update an email address on a web-page.
Deadline : May 14, until 23:59:59

You should keep upper form.

★ Caution
- Over the deadline ; after May 14, 23:59:59 – minus 20% score
- 2days late, 0 point
- Do not keep the upper form – minus 20% score
- Compile error question - 0 point
- Check a code copy using Clone checker – related students 0 point

1. Source Code
   Visual, gcc file, both are ok.
   Make readme file is ok.

2. Report
   - Contain specific explain about code
   - Contain screen capture file.
   - PDF, DOC, HWP file.