CSCI 235, Programming Languages, C^{++}
Exercise 4

Deadline: 17/19.09.2018 (the day of your assigned lab)

Topic of this exercise are moving semantics, rvalue references, and temporaries.

1. Extend the **string** class, which you have downloaded and extended last week, with a moving constructor and a moving assignment.

   ```
   string( string&& s ) noexcept;
   const string& operator = ( string&& s ) noexcept;
   ```

   Check that the moving operators do indeed move. Test carefully with **valgrind**. Since we want to study the essential methods (assignment, construction, and destruction), put print statements in the two copy constructors, the two assignment operators, and the destructors. It must be possible to see which one is being called.

   Unfortunately, one cannot call
   `std::ostream& operator << ( std::ostream&, const string& )` from inside the definition of class **string**, because it is declared only after the class. This problem can be solved in two possible ways:

   (a) Write

   ```
   class string;
   std::ostream& operator << ( std::ostream&, const string& );
   ```

   in front of **class string** in **string.h**. Now you can put print statements in each of the 5 operators inside the class definition.

   (b) Move all implementations (of copy constructors, assignments and desctructor) to file **string.cpp**.

   I think that **(1)** is better.

2. Find out, in the following situations, which of the constructors, assignment operators, are called. is used:

```
string s1 = (your name);
string s2 = "hallo";

std::swap( s1, s2 );

std::cout << ( s1 + ", " + s2 ) << "\n";
std::cout << string( "this is a string" ) << "\n";
s2 = ( s1 + "!" );
```

3. Let us study the life time of temporaries a bit better: Write a function

   ```
   const string& greater( const string& s1, const string& s2 );
   ```

   that returns the greater of two strings, using standard lexicographic ordering. This is very easy to write because you can use `operator <` that you wrote last week.

   Try it for example on

   ```
   string zz = "zz";
   std::cout << "before:\n";
   string max = greater( greater( "aa", string( "b" ) + string( "c" )),
                         greater( zz, string( "cc" )));
   std::cout << "after:\n";
   std::cout << "max = " << max << "\n";
   ```

   What are the temporaries? How long did they exist? Make sure that you understand all output (from the prints in the essential methods) that you see.

4. Next, add a function

   ```
   string&& greater( string&& s1, string&& s2 );
   ```

   Make sure that it really moves. Put the following lines in the two `greater` functions, so that you can see which one is being called:

   ```
   std::cout << "reference greater " << s1 << " " << s2 << "\n";

   std::cout << "moving greater " << s1 << " " << s2 << "\n";
   ```

   Run the previous example and see what is changed. Make sure that you fully understand what is going on.