CS205 C/ C++ Programming - Assignment 5

Name: 唐千栋（Qiandong Tang)

SID: 11612730

Part 1 - Analysis

In this assignment, we need to extend the UTF8string class by adding some functions and operators. In order to support UTF-8 characters, we use functions in `utf8.h`, such as `utf8_charlen()` and `utf8_bytes_to_charpos()`. We use `std::string` as a member variable to store the string, and use its member functions to support UTF-8 characters.

Part 2 - Code

UTF8string.cpp

```
#include "UTF8string.hpp"

int UTF8string::length() {
    auto *p = (unsigned char *) _str.c_str();
    return utf8_charlen(p);
}

int UTF8string::bytes() {
    auto *p = (unsigned char *) _str.c_str();
    return utf8_charpos_to_bytes(p, length());
}

int UTF8string::find(std::string substr) {
    auto *p = (unsigned char *) _str.c_str();
    unsigned char *ret = utf8_search(p, (unsigned char *) substr.c_str());
    return utf8_bytes_to_charpos(p, ret - p);
}

void UTF8string::replace(UTF8string to_remove, UTF8string replace) {
    auto *start = (unsigned char *) _str.c_str();
    unsigned char *p = start;
    unsigned char *ret;
    while (true) {
        ret = utf8_search(p, (unsigned char *) to_remove.str().c_str());
        if (ret == NULL) break;
        _str.replace(ret - start, to_remove.str().length(), replace.str());
```

```cpp
        p = ret + to_remove.str().length();
    }
}

std::ostream &operator<<(std::ostream &out, const UTF8string &us) {
    out << us.str();
    return out;
}

UTF8string UTF8string::operator+(const UTF8string &rhs) const {
    UTF8string res(_str + rhs.str());
    return res;
}

UTF8string &UTF8string::operator+=(const UTF8string &rhs) {
    _str += rhs.str();
    return (*this);
}

UTF8string UTF8string::operator*(const int &rhs) {
    std::string str;
    for (int i = 0; i < rhs; i++) {
        str += _str;
    }
    UTF8string res(str);
    return res;
}

UTF8string UTF8string::operator!() {
    auto *start = (unsigned char *) _str.c_str();
    unsigned char *p = start;
    std::string str = _str;
    int bytes_in_char;
    unsigned int codepoint;
    while (*p) {
        codepoint = utf8_to_codepoint(p, &bytes_in_char);
        if (codepoint) {
            std::reverse(str.begin() + (p - start), str.begin() + (p - start + bytes_in_char));
            _utf8_incr(p);
        } else {
            printf("%c Invalid UTF-8\n", *p);
            p++;
        }
    }
    std::reverse(str.begin(), str.end());
    return UTF8string(str);
}

```

```
UTF8string operator*(const int &lhs, const UTF8string &rhs) {
    std::string str;
    for (int i = 0; i < lhs; i++) {
        str += rhs._str;
    }
    UTF8string res(str);
    return res;
}

```

UTF8string.hpp

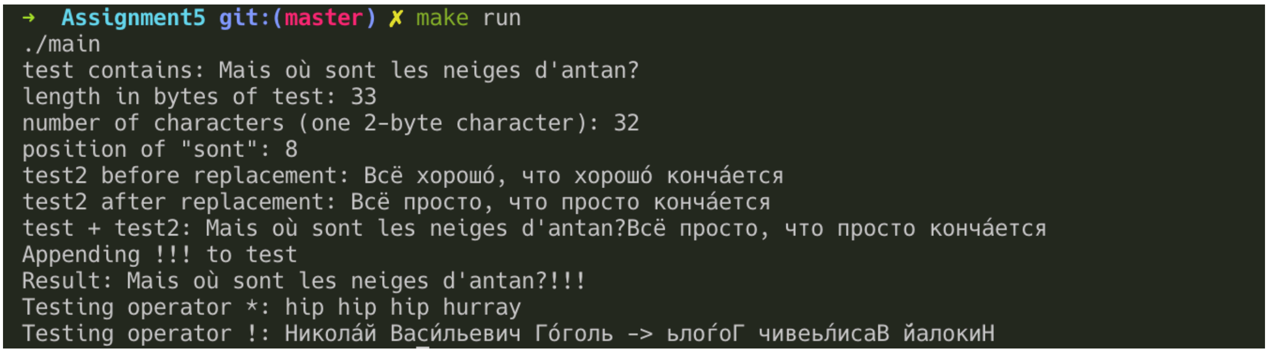
```
#ifndef UTF8STRING_HPP
#define UTF8STRING_HPP

#include <cstdio>
#include <cstring>
#include <string>
#include <algorithm>
#include <iostream>
#include "utf8.h"

class UTF8string {
public:
    UTF8string() {};

    UTF8string(const char *s) : _str(s) {}

    UTF8string(std::string s) : _str(s) {}

    std::string str() const {
        return _str;
    }

    int length();

    int bytes();

    int find(std::string substr);

    void replace(UTF8string to_remove, UTF8string replace);

    friend std::ostream &operator<<(std::ostream &out, const UTF8string &us);

    UTF8string operator+(const UTF8string &rhs) const;

    UTF8string &operator+=(const UTF8string &rhs);

    UTF8string operator*(const int &rhs);

    friend UTF8string operator*(const int &lhs, const UTF8string &rhs);

    UTF8string operator!();

private:
    std::string _str;
};

#endif

```

Part 3 - Result & Verification

> The verification is combined with the result queried in the website.

Test case #1:

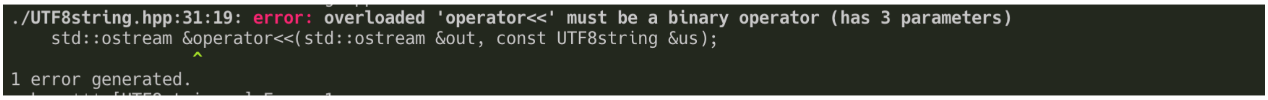
```
#include <iostream>
#include "UTF8string.hpp"

using namespace std;

static void func(UTF8string u) {
    // Function to make sure that nothing crashes
    cout << "Testing operator !: " << u << " -> " << !u << endl;
}

int main() {
    //
    // Expected output:
    // ---------------
    // test contains: Mais où sont les neiges d'antan?
    // length in bytes of test: 33
    // number of characters (one 2-byte character): 32
    // position of "sont": 8
    // test2 before replacement: Всё хорошó, что хорошó кончáется
    // test2 after replacement: Всё просто, что просто кончáется
    // test + test2: Mais où sont les neiges d'antan?Всё просто, что просто кончáется
    // Appending !!! to test
    // Result: Mais où sont les neiges d'antan?!!!
    // Testing operator *: hip hip hip hurray
    // Testing operator !: Николáй Васíльевич Гóголь -> ьлоѓоГ чивеьл́исаВ й́алокиН
    //
    UTF8string test = UTF8string("Mais où sont les neiges d'antan?");
    UTF8string test2 = UTF8string("Всё хорошó, что хорошó кончáется");
    UTF8string test3("hip ");
    UTF8string test4("Никола́й Васи́льевич Го́голь");
    cout << "test contains: " << test << endl;
    cout << "length in bytes of test: " << test.bytes() << endl;
    cout << "number of characters (one 2-byte character): " << test.length() <<
endl;
    cout << "position of \"sont\": " << test.find("sont") << endl;
    cout << "test2 before replacement: " << test2 << endl;
    test2.replace("хорошо́", "просто");
    cout << "test2 after replacement: " << test2 << endl;
    cout << "test + test2: " << test + test2 << endl;
    cout << "Appending !!! to test" << endl;
    test += UTF8string("!!!");
    cout << "Result: " << test << endl;
    cout << "Testing operator *: " << test3 * 3 << "hurray" << endl;
    func(test4);
    return 0;
}

```



```
➜  Assignment5 git:(master) ✗ make run
./main
test contains: Mais où sont les neiges d'antan?
length in bytes of test: 33
number of characters (one 2-byte character): 32
position of "sont": 8
test2 before replacement: Всё хорошо́, что хорошо́ конча́ется
test2 after replacement: Всё просто, что просто конча́ется
test + test2: Mais où sont les neiges d'antan?Всё просто, что просто конча́ется
Appending !!! to test
Result: Mais où sont les neiges d'antan?!!!
Testing operator *: hip hip hip hurray
Testing operator !: Никола́й Васи́льевич Го́голь -> ьлоѓоГ чивеьл́исаВ йалокиН
```

Part 4 - Difficulties & Solutions

1. overload operator << meets error



```
./UTF8string.hpp:31:19: error: overloaded 'operator<<' must be a binary operator (has 3 parameters)
    std::ostream &operator<<(std::ostream &out, const UTF8string &us);
                  ^

1 error generated.
```

Declare friend for operator <<