CS205 C/C++ Programming Lab Assignment 5

Name: 钟兆玮 (Zhaowei Zhong)
SID: 11611722

Part 1 - Analysis

Note:

For this lab assignment, you **ONLY** upload UTF8string.cpp and UTF8string.hpp.
You **MUST NOT** modify utf8.h and utf8.c because they are external libraries.
You **MUST** use std::string as a member variable to store the string, and you are recommended to use its member functions to make your code clearer.
You **MUST NOT** use std::u16string or std::u32string because you are required to use the utf8.c library to deal with Unicode.
Your code **MUST** pass the given test program and have the expected output.
Please write necessary comments for your code.

Part 1

This lab will use the UTF8 functions that you are now familiar with and will make you combine C and C++.
You are asked to create a class called UTF8string; the difference between UTF8string and a regular C++ string is that UTF8string knows "characters" when a string only knows bytes.
The following is provided to simplify your work:

- Test program (testUTF8string.cpp)

You must also use utf8.c and utf8.h. However, you should note that some modifications are required for the C++ compiler to know that the code needs to be compiled in C (C and C++ are incompatible in various ways).

```
#ifdef __cplusplus
extern "C" {
#endif
extern int utf8_charlen(unsigned char *p);
extern int utf8_bytes_to_charpos(unsigned char *s, int pos);
extern ...
#ifdef __cplusplus
}
#endif
```

Because rules for finding the right function (the technical name is "resolving") are different in C and C++, this is required to tell the linker that these are C, not C++, functions and that C rules should apply.
You mustn't derive the class from the string class (which wasn't designed as a base class); however, you should use a string attribute to store the string.
You are asked to write the four following methods:

- length(), that returns the length IN CHARACTERS of the UTF8string
- bytes(), that returns the number of bytes used for storing the UTF8string
- find(string substr), that returns the CHARACTER POSITION where substr starts.

  For instance, in "Mais où sont les neiges d'antan", find() should find that "sont" starts at character 8, even if 'ù' is stored on two bytes.
- replace(UTF8string to_remove, UTF8string replacement), that replaces to_remove with replacement.

You'll have to mix C (char *) strings with the C++ std::string type. It's fairly easy to switch between both; there is a constructor that constructs a string from a char * C string passed as parameter; and the method c_str() applied to a C++ std::string returns a pointer to a '\0' terminated sequence of C chars.

Part 2

We'll extend the UTF8string class by adding overloaded operators.
You are asked to redefine:

<< i.e. support std::cout << ustr << std::endl;

+ that gives regular concatenation (if two objects are called u1 and u2, u1 + u2 changes neither u1 nor u2)

+= to append another string (u1 += u2 changes u1, not u2)

* for repeating a string n times (if u is "àéèç", u * 2 or 2 * u should return "àéèçàéèç" without changing u)

! for reversing a string (without modifying original string), which means reversing the characters (not the bytes!), for instance if u is "étudiant" (student in French), !u should be "tnaiduté".

Part 2 - Code

UTF8string.hpp

```
#ifndef UTF8_STRING_HPP
#define UTF8_STRING_HPP
```

```
#include <iostream>
#include <string>

class UTF8string {
public:
  UTF8string(std::string str);
  int length();
  int bytes();
  int find(std::string substr);
  void replace(std::string to_remove, std::string replacement);
  friend std::ostream & operator<<(std::ostream& os, const UTF8string&
str);
  UTF8string operator+(const UTF8string &str2) const;
  void operator+=(const UTF8string &str2);
  friend UTF8string operator*(const UTF8string str, const int times);
  friend UTF8string operator*(const int times, const UTF8string str);
  UTF8string operator!() const;
private:
  std::string str;
};

#endif //UTF8_STRING_HPP
```

UTF8string.cpp

```
#include <iostream>
#include <string>
#include <algorithm>
#include "UTF8string.hpp"
#include "utf8.h"

UTF8string::UTF8string(std::string str) : str(str) {}

int UTF8string::length() {
  int length = 0;
  unsigned char *p;
  p = (unsigned char *)(this->str.c_str());
  while (*p) {
    length++;
    int bytes_in_char;
    int codepoint = utf8_to_codepoint(p, &bytes_in_char);
    if (codepoint) {
      _utf8_incr(p);
    } else {
      exit(-1);
```

```
    }
  }
  return length;
}

int UTF8string::bytes() {
  return this->str.length();
}

int UTF8string::find(std::string substr) {
  int pos = this->str.find(substr);
  int length = 0;
  unsigned char *p;
  p = (unsigned char *)(this->str.substr(0, pos).c_str());
  while (*p) {
    length++;
    int bytes_in_char;
    int codepoint = utf8_to_codepoint(p, &bytes_in_char);
    if (codepoint) {
      _utf8_incr(p);
    } else {
      exit(-1);
    }
  }
  return length;
}

void UTF8string::replace(std::string to_remove, std::string replacement) {
  int position = 0;
  while((position = this->str.find(to_remove, position)) !=
std::string::npos) {
    int begin = position;
    this->str.replace(begin, to_remove.length(), replacement);
  }
}

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

UTF8string UTF8string::operator+(const UTF8string &str2) const {
  return UTF8string(this->str + str2.str);
}
```

```
void UTF8string::operator+=(const UTF8string &str2) {
  this->str = this->str + str2.str;
}

UTF8string operator*(const UTF8string str, const int times) {
  UTF8string new_string = UTF8string("");
  for (int i = 0; i < times; i++) {
    new_string += str;
  }
  return new_string;
}

UTF8string operator*(const int times, const UTF8string str) {
  UTF8string new_string = UTF8string("");
  for (int i = 0; i < times; i++) {
    new_string += str;
  }
  return new_string;
}

UTF8string UTF8string::operator!() const {
  int arr[str.length()];
  int length = 0;
  unsigned char *p;
  p = (unsigned char *)(str.c_str());
  while (*p) {
    int bytes_in_char;
    int codepoint = utf8_to_codepoint(p, &bytes_in_char);
    if (codepoint) {
      arr[length] = codepoint;
     _utf8_incr(p);
    } else {
     exit(-1);
    }
    length++;
  }
  std::string result = "";
  for (int i = length - 1; i >= 0; i--) {
    unsigned char *utf8 = (unsigned char *)malloc(sizeof(unsigned char) *
5);;
    result = result + std::string((char*)codepoint_to_utf8(arr[i], utf8));
  }
  return result;
```

```
}
```

Part 3 - Result & Verification

Test Case: testUTF8string.cpp

```
~/Courses/CPP/assignment5  master  ./testUTF8string
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 чивеьлисаВ йалокиН
```

Part 4 - Difficulties & Solutions

Reversing the UTF-8 string is difficult to be implemented. After thorough deliberation, I found the right solution of this problem.