CS205 - LAB4

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.

What is provided to you is the following:

- Header class skeleton

- Test program

You must also use utf8.c and utf8.h. However (IMPORTANT!) you must include the following lines (in red and bold) around the function definitions in utf8.h:

```
#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 9, 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++ 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++ string returns a pointer to a '\0' terminated sequence of C chars.