SHIFT CIPHER LAB

CSCI 2270, Fall 2014

8/26/2014

Purpose: to get you warmed up on using code right away, and to help you remember how to manipulate an array, which in this case is a C++ string.

Background on C++ strings if you need it: http://www.mochima.com/tutorials/strings.html

It is possible to 'encrypt' text by encoding it with a shift cipher, which replaces every letter in a lower-case word (using only a-z) by a letter a certain distance away, so that:

"abc" shifted by 1 becomes "bcd"

"abc" shifted by 3 becomes "def"

"abc" shifted by 26 becomes "abc " again

"abc" shifted by 25 becomes "zab"

(Obviously, this cipher's still pretty easy to break.) In this lab, you'll write the code to take a string of lowercase letters (like "bottle") from the user, and make a new string with the shifted word.

Here are a few hints, so you have the functions I used to solve this problem:

In C++, the string class requires you to have the line:

```
#include <string>
```

at the top of the file, and it's in the std namespace, like cout and endl, so it's good to have the line:

```
using namespace std;
```

if you want to avoid typing std::string all the time.

C++ strings can tell you their length:

```
string s = "banana";

cout << s[0] << endl;      // prints the first b

cout << s.length() << endl;          // prints 6
```

And strings, like other C++ arrays, can give you individual characters (letters) in the string. The first letter in the string above, 'b', is s[0], and the second, 'a', is s[1], and the last letter, 'a', is s[5]. (Remember that we start counting array indexes at 0 in C++.)

Any C++ character (a letter in a string) can also be interpreted as a number, called its ASCII code. The letter 'a', for instance, has the ASCII code 97. The letter 'b' has ASCII code 98, and 'c' has ASCII code 99. If we declare an int variable and set it using a character in the string, we get the character's ASCII code:

```
int ascii_letter = s[1];     // sets ascii_letter to the second letter in the string, the 'a', which is 97
```

We can also change a character in the string using the ASCII code:

```
s[2] = 98;                   // changes the letter 'n' in 'banana' to a b

cout << s << endl;           // prints "babana"
```

Using the ASCII codes makes our shift cipher job look easier; we'll just add the shift to the ASCII code for each letter. But what about shifting a string like "xyz" by 3 to "abc"? in this case, you need the shift to wrap around from the end of the alphabet to the front. You can solve this problem using the modulus function (%), which gives you the remainder of a division:

```
cout << 45 % 7 << endl;  // prints 3
```

I will leave it to you and your labmates to work out the exact formula; it's fine to do this collaboratively. The general idea is to get a string from the user, and then write a loop over the letters of the string that replaces each letter with its shifted counterpart, and then print the shift. You should be able to do this with a little C++ help from the TA (that is fine) and a little bit of math. When you are done, upload your code to the moodle, show the TA that your shift works, receive praise, and leave happy. (And seriously, come to my office hours if this lab was hard to complete; we'll get you caught up.)