COMP-SCI 201L (FS17) - **Section 2**

Lab 3 - 09/19/17

Caesar Cipher

In cryptography, **Caesar cipher** is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, `'D'` would be replaced by `'A'`, `'E'` would become `'B'`, and so on.

The transformation can be represented by aligning two alphabets; the cipher alphabet is the plain alphabet rotated left or right by some number of positions. For instance, Figure 1 is a Caesar cipher using a left rotation of 3 places, equivalent to a right shift of 23 (the shift parameter is used as the **key**).

```
Plain:  ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher: XYZABCDEFGHIJKLMNOPQRSTUVW
```

Figure 1. Plain and cipher text in a Caesar cipher using a left rotation of 3 places.

When encrypting, the program looks up each letter of the message in the "plain" line and writes down the corresponding letter in the "cipher" line. Deciphering is done **in reverse**, with a right shift of 3. The encryption can also be represented using **modular arithmetic** by first transforming the letters into numbers, according to the scheme of `'A'` ↦ 0, `'B'` ↦ 1, ..., `'Z'` ↦ 25.

Then, the encryption of a letter x by a shift n can be described mathematically as

$$E_n(x) = (x + n) \bmod 26.$$

Decryption is performed similarly as

$$D_n(x) = (x - n) \bmod 26.$$

Lab Instruction

1. Create a new empty visual C++ project named `"Lab3"`.
2. Download files `"CaesarCipher.h"`, `"CaesarCipher.cpp"` and `"Lab3.cpp"` from UMKC Blackboard. Then add the files to the existing project.
3. In the file `"CaesarCipher.h"`, write a **`class`** named `CaesarCipher` with the members below.
   - The `class` has two `private` variables `"key1"` and `"key2"`, which are both of integer type.
   - `private` member function **`Encode()`** that takes a single character as parameter, and returns the encrypted character, by rotating the character using `key1` followed by `key2`.
   - `private` member function **`Decode()`** that takes a single character as parameter, and returns the decrypted character, by **reversing** the `Encode()` operation using `key1` followed by `key2`.

- **Default constructor** that initializes `key1` and `key2` as **2 random numbers in the range [−9, 9]**. If a key is positive, it means "right rotation"; if a key is negative, it means "left rotation". In C++, the method to generate a random number (or pseudo-random number) is as followed.

```
srand(time(0));                    // Reset the "seed" using current time.
int value = rand() % 10;           // Generate a random number between 0 and 9.
int sign = rand() % 2;             // Generate a random number which is either 0 or 1.
sign = sign == 0 ? -1 : 1;         // Change the value to either -1 or 1.
int randomNumber = value * sign;   // Final random number
```

- `public` member function **`Encode()`** that takes a **`string`** as parameter, and returns the encrypted `string`, by encoding each character one-by-one in the `string`. For each space character in the `string`, the function just adds a space in the encrypted text **without rotation**.
- `public` member function **`Decode()`** that takes a **`string`** as parameter, and returns the decrypted `string`, by decoding each character one-by-one in the `string`.

4. In the file `"CaesarCipher.cpp"`, implement the member functions of `class CaesarCipher`.
5. Write the **`main()`** function in the file `"Lab3.cpp"`.
   - The program reads several sentences stored in the file `"input.txt"`. Each sentence occupies a whole line in the input file. Since a sentence may contain several spaces, the `">>"` operator is not quite useful here. Instead, we can use the `getline` method.

     ```
     string s;          // A string variable
     getline(fin, s);   // Read a entire line from fin and save the content to s.
     ```

     The input file is guaranteed to have **no more than 20 sentences**, and **only contains lower-case letters and spaces**.
   - The program encodes each sentence read from the input file, and outputs the encrypted text to the file `"output.txt"`.
   - The program decodes each encrypted sentence performed in the previous step, and outputs the decrypted text to the file `"output.txt"`.
   - **Remember: after an `Encode()` operation and a `Decode()` operation, the text should be exactly the same as the original text!**

Submission and Grading

1. Submit your code via Blackboard **no later than Thursday, 09/21/17 @ 11:59 pm**. Only submit files of `"CaesarCipher.h"`, `"CaesarCipher.cpp"` and `"Lab3.cpp"` (**unzipped**).
2. Your code will be graded using another set of data (still named `"input.txt"`) to see if your program can generate the correct results. The lab instructor will also manually look at your code for other grading criteria.
3. The input file used for grading is guaranteed to have no errors of any kind.