Project 4: poly-Caesar Cipher

Due: Monday June 4, 11:59 pm

1 Assignment Overview

This project focuses on manipulating strings. You will write functions that perform a variation on the popular Caeasar cipher.

2 Assignment Deliverables

You must turn in completed versions of the following files:

- proj04/cipher.cpp

Be sure to use the specified file name and to submit your files for grading via **Mimir** before the project deadline.

3 Background

The Caesar cipher is one of the simplest and most widely known encryption techniques. In the Caesar cipher, each letter of the original plaintext is replaced by another letter by shifting by a number of positions (with 'a' following after 'z' as necessary). For instance, if we shift by 3 then 'a' becomes ''d' and 'y' becomes 'b'.

Unfortunatly, the Caesar cipher offers very little information security as it is very easy to break. We will make it significantly harder to break by using different substitutions for each character by using a secret *keyword*. For each letter in the plaintext (the message to be encoded), we match it with one letter from the keyword, matching the 1st character of each, and then the second and so on, repeating the keyword as necessary. We then compute a ciphertext character for each pair of letters.

To encode letter x with ciphertext letter n we interpret both letters as numbers with $'a' = 0$, $'b' = 1$ and so on to $'z' = 25$. We then add the two together (in mod 26 arithmetic) and then reinterpret the number as a character.

To decode a cipher, we subtract instead of add, but the steps are otherwise the same.

4 Assignment Specifications

In this and in future projects we will provide *exactly* our function specifications: the function name, its return type, its arguments, and each arguments type. The functions will be tested individually in Mimir using these exact function specifications. If you do not follow the function specifications, these independent tests of your functions will fail. Do not change the function declarations!

What you test on Mimir is a file that contains only the functions. You do not turn in a main program. We can test the functions individually on Mimir. However, you should write your own main program to test your functions separate from Mimir. It is more flexible and you can debug more easily.

function `EncodeLetter`
params: plain: char, key: char
returns: char
function `DecodeLetter`
params: cipher: char, key: char
returns: char
function `EncodeString`
params: plaintext: const string&, keyword: const string&
returns: string
function `DecodeString`
params: ciphertext: const string&, keyword: const string&
returns: string

5 Assignment Notes

- You will receive no points if your solution does not compile on Mimir.
- Points will be deducted if your solution has any compiler warnings.
- The `<string>` library contains a number of useful methods.
- You *do not* need to check for bad input values for this project. In future projects we will explicitly indicate the errors we are looking for.
- Note that the C++ standard rounds integer division towards zero (truncates) which gives you a negative remainder in some cases, but you need to always have positive remainders. You will need to check for this and get a positive remainder.
- You may assume that the string is all lowercase.
- The encoding of "attackatdawn" with the keyword "lemon" should yield the ciphertext "lxfopvefrnhr".
- If you decode the string that you encode using the same keyword you should get the same string back.