**Caesar Cipher**

**Project Description & Goal**

The goal of this project is to develop an application that will perform a Caesar Cipher shift on a user-inputted string and shift #, and return back to the user the encrypted string. A Caesar Cipher shift is simply a wrapping letter change based on the shift # such that on input “hello”, and shift # +1 the result would yield “Ifmmp”. The bulk of this project is straight forward, considering chars can be converted implicitly to integers, shifted, and then converted back to chars. However, the tricky part of this assignment is handling the wrapping of characters towards the beginning or end of the alphabet. For example, if ‘z’ is shifted +2 times, the result should yield ‘b’. Likewise, if ‘b’ is shifted -2 times, the result should yield ‘z’. We will also build a “brute force” shifter, that will take in an encrypted string and display all 25 possible shift options.

**Project Specifications**

* Assume that the user will ONLY input lowercase letters (and uppercase if you are doing the optional part).
* Assume that the user will ONLY enter shift values between -25 and 25.

**Project Learning Objectives**

* Know how to handle implicit and explicit data conversions.
* Learn how to manipulate user input and return it back to the user.
* Learn about brute forcing and understand the complexities that come with it.
* Learn how methods can be incredibly useful for neat code.

**Project Demonstrated Competencies**

1. Alphabet correctly shifts forwards and backwards circularly
2. Solution makes use of implicit data conversions to shift each character
3. Brute force displays all possible outcomes of the Caesar Cipher shift.
4. Encrypt, Decrypt, and Brute Force are put into methods and called from main.
5. Optional: Program handles both lowercase and uppercase letters. A uppercase letter shifted will always yield an uppercase letter, and a lowercase letter will always yield a lowercase letter.
6. Optional: Program can handle punctuation and spaces. It handles them by ignoring them completely and simply printing the original symbol.

**Rubric**

|  |  |  |
| --- | --- | --- |
|  | **Description of perfect implementation** | **Score** |
| Competency #1 | Each individual character of the input string correctly shifts based on the shift #. The shift will also wrap around the alphabet such that ‘z’ + 2 = ‘b’. | \_\_\_  25 |
| Competency #2 | Each character is shifted by implicit conversion to an integer, adding the shift #, and then implicitly converting back to a character for display. IE there is NOT a huge list of if statements along the lines of, “if (char == ‘a’ && shift == 2) outputChar = ‘c’;” | \_\_\_  25 |
| Competency #3 | Each possible outcome of the brute force technique is shown on individual lines in the command window (all 25 possibilities). Human eyes are amazing, out of this list we will easily be able to find decrypted phrases. | \_\_\_  25 |
| Competency #4 | You have your main method, but you also have Encrypt(input, shiftVal), Decrypt(input, shiftVal), and BruteForce(input) methods. | \_\_\_  25 |
| Competency #5 | Program considers both uppwercase and lowercase letters. “Hello” shifted 24 times will yield “Gdkkn” | \_\_\_  +10 |
| Competency #6 | Any punctuation and spaces in your string will be ignored and will be printed out what it originally was. | \_\_\_  +10 |