Lab Exercise 02: Brute Force Cryptanalysis

BACKGROUND:

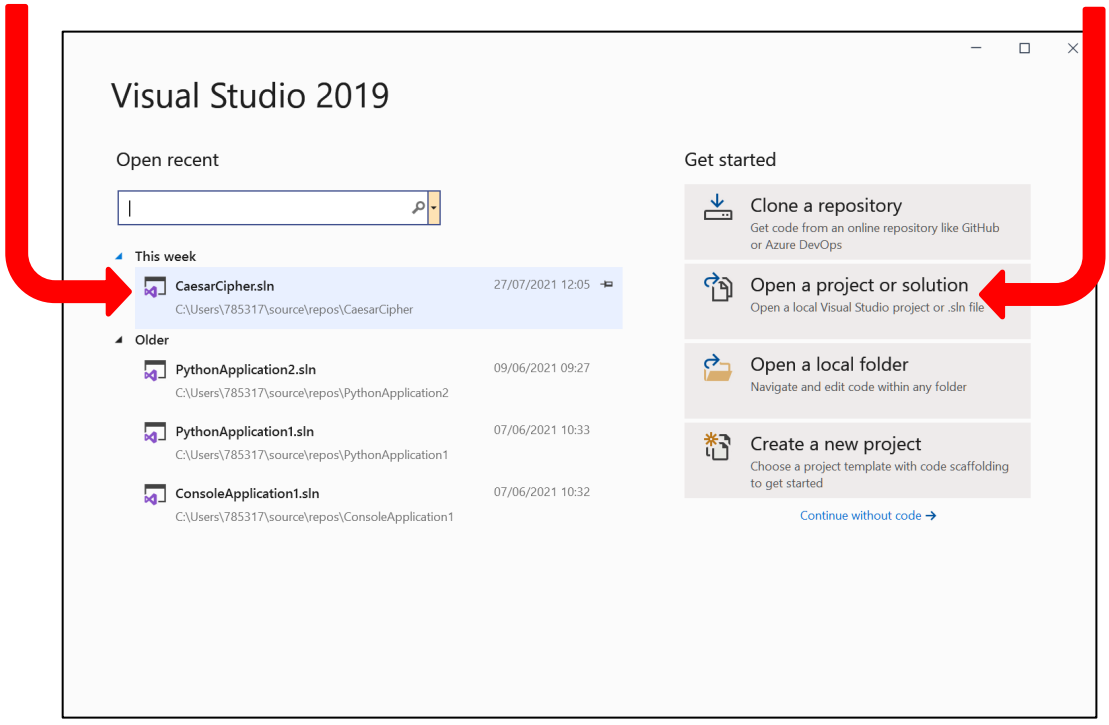
In the past Labs, we developed Caesar Algorithm to encrypt and decrypt messages. Using these methods we will develop a program that applies Brute Force Cryptanalysis on Caesar Algorithm.

Brute force attack aims to crack the cipher by revealing the key used in the cryptographic system. It is assumed that the attacker knows the underlying cipher is Caesar and s/he has captured a ciphertext. The aim is to find out the key using this ciphertext. It is carried out by decrypting the ciphertext with all possible keys. The obtained plaintexts are displayed row by row along with corresponding keys and examined if there is any meaningful plaintext in one of the rows (there may be more than one meaningful plaintext but the probability of such a case is very low in a real world scenario).

REMINDER:

1. Run `Visual Studio 2019`.

2. Open your `CaesarCipher` project using either **Open Recent** or **Open a Project or Solution** Tab.



3. If you have not finished the Caesar Cipher before, open "**`CaesarCipher - Cpp.txt`**" in the Lab folder and copy the missing code to your project.

4. Examine the `cpp` file to see how the Caesar algorithm works. The Alphabet is shifted according to the key determined by the user before encrypt or decrypt the string.

5. Run the program and see how it works. **USE ONLY CAPITAL LETTERS WITHOUT ANY BLANK SPACES BETWEEN THE WORDS. You can use Caps Lock to be sure that.**

   Tip: To run the program, use either `Debug -> Start Without Debugging` or `CTRL + F5`.

   Enter a plain text and key and observe the corresponding ciphertext and deciphered text from the ciphertext. **Is the deciphered text the same as the original plaintext you entered?**

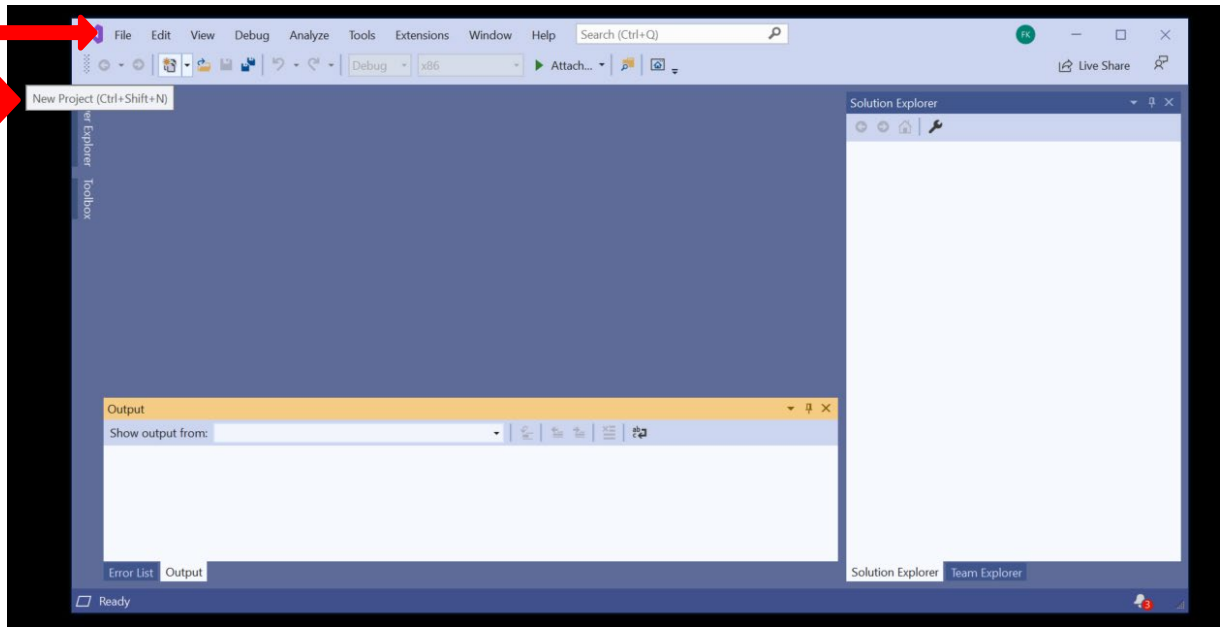
   Tip: Pencil down the plaintext, ciphertext and key. They will be used in the exercise.

PREPARATION:

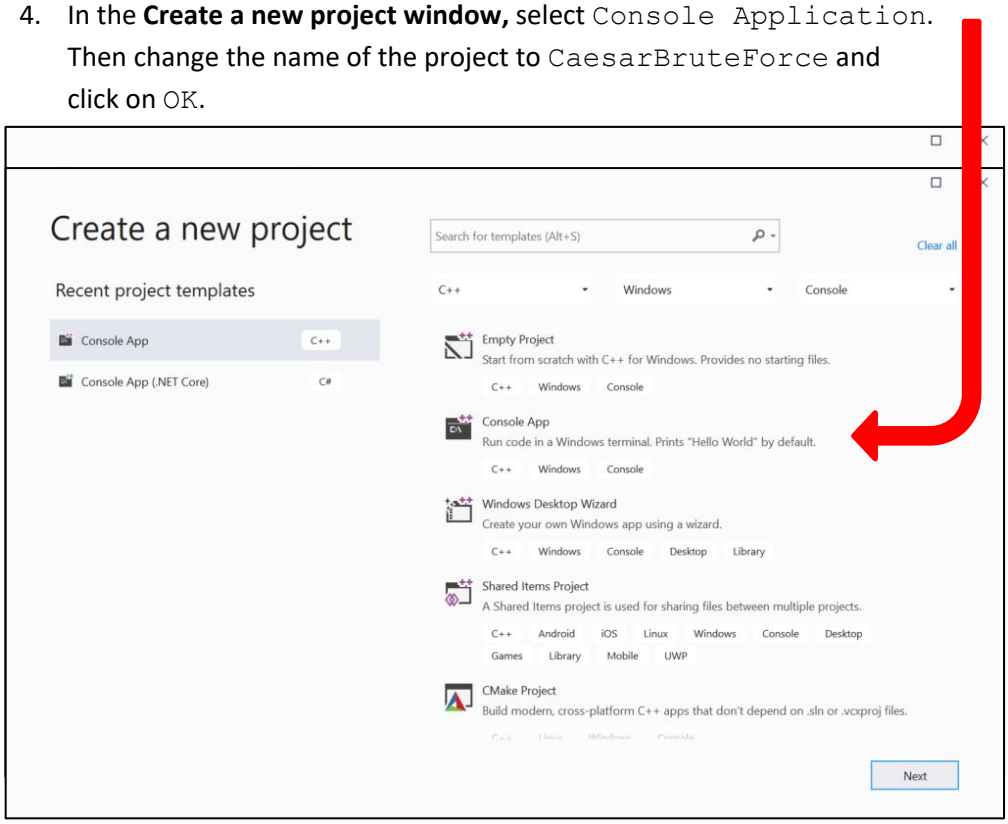
1. Run `CaesarCipher` again. Enter a plaintext and a key and observe the corresponding ciphertext. Pencil down the plaintext, the key and the ciphertext.

2. Close `CaesarCipher` using `File -> Close Solution`.

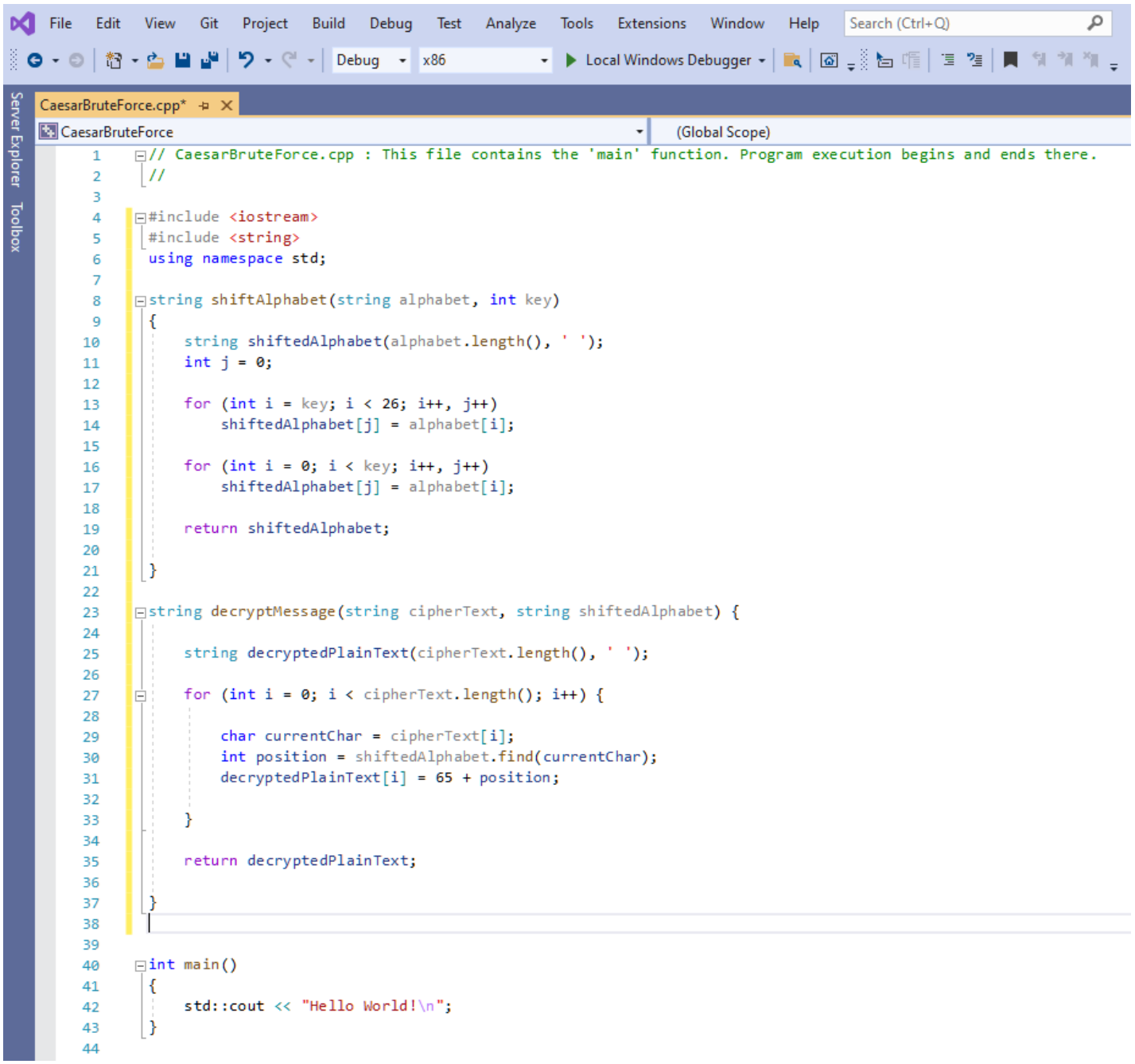
3. If the "What would you like to do?" window appears, click on the "Create a new project" tab. Otherwise, create a new project by clicking on `New project` or `Ctrl+Shift+N` icon the main screen or selecting `FILE -> New -> Project` in the menu.



4. In the **Create a new project window,** select `Console Application`. Then change the name of the project to `CaesarBruteForce` and click on `OK`.

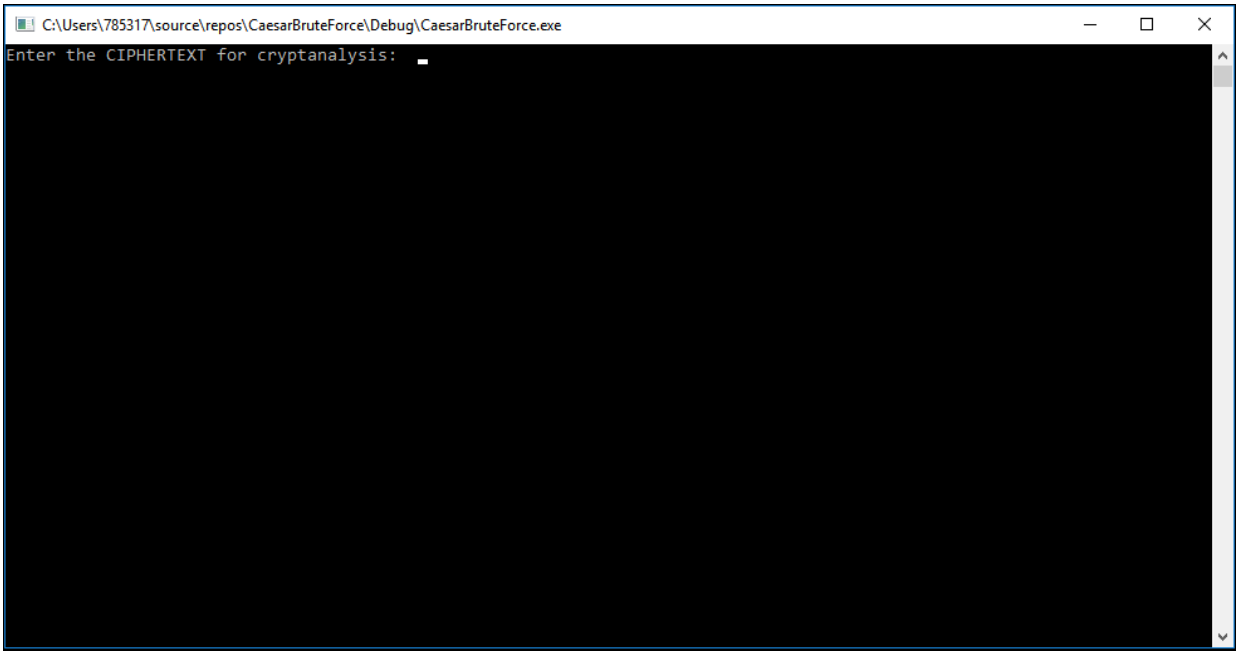


This will create an empty `main()` function for the project in which you will develop the brute force attack. You need to copy and paste `shiftAlphabet()` and `decryptMessage()` methods as well as their templates including `using namespace std;` and `#include <string>` from `CaesarCipher.cpp`. Your new `CaesarBruteForce.cpp` should be similar to the following:


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Debug
x86
Local Windows Debugger
Server Explorer
Toolbox
CaesarBruteForce.cpp*
CaesarBruteForce
(Global Scope)
// CaesarBruteForce.cpp : This file contains the 'main' function. Program execution begins and ends there.
//
#include <iostream>
#include <string>
using namespace std;
string shiftAlphabet(string alphabet, int key)
{
string shiftedAlphabet(alphabet.length(), ' ');
int j = 0;
for (int i = key; i < 26; i++, j++)
shiftedAlphabet[j] = alphabet[i];
for (int i = 0; i < key; i++, j++)
shiftedAlphabet[j] = alphabet[i];
return shiftedAlphabet;
}
string decryptMessage(string cipherText, string shiftedAlphabet) {
string decryptedPlainText(cipherText.length(), ' ');
for (int i = 0; i < cipherText.length(); i++) {
char currentChar = cipherText[i];
int position = shiftedAlphabet.find(currentChar);
decryptedPlainText[i] = 65 + position;
}
return decryptedPlainText;
}
int main()
{
std::cout << "Hello World!\n";
}

EXERCISE:

1. First, you need to ask the user to enter the ciphertext. The output of your program should be as follows:



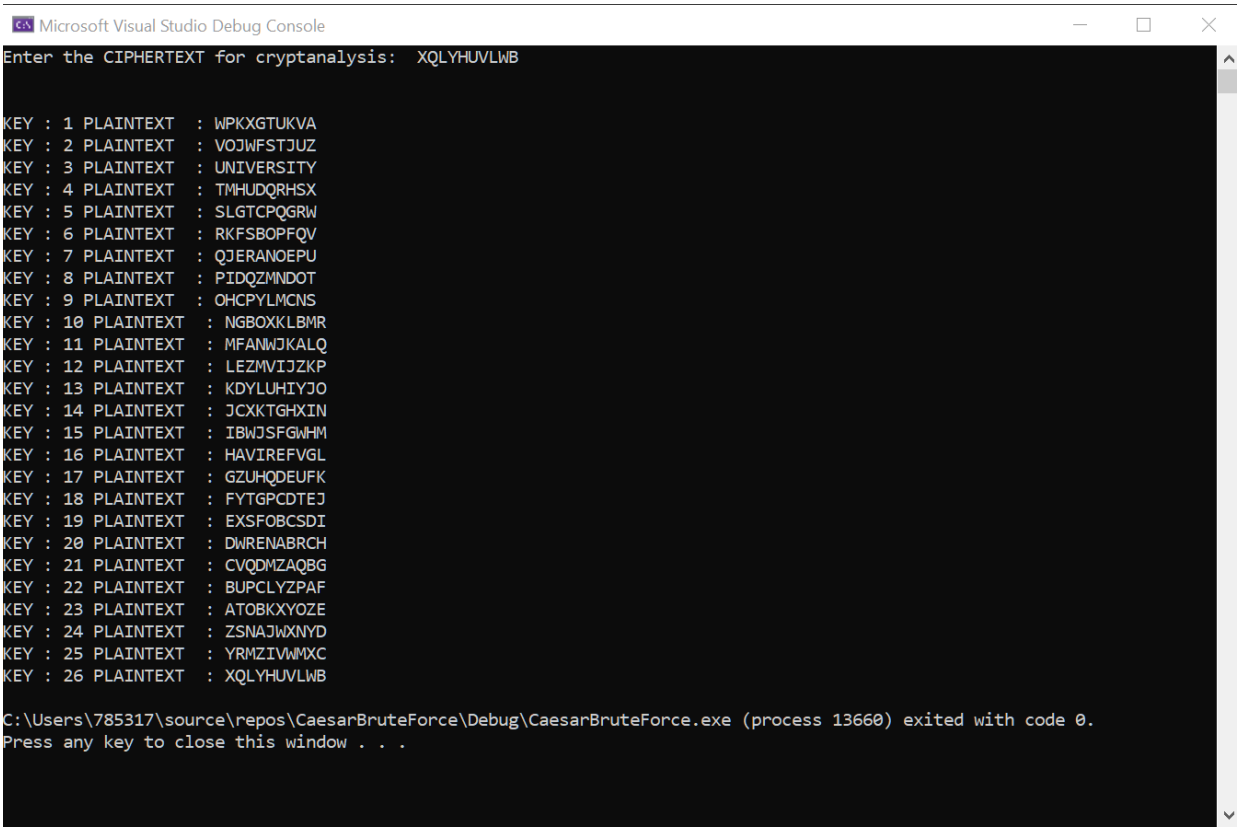
Tip: Use the ciphertext you recorded in the reminder section.

2. For all possible keys obtain plaintext using `decryptMessage()` and display the result using the following format:

```
KEY : 1 PLAINTEXT : <xxxxx>
KEY : 2 PLAINTEXT : <yyyyyy>
……
……
KEY : 26 PLAINTEXT : <zzzzzz>
```

Tips:
1. Use a `for` loop to iterate all keys.
2. Use the previous project to define variables for alphabet, shifted alphabet, plaintext, ciphertext and key.

The output of your code should be like that :



3. Check the displayed results and identify the plaintext. **Can you figure out the key for this ciphertext?**

4. Run the program again. Use the following ciphertext:

   `ZSNAJWXNYDTKIJWGD`

   What are the plaintext and the key?