

In this project we were to implement an encryption/decryption program that uses a Vigenère cipher to encrypt a user inputted message using a user inputted key. The Vigenère cipher uses a series of Caesar ciphers, each subsequent letter of the message uses a subsequent letter of the key to shift places. The project is broken into two parts. The first part simply had the user input the message and key words, and both must consist only of capital letters. In part two the message was to be read in from a text file and the encrypted message was written into another file. In addition, the program had to work with uppercase and lowercase letters, numerals, and symbols. Part two was to be executed from the command prompt and the user would specify whether they wanted to encrypt or decrypt, as well as specifying the files to be read from and written to.

1. How did you implement the encipher function in Part 1? Briefly discuss the Vigenère cipher and provide a copy of your function as part of your answer.

The function enciphers one character at a time and returns the enciphered character. It first checks to be sure that both the key letter and plaintext letter is a capital letter. If it is not it returns the plaintext letter as it is. If it passes that test, it applies the shift to the plaintext letter. To figure out the shift I took the key letter and subtracted the character 'A' to figure out how the number of characters for the shift. I then added the shift number to the plaintext character to arrive at the ciphertext character. If the shift took the plaintext character past 'Z' ASCII value, I figured out how far past 'Z' the ASCII value of the shifted character went and adds the difference between the character and 'Z' back to the beginning of the alphabet.

```

char encipher (char key, char plain )
{
    char encLetter; //holder for the letter as it gets encrypted

    if (plain > 'Z' || plain < 'A') //makes sure plaintext is a capital letter, returns it if not
        return plain;
    if (key > 'Z' || key < 'A') // makes sure key is a capital letter, returns it if not
        return plain;

    encLetter = plain + (key - 'A'); // adds the shift to the plaintext letter

    //checks if the shift takes it out of the range of the capital letters
    //64 is used as the ASCII value to roll over to the correct letter
    if (encLetter > 'Z')
    {
        encLetter = 64 + (encLetter - 'Z'); //rolls over out of range char back to 'A'
    }

    return encLetter;
}

```

2. How did you implement the decipher function in Part 1? Provide a copy of your function as part of your answer.

The decipher function works much the same as encipher function. However, it passes an encrypted letter and a key letter to it. It figures out the shift in the same way the encipher function figured out the shift. But, instead of adding it the plaintext letter it subtracts the shift from the ciphertext. It checks to see if the shift takes the letter less than the ASCII value of 'A'. If after the shift the character is less than 'A' the program figures out how far past 'A' the character is, and subtracts that from the ASCII value of 'Z' to essentially loop the letters back to the end of the alphabet.

```

char decipher ( char key, char cipher )
{
    char plain; // store the character as it gets deciphered back to plaintext

    if (cipher > 'Z' || cipher < 'A') //makes sure plaintext is a capital letter, returns it if not
        return cipher;
    if (key > 'Z' || key < 'A') // makes sure key is a capital letter, returns it if not
        return cipher;

    plain = cipher - (key - 'A'); //subtracts the shift from the encrypted letter

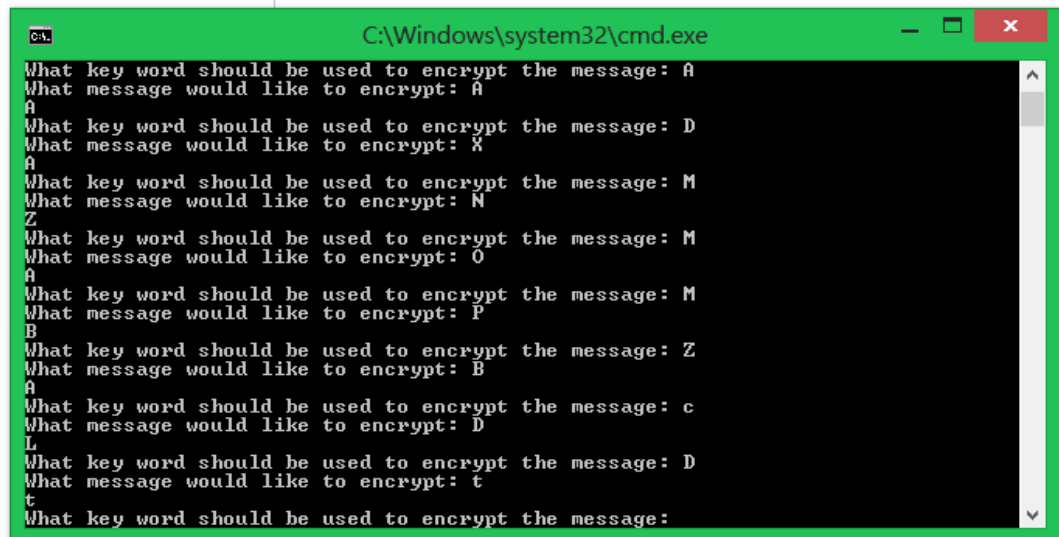
    //rolls letter less than 'A' back to the end of the alphabet 'Z'
    // ASCII value 91 is used to correctly roll letters over to the end of the alphabet
    if (plain < 'A')
        plain = 91 - ( 'A' - plain);

    return plain;
}

```

3. How did you verify the correctness of the two required functions in Part 1? Please include any test data used for this purpose.

To verify the correctness of the two functions in part 1 I ran several carefully chosen combinations of messages and keys. While I only had the encipher function written I simply went through the encryption process by hand to verify that the encryption process ran correctly. After writing the decipher function I ran the encipher function with the original message, and immediately deciphered the newly created encrypted message with the decipher function. If the deciphered message was the same as the original function I took that as a sign that the functions were correct. If this occurred I verified by performing both the encryption and decryption by hand to see if they were correct. I also tried to encipher and decipher single characters and carefully chose the key character so the enciphered character would land on a precise character. I tested both characters that wrapped around to the beginning of the alphabet and those that did not.



```
C:\Windows\system32\cmd.exe
What key word should be used to encrypt the message: A
What message would like to encrypt: A
A
What key word should be used to encrypt the message: D
What message would like to encrypt: X
A
What key word should be used to encrypt the message: M
What message would like to encrypt: N
Z
What key word should be used to encrypt the message: M
What message would like to encrypt: O
A
What key word should be used to encrypt the message: M
What message would like to encrypt: P
B
What key word should be used to encrypt the message: Z
What message would like to encrypt: B
A
What key word should be used to encrypt the message: c
What message would like to encrypt: D
L
What key word should be used to encrypt the message: D
What message would like to encrypt: t
t
What key word should be used to encrypt the message:
```

```
C:\Windows\system32\cmd.exe
What key word should be used to encrypt the message: L
What message would like to encrypt: P
P
What key word should be used to encrypt the message: Q
What message would like to encrypt: K
K
What key word should be used to encrypt the message: K
What message would like to encrypt: Q
Q
What key word should be used to encrypt the message:
```

4. How did you implement the cipher improvements asked for in Part 2? Provide code examples as appropriate.

The improvements to the cipher were relatively simply. Since all characters are assigned a numerical ASCII value, to implement the improvements I had to change the bounds of the characters used in the process. The lower bound instead of being 'A' is changed to 'SPACE', the lowest character ASCII value. 'Z' would get changed to '~' the highest character ASCII value. Between 'SPACE' and '~' are all capital letters, lowercase letters, numerals, and common symbols. The only problem that I ran into is when rolling characters that go over '~' back to the beginning, 'SPACE'. The character values could not go over '~' and the program would throw an error. To correct this I had to find how far the character went over '~' without actually finding out the ASCII value of it. I figured out the value of the shift and subtracted how far the character was from the end of the symbols. This is how far the letter would go over the end, and therefore be how far in from the start it would have to be. I did not have to go through this process for decryption because the lowest ASCII value used was 32, the value for 'SPACE'. I was able to go lower than this value so a special process was not required

From encryption function:

```
encLetter = plain + (key - ' '); // adds the shift to the plaintext letter

//returns the encrypted character if the shift does not need to be rolled over
//to the front of the table
if (encLetter <= '~' && encLetter >= ' ')
{
    return encLetter;
}

//handles the process of rolling over the characters to the front of table
//figures out how far the character should be from the front of the table
//takes how much the word should be shifted minus the distance the plaintext
//is from the end of the table, adds it to the front of the table of characters
rollover = (key - ' ') - ('~' - plain);
encLetter = 31 + rollover; //ASCII value 31 is used to appropriately assign characters
return encLetter;
```

From decryption function:

```
plain = cipher - (key - ' '); //subtracts the shift from the encrypted letter

//rolls character value less than ' ' back to the end of the table '~'
// ASCII value 127 is used to correctly roll letters over to the end of the alphabet
if (plain < ' ')
    plain = 127 - ( ' ' - plain);

return plain;
```