Programming Exercise 13
Encrypting and Decrypting Messages

C# Step by Step

1 Introduction

This programming exercise is a substantial assignment. The assignment is to write a program with will encrypt and decrypt messages. It has two sections, one to encrypt a message, and the second to decrypt the same message. Each part has three sections, each one more difficult than the last. Even though this is more difficult than your previous assignments, you have all you need to complete this successfully. Just be prepared to think very hard. The code is more difficult than in the previous exercises, but the thinking part is much harder.

Each letter of the alphabet has a numbered place, from 1 to 26. A = 1, B = 2, C = 3, and so on to Z = 26. Messages are encrypted by substituting each letter in the plain text message for another letter, which is determined by a key. Encrypted messages are decrypted by substituting each letter in the encoded message with the plain text key, again depending on the key. This is explained more fully below.

In the English speaking world, alphabetical letters are encoded by ACSII. You can look up the ASCII table to get the numerical code for each letter. For example, an uppercase "A" has the ASCII code 65, while the lower case "a" has the ASCII code 97. When letters are read by the computer, they are internally converted to the ASCII code. C# allows you to cast an alphabetical character to an integer, and cast an integer to a character provided that the integer is convertable to a character. In practice, this means that you are restricted to integers between 65 and 90 for upper case characters, and 97 to 122 for lower case characters.

Finally, users will enter invalid characters in their plain text message, such as blank spaces, numerical characters (we will not use numbers in this program), punctuation marks, etc. You will have to delete all these non-alphabetical characters. Part of the challenge is to "clean" the text the user enters by deleting all non-alphabetical characters (including digits, spaces, and punctuation) and convert all lower case characters to upper case characters.

2 Simple cipher

A very simple cipher entails the movement of each letter the number of places specified by a single letter key. For example, if the key were "C", each letter would be substituted for the third letter down from the place: A becomes D, B becomes E, C becomes F, W becomes Z, X becomes A, W becomes B, and Z becomes C. Notice that the letters at the end of the alphabet wrap around to the beginning. If the message was "We attack at dawn.", and the key was "C", the encoding would be done like this:

Plain	W	E	A	T	T	A	C	K	A	T	D	A	W	N
Key	C	C	C	C	C	C	C	C	C	C	C	C	C	C
Encoded	Z	H	D	W	W	D	F	N	D	W	G	D	Z	Q

Using numbers, the result would be:

ASCII	87	69	65	84	84	65	67	75	65	84	68	65	87	78
Key	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Code	90	72	68	87	87	68	70	78	68	87	71	68	90	81

The first line represents the ASCII values of the upper case alphabetical characters. The second line is the key, "C" using the nominal numerical value of "3". The third line is the coded message. Obviously, you will have to convert back and forth between the ASCII values and the nominal numerical value, that is, converting 65 to 1, 66 to 2, 67 to 3, and so forth, wrapping around to the beginning as necessary.

Decrypting is the opposite, taking the third line, adding the key, and displaying the plain text. If you can write the first part, you can write the second part, if you think hard enough about it.

3 Less simple cipher

This is the same as the first part, but instead of using just one charcter as a key, you will use a word. In this example, the key is CAT. If the key is shorter than the message, you repeat the key as often as necessary. If you wrote the first part, you can write the second part, but it's more difficult.

Plain	W	E	A	T	T	A	C	K	A	T	D	A	W	N
Key	C	A	T	C	A	T	C	A	T	C	A	T	C	A
Encoded	Z	F	U	W	U	U	F	L	U	W	E	U	Z	O

Here it is with numerical values:

ASCII	87	69	65	84	84	65	67	75	65	84	68	65	87	78
Key	3	1	20	3	1	20	3	1	20	3	1	20	3	1
Code	90	70	85	87	85	85	70	76	85	87	69	85	90	79

4 Still less simple cipher

This part has a key that's a word, but the word isn't repeated. Instead, the plain text becomes part of the key. Look at the example below. Notice that the second line, the encryptiion key, is "CATWEATTACKATD." This technique makes the encrypted message much harder to crack.

Plain	W	E	A	T	T	A	C	K	A	T	D	A	W	N
Key	C	A	T	W	E	A	T	T	A	C	K	A	T	D
Encoded	Z	F	U	Q	Y	B	W	E	B	W	O	B	Q	R

Here it is with numerical values:

ASCII	87	69	65	84	84	65	67	75	65	84	68	65	87	78
Key	3	1	20	23	5	1	20	20	1	3	11	1	20	4
Code	90	70	85	81	89	66	87	69	66	87	79	66	81	82

Notice that in the fourth column, the plain text letter is "T" and the key is "W". 84 (representing T in ASCII) and 23 (representing W in ASCII) resulting in 107 (84 + 23), which does not give the correct value. In order to wrap this character around, it's necessary to subtract 26 from 107, which gives 81, the ASCII value of "Q".

Here is my listing. You can use this as a starter. Note that you will have to write all the methods in class `Util`.

```
1  class Program
2  {
3      static void Main(string[] args)
4      {
5          string plain_text = Util.GetPlainText();
6          string single_key = Util.GetSingleKey();
```

```
7           string multi_key = Util.GetMultiKey();
8           Console.WriteLine();
9
10          Console.WriteLine($"You entered [{plain_text}] as plain text");
11          Console.WriteLine($"You entered [{single_key}] as your single key");
12          Console.WriteLine($"You entered [{multi_key}] as your multi key");
13          Console.WriteLine();
14
15          int[] clean_text = Util.Clean(plain_text);
16          int[] clean_skey = Util.Clean(single_key);
17          int[] clean_mkey = Util.Clean(multi_key);
18
19          string enc_single = Util.SingleEnc(clean_text, clean_skey);
20          string enc_multi = Util.MultiEnc(clean_text, clean_mkey);
21          string enc_conti = Util.ContiEnc(clean_text, clean_mkey);
22
23          Console.WriteLine($"Encrypted message with single key is [{enc_single}]");
24          Console.WriteLine($"Encrypted message with multi key is [{enc_multi}]");
25          Console.WriteLine($"Encrypted message with continuous key is [{enc_conti}]");
26          Console.WriteLine();
27
28          string dec_single = Util.SingleDec(enc_single, clean_skey);
29          string dec_multi = Util.MultiDec(enc_multi, clean_mkey);
30          string dec_conti = Util.ContiDec(enc_conti, clean_mkey);
31
32          Console.WriteLine($"Decrypted message with single key is [{dec_single}]");
33          Console.WriteLine($"Decrypted message with multi key is [{dec_multi}]");
34          Console.WriteLine($"Decrypted message with continuous key is [{dec_conti}]");
35          Console.WriteLine();
36
37      }
38  }
```

Figure 1 contains an example run of the above program, with all the utility methods completed.

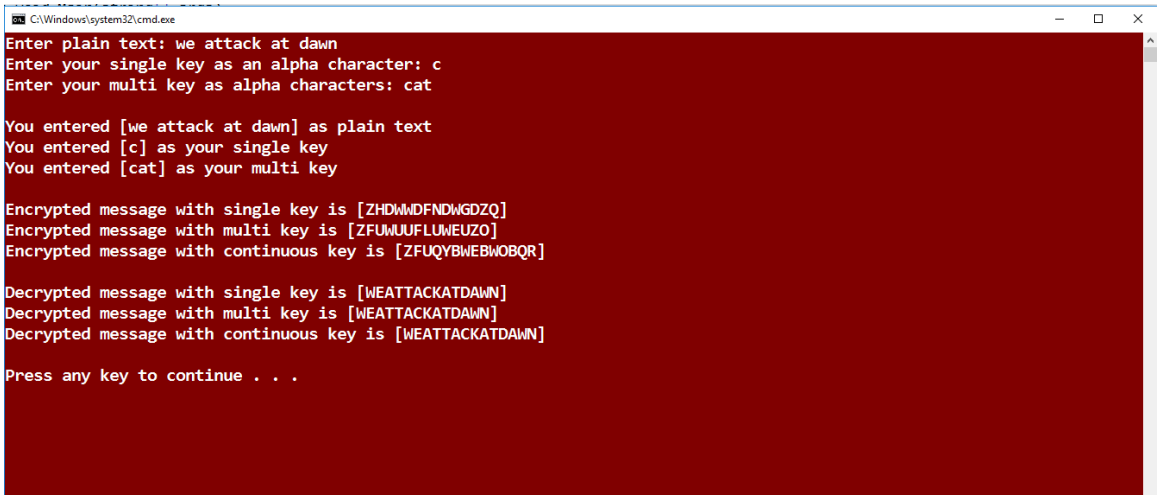

Figure 1: Example output for encryption/decryption program

5 Scoring

Scoring for this exercise is as follows:

50 points Single character key encryption

60 points Single character key decryption

70 points Multi character key encryption

80 points Multi character key decryption

90 points Continuous character key encryption

100 points Continuous character key dencryption

6 Hints

Do not hard code any input. The message to encrypt and the key will be entered by the user at the command prompt. The encrypted string to decrypt and the key will also be entered by the user at the command prompt. In my example, I used the same program for both parts, which allows me to save the user input and reuse the input as necessary. In practice. you would use one program for encryption and a second program for decrytion.

The first thing you should do is to encrypt some short messages, and write down your steps. Essentially, you will instruct the computer to follow the same steps. In a way, all you are doing is automating a manual process. This may be hard, but it isn't magic.

You know the software process: specify the requirements, complete a design, implement the design, and test the implementation. This project is not a trivial project, but one that will stretch your abilities and build your strength. Remember to think this through before writing any code. Write one little piece at a time, testing it to see if it does what it's supposed to do. Work carefully. You will have fun, and be proud that you can write a program that does something useful. Just don't tell the F.B.I. what you are doing, because encryption software is considered a munition, and export is prohibited.