Submit <u>one</u> zip or rar file named as yourlastname_yourfirstname_moo. It should include the following files:

- An executable of your program
- Source code in one file as a .c, .cpp or a .java
- Input and output file that you used to test your file
- A detailed report on your assignment

CIS 3360 – Security in Computing
Summer 2011
Programming Assignment (100 points)
Bonus Assignment (50 points)

In this programming assignment you are going to implement the problem you solved in assignment 4 problem 6 using modified CBC as described in the assignment. The assignment problem is restated with modified CBC making it suitable for programming assignment. Assume that our encryption method is Vigenère Cipher with four characters key provided by input. Our hardware can encrypt four characters at a time. We want to send messages using modified CBC mode. In this modified CBC change all exclusive operations of CBC by addition and then mod 26. Assume that the value of Key is read as an input of four alphabetic characters. You should check the validity of the input. Your program should reject the invalid input and give a message and then ask to provide valid input again. Assume that value of IV is read as an input of four alphabetic characters. You should check the validity of the input. Your program should reject the invalid input and give a message and then ask to provide valid input again. Now read a message of alphabetic string of maximum 100 characters. If the message is not a multiple of four characters, pad the input message with character Z at the end of the message by 1 to 3 characters to make the message a multiple of four characters. Show the encrypted message output of the input message (multiple of four characters) by using CBC. Now decrypt the message to show that you get your message back. For decryption, you will subtract (instead of addition) and then take mod 26.You need to see modes of operation Tegrity lecture or modes of operation slides to work on this problem.

Here is an example given in the solution of assignment 4.

<u>CBC</u>
Input for Key is the string : FOUR (coded as - 05 14 20 17)
Input for IV is the string : I N I T (coded as - 08 13 08 19)
Initial [08 13 08 19]
Key [05 14 20 17]

Message	[12 14 03 04]	[18 14 05 14]	[15 04 17 00]	[19 08 14 13]
<u>Add</u>	<u>[08 13 08 19]</u>	<u>[25 15 05 14]</u>	<u>[22 17 04 19]</u>	<u>[16 09 15 10]</u>
Result	[20 01 11 23]	[17 03 10 02]	[11 21 21 19]	[09 17 03 23]
<u>Key</u>	<u>[05 14 20 17]</u>	<u>[05 14 20 17]</u>	<u>[05 14 20 17]</u>	<u>[05 14 20 17]</u>
Vigenère	[25 15 05 14]	[22 17 04 19]	[16 09 15 10]	[14 05 23 14]
Encrypted	[ZPFO]	[WRET]	[QJFK]	[OFXO]

Message **zpfow re tqjfkofxo**

Decryption

Encrypted Message:	[25 15 05 14]	[22 17 04 19]	[16 09 15 10]	[14 05 23 14]
Subtract Key	[05 14 20 17]	[05 14 20 17]	[05 14 20 17]	[05 14 20 17]
Result	[20 01 11 23]	[17 03 10 02]	[11 21 21 19]	[09 17 03 23]
Subtract init/Cipher	[08 13 08 19]	[25 15 05 14]	[22 17 04 19]	[16 09 15 10]
Message	[12 14 03 04]	[18 14 05 14]	[15 04 17 00]	[19 08 14 13]

Bonus Problem

Create a check sum of four characters on the code using add and then mod 26 on each character. Create a 26's complement of checksum and add it at the end of the encrypted message. The 26's complement of a number X between 0 and 25 is defined as (26 – X) mod 26. For example if X is zero, 26's complement of X is also zero. If X is 5, 26's complement of X is 21.

When you decrypt the message, you first check the checksum of the message. If the result is all zero, then the message is okay. Decrypt the message except the last four character of the encrypted message.

Example continued

Encrypted Message: [25 15 05 14] [22 17 04 19] [16 09 15 10] [14 05 23 14]

Check sum generation

Code of first four characters	[25 15 05 14]
Add next four characters	[22 17 04 19]
Result	[21 06 09 07]
Add next four characters	[16 09 15 10]
Result	[11 15 24 17]
Add next four characters	[14 05 23 14]
Result	[25 20 01 05]
26's complement	[01 06 25 21]

Transmitted message: [25 15 05 14] [22 17 04 19] [16 09 15 10] [14 05 23 14] [01 06 25 21]

Checking for Checksum

If you add first four blocks as shown above, you will get: [25 20 01 05]

Now you add [01 06 25 21]. You will get [00 00 00 00]

What to submit:

Submit an archive file (zip or rar) with the name yourlastname_yourfirstname_crc. Include the following three files in your archive file:

- An executable of your program
- Source code in one file as a .c, .cpp or a .java
- Input and output file that your created to test your program
- A detailed report on your assignment

Report format:

In your report, include the following information:

- Description of the program's functionality. Explain each functional code block.
- Include any missing functionality or erroneous behavior
- Description of input to the program and output of the program
- What language was used and what compiler was used
- One test file that you used and output from that test file for a sample run of your program

Input format:

Your program should prompt the user for the following input:

Enter the alphabetic key (4 character) for encryption:
Enter four value alphabetic character IV:
Enter the name of input file with message M:
----------Menu----------

1. Encrypt M
2. Decrypt encrypted message
3. Encrypt M and add checksum
4. Determine checksum and decrypt encrypted message
5. Exit

Choose from the above menu:

You must check for input correctness also, by implementing code to:

- Verify that user has input a valid option in the menu. If invalid, prompt for menu again.
- Verify that the file that the user has entered actually exists. If the file does not exist, prompt for file again.
- Verify that each character in the input file is a valid alphabetic character

Output format:

Print out the key value (four alphabetic characters) given as input.
Print out the IV value (four alphabetic characters) given as input.
The input file will contain data in alphabetic character. Print out the alphabetic file that is read.
Next, you will encrypt the file with the key and IV using 16 characters at a time.

Determine number of message segments for the message, you can name them segment1, segment2 etc.
Print all segments with their name first.
You must print the encryption process as shown in the example for each segment of message (16 characters) in one line.
Provide the encrypted message in alphabetic form
Output a file with key value, one space, IV value, one space, and encrypted message in mod 26 code (you select how you want to store this code and write in your report).
Use this file for decryption.
For bonus problem, create the checksum at the end. Add the check sum to the encrypted message and output a file as before with the checksum at the end.
Use this file to check for checksum and then decrypt the message.