Program Submission Instructions:

- You must submit one archive file in Webcourses.
- Your archive file must be named: **yourlastname_yourfirstname_CRC**
- Acceptable file format for the archive file: .zip, .tar, .gz, .bz
- Your submission must contain all of the items described in the "What to submit" section below.
- Acceptable file format for your report: .doc, .docx, or pdf format

CIS 3360 – Security in Computing
Fall 2013
Program #2: CRC Codes (100 points)

Write a program that calculates the CRC-16 value for a given file and also verifies the correctness of a given file that already has CRC-16 value appended at the end, as more fully described below.

Use the CRC polynomial $x^{16} + x^{11} + x^8 + x^7 + x^3 + x^2 + 1$.

Program operation:

1. The program must compile from the command line.
2. The program executable file name must be "**crcfile**".
3. The program must run from the command line and take two (2) command line parameters.
4. The first command line parameter will be a flag value that identifies the the mode of operation: "c" for calculating a CRC value, or "v" for verifying a CRC value. Only these two values are allowed. Any other values should produce a simple error message and a graceful exit from the program.
5. The second command line parameter will be the name of the file to be examined. The file should be a text file that is in the same folder as the program executable. If the file is not found, the program should issue a simple error message and exit gracefully.
6. The program should direct all output to the command window (terminal) screen. The details of what to output are described below.

What to submit:

Submit an archive file (zip, tar, gz, or bz) with the name "**yourlastname_yourfirstname_crc**." Include the following files in your archive file:

- An executable of your program

- Source code in one file as a .c, .cpp or .java file
- A sample test input file that is a valid input file (see "Input file format" section below) and contains at least 40 characters.

  NOTE: You must include a test input file. If you merely include the input file contents in your report (which would require a grader to create a file and paste your input into it in order to run your program against the test input), there will be a **5-point deduction**.

- A detailed report on your assignment, as described below.

Report format:

You must include in your submission a report in .doc, .docx, or pdf format. The report must contain the following information:

- Your name
- Identify the programming language and development environment that were used
- Description of the program's functionality. Identify each functional component and explain what it does.
- Explain any missing functionality or erroneous behavior
- State the exact command that a grader should use to compile your program from the command line.
- State the exact command that a grader should use to run your program from the command line.
- Include all of the output from running your program in both modes against the test file that you have included with your submission.
- Your statement that the program is entirely your own work and that you have neither developed your code together with any another person, nor copied program code from any other person, nor permitted your code to be copied or otherwise used by any other person, nor have you copied, modified, or otherwise used program code that you have found in any external source, including but not limited to, online sources.

  NOTE: If you need assistance in understanding the assignment or any necessary skill needed for the assignment, please contact any of our TAs or your instructor.

Input file format:

Valid input files will be text files that contain only hexadecimal characters. Either upper or lower case letters may appear, but they will only be letters that correspond to hexadecimal characters. There will be no whitespace, punctuation, or special characters in the input files.

You must check for input correctness by implementing code to verify that each character in the input file is a valid hexadecimal character (0,1,2,3,4,5,6,7,8,9A,B,C,D,E,F,a,b,c,d,e,f). Valid upper and lower case hex letters will be given the same meaning ("a" = "A" = 1010, etc.)

If an input file contains any invalid characters (other than an end-of-file marker), then the program should issue an appropriate brief error message and terminate gracefully.

Output format:

Print out the hexadecimal file that is read, 80 hex characters to a line.

Next, you must convert the hexadecimal file into binary. Recall that one hex character is four bits. Read the input file and form a binary string. The length of the binary string will be four times the length of the input file because each hexadecimal character will be replaced by 4 binary bits. You will have to define a separate function to get the binary equivalent of a hexadecimal character.

You must state how many zeroes are appended to the input. You must then print the binary version of the input file (after appending) 64 binary bits per line. You must also insert a space after every four binary bits that you print (increasing the line length to 80 characters).

Next, you will show the result of each XOR operation involved in the CRC calculation or verification. The result value at each step is a binary value that contains all of the digits of the dividend after an XOR operation. For example, the first table below is from our Exam #2 review slide set, in which we used the polynomial $x^3 + x^2 + 1$ to calculate the CRC-3 code for the message 5AE, with the divisor shown in red and the dividend in blue. The red remainder 100 is the CRC-3 code for this example.

	1	1	0	1	|	0	1	0	1	1	0	1	0	1	1	1	0	0	0	0
⊕							1	1	0	1										
								1	1	0	0									
⊕								1	1	0	1									
											1	1	0	1						
⊕											1	1	0	1						
															1	1	0	0		
⊕															1	1	0	1		
																		1	**0**	**0**

The second figure is the same calculation, with the desired output lines completed and in bold. Note the leading zeroes after each XOR and trailing values brought down from the remaining dividend.

	1	1	0	1	|	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**
⊕							1	1	0	1										
						0	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**
⊕								1	1	0	1									
						0	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**
⊕											1	1	0	1						
						0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**
⊕															1	1	0	1		
						0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**

At the end, when calculating CRC, you must show the CRC answer in binary, then convert it to hexadecimal and print out both the binary and hex values for the CRC. Please note that you must pad the remainder with leading zeroes as needed to obtain a 16-bit value for the CRC-16 code.

On the other hand, if verifying CRC, you must print (a) the CRC value (both binary and hexadecimal) observed in the input file; (b) the CRC (both binary and hexadecimal) calculated by the program; and (c) a message whether the CRC check passed or failed.

Specific Functions inside code:

You must implement the following functions/methods:

- A function/method for converting a hexadecimal string into binary string
- A function/method for converting a binary string to hexadecimal
- A function/method to validate the characters in the input file
- An XOR function/method that takes as input two binary strings and returns the XOR result.
- A function/method for CRC calculation
- A function/method for CRC verification

Sample Output from Operating Mode "c" : Calculate CRC

As an example, suppose we wish to calculate the CRC-8 using the polynomial $x^8+x^7+x^6+x^5+x^4+x^3+x+1$.

And suppose also that the input file has the following contents: AB1245

Then, the CRC calculation output would be:

The input file (hex): AB1245
The input file (bin):

1010 1011 0001 0010 0100 0101

The polynomial that was used (binary bit string): 1 1111 1011
We will append eight zeros at the end of the binary input.

The binary string answer at each XOR step of CRC calculation:

1010 1011 0001 0010 0100 0101 0000 0000

0101 0110 10**01 0010 0100 0101 0000 0000**

0010 1000 010**1 0010 0100 0101 0000 0000**

0001 0111 0011 **0010 0100 0101 0000 0000**

0000 1000 1000 0**010 0100 0101 0000 0000**

0000 0111 0101 10**10 0100 0101 0000 0000**

0000 0000 1011 0110 0**100 0101 0000 0000**

0000 0000 0100 1011 11**00 0101 0000 0000**

0000 0000 0011 0101 000**0 0101 0000 0000**

0000 0000 0000 1010 0110 0**101 0000 0000**

0000 0000 0000 0101 1011 11**01 0000 0000**

0000 0000 0000 0010 0101 000**1 0000 0000**

0000 0000 0000 0001 1010 0111 **0000 0000**

0000 0000 0000 0000 0101 1100 00**00 0000**

0000 0000 0000 0000 0010 0010 110**0 0000**

0000 0000 0000 0000 0001 1101 1010 **0000**

0000 0000 0000 0000 0000 0010 0001 000**0**

0000 0000 0000 0000 0000 0001 1110 0110

0000 0000 0000 0000 0000 0000 0001 1101

The computed CRC for this file is 0001 1101 (bin) = 1D (hex)

Remember that this is just an example using CRC-8. In the programming assignment you must compute CRC-16 codes using a different polynomial.

Sample Output from Operating Mode "v": Verify CRC

As an example, consider that you have to verify the CRC-8 of an input file using the polynomial $x^8+x^7+x^6+x^5+x^4+x^3+x+1$

And assume the input file has the following contents: AB12451D

Then, the CRC verification output would be:

The input file (hex): AB12451D
The input file (bin):

1010 1011 0001 0010 0100 0101 0001 1101

The polynomial that was used (bin): 1 1111 1011
The 8-bit CRC observed at the end of the file: 1D (hex) = 0001 1101 (bin)

The binary string answer at each XOR step of CRC verification:

1010 1011 0001 0010 0100 0101 0001 1101

0101 0110 10**01 0010 0100 0101 0001 1101**

0010 1000 010**1 0010 0100 0101 0001 1101**

0001 0111 0011 **0010 0100 0101 0001 1101**

0000 1000 1000 0**010 0100 0101 0001 1101**

0000 0111 0101 10**10 0100 0101 0001 1101**

0000 0000 1011 0110 0**100 0101 0001 1101**

0000 0000 0100 1011 11**00 0101 0001 1101**

0000 0000 0011 0101 000**0 0101 0001 1101**

0000 0000 0000 1010 0110 0**101 0001 1101**

0000 0000 0000 0101 1011 11**01 0001 1101**

0000 0000 0000 0010 0101 000**1 0001 1101**

0000 0000 0000 0001 1010 0111 **0001 1101**

0000 0000 0000 0000 0101 1100 00**01 1101**

0000 0000 0000 0000 0010 0010 110**1 1101**

0000 0000 0000 0000 0001 1101 1011 **1101**

0000 0000 0000 0000 0000 0010 0000 110**1**

0000 0000 0000 0000 0000 0001 1111 1011

0000 0000 0000 0000 0000 0000 0000 0000

The computed CRC for this file is 0001 1101 (bin) = 1D (hex)

Did the CRC check pass? (Yes or No): Yes

Remember that this is just an example using CRC-8. In the programming assignment you must compute CRC-16 codes using a different polynomial.