Project 1: CRC Experiments

This project MUST be completed individually.

1. PROJECT GOALS

Project 1: Write a program (in C++) for a sender that takes a bit stream and computes several experiments to do with CRCs. Whichever language you choose to use, you should provide a Makefile consistent with the project submission guidelines below. The input to the program should be a string (which you read from the command line) containing ASCII 0's and 1's that represents the message to be check summed. Your output should be a 16-bit CRC for the CRC-16 polynomial $x^{16} + x^{12} + x^7 + x^5 + 1$. Please read the hints also on the class web page.
This project will be due on Sunday November 7th, at 11:55 PM PST.

2. PROGRAMMING ENVIRONMENT

You should test and develop your code on the SEASnet system, as your submission will be compiled and graded on this environment.
On SEASnet, you can use Vim, Emacs, or Nano to edit your source code and any compiler you'd like to build it. However, you must submit a Makefile that can be used to compile your code into an executable named "crcExperiments".

3. INSTRUCTIONS

(1) If you haven't already done so, familiarize yourself with the basics of CRCs. An overview can be seen in the lecture materials or in the textbooks.
(2) Create a program that named `crcExperiments` that accepts an arbitrary bitstring (formatted like the following without surrounding quotes: "10101010101000001111110000001") as an argument, and, given the correct command line arguments does one of the following: computes the CRC, checks if the CRC is correct, computes and outputs all undetected 4-bit errors, and computes and outputs the fraction of undetected 2 bit errors
   Requirements:
   - The program must accept a string as argument provided in the format `-c [string representing bits, e.g. 01010101]`. It must then output the correct bitstring WITH the attached CRC to stdout, using the generator $x^{16} + x^{12} + x^7 + x^5 + 1$.
   - The program must accept a string (that contains a CRC) as argument provided in the format `-v [string representing bits, e.g. 01010101[...]1]`. It must then validate whether or not the string is consistent with the attached CRC, using the generator $x^{16} + x^{12} + x^7 + x^5 + 1$. In the case of a valid input, the program should output a '1' to stdout, and if invalid, should output a '0'.
   - The program must accept a string as argument provided in the format `-f [string representing bits, e.g. 01010101]`. The input string will include a message without a CRC. It must then output all undetected 4 bit errors to stdout, with newline characters between each error. Errors should take the form of the initial string of bits WITH the error added in. The output should include the CRC (possibly modified).
   - The program must accept a string as argument provided in the format `-t [string representing bits, e.g. 01010101]`. It must then output the fraction of undetected two bit errors to stdout. As above, the input will be a message without a CRC. You will only be graded on the first two significant figures of your answer.
   - If no argument is specified, your program must exit with return code 2 and print nothing to stdout.

   We strongly recommend you use `getopt()` to parse command line options.

4. Grading Criteria

This is an **individual project**, meaning that no collaboration is allowed. You are allowed to use online resources to understand how to use CRCs; however, you must not copy code from the Internet and must credit any resources used in comments contained in your source code.

Your code will be graded based upon the following criteria:

- Whether your Makefile compiles your code properly on the SEASnet systems.
- Whether you included a file named "README" that contains your name on the first line and your UID on the second line.
- Whether your program runs on SEASnet without any errors or segfaults.
- Whether your program is able to find the value for the CRC of an arbitrary frame, and be able to check an arbitrary frame, given the appropriate command line input.
- Whether or not the program is able to find all possible 4 bit random errors that can occur in the frame. It should send this to stdout in a string of 0's and 1's (e.g. '101010101'). The string printed should be the modified version of the bit string provided (e.g. the string with the error added in).
- Whether your program can output the fraction of undetected 2 bit errors in the above case.

5. Project Submission

Put all your files into a directory and compress the contents of this directory into a file named "UID.tar.gz" (replacing UID with your UCLA ID). You **MUST** put all your files directly at the root of this archive (and not inside a directory) to ensure your code is graded properly.

Please submit your project via CCLE – submissions via any other method will not be accepted.

Your submission should include the following:

- Your source code.
- A Makefile that builds your code. This should create an executable named `crcExperiments` in the same directory when when one types "make".
- A file named "README" containing your name on the first line and your UID on the second line.

Acknowledgements

This project was adapted and modified from an earlier instance from George Varghese.