CNT5505 Programming Assignment No. 1: CRC checksum
(This is an individual programming assignment. Due October 11)

Purpose

- Implement a routine to compute CRC checksum
- Practice processing frames

Description

In this project, you will be given a dump file containing an unknown number of frames. The framing is done using a byte-oriented framing scheme with FLAG bytes and byte stuffing. Each frame starts and ends with a FLAG byte. To perform byte stuffing, an ESC byte is defined. When an ESC or FLAG byte occurs in the user data or checksum, an ESC is inserted before the special character for the escape. Each flame contains at least one byte of user data and 2 bytes of checksum. FLAG is the ASCII character 'a' and ESC is the ASCII character 'b' in our system. The format of the frame is as follows with user_data and CRC_checksum being byte-stuffed

FLAG	user_data	CRC_checksum	FLAG

The dump file contains a sequence of such frames, which may or may or be valid. Your job is to identify valid frames, remove invalid frames, and re-assemble the original data in the valid frames. The CRC checksum is computed with the following generator polynomial

$$x^{16}+x^{14}+x^{13}+x^8+x^4+x+1$$

You should implement your routine to allow flexible generator polynomial. The channel supports two bit stream. So the CRC checksum is not computed in the conventional way in that the bit order are different. In particular, for each byte that has 8 bits from the most to the least significant bit: $b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0$, our channel sees the bits in the following order when computing the checksum: $b_7, b_3, b_6, b_2, b_5, b_1, b_4, b_0$. This order affects how the CRC checksum is calculated.

Grading Policy

You can write the program in any language that is available on linprog, where your program will be graded. You must supply a README file describing how to run your program.

- Program compiles with no compiler error (10 points) -- programs with compiling error, get 0 points.
- README file, make file, etc (10 points)
- Program can correctly handle all cases with 1-byte user data in a small file (40 points)
- Program can correctly handle all cases with 1-byte and 2-byte user data in a small file(10 points)

- Program can correctly handle all cases with no more than 3 bytes user data in a small file (10 points)
- Program can correctly handle files of any size and any random size frames (20 points)
- Being the first one to report a bug in the sample code and/or the dump file (5 extra points per bug).

Deadline and materials to be submitted

- Submit all files including readme and makefile through campus.fsu.edu. Mr. Mehran (our TA) may contact you to setup a demo.

Miscellaneous

All programs submitted will be checked by a plagiarism detection software. Honor code policy will be strictly enforced. Write the code by yourself and protect your code.

The assignment is small, but you must be comfortable with manipulating bits and bytes, and dealing with non-ASCII files.

Some sample testing files will be provided to you for your convenience. But thorough testing will be made in grading the assignment -- so you should do more testing beyond the files given.