Problem 1 -- Canonical mode processing

Write a program to demonstrate canonical mode input processing. Your program should put the tty (assume that stdin is a tty) in non-canonical mode, then loop reading characters and processing them, applying the local editing functions of erase, word erase and line kill. You also handle echo, including proper echo of the 3 erase functions above.

The structure of your program will be a function `my_lineread` (or similar name) which blocks until an entire line or end-of-file is read. Your code should be bulletproof wrt backspacing beyond the beginning of the line or keying in an absurdly long line (2048 characters is a reasonable line length limit). Your function emulates `read`, so it returns the number of characters in the line, and does not NUL terminate.

Encapsulate your function with a simple test main that reads each line of input, spits it back to stdout, and exits at EOF. Use the pre-existing values of VERASE, VWERASE, VKILL and VEOF for your erase, word erase, kill (line erase) and eof characters.

Make sure to restore the original tty settings! If you don't, or your program terminates abruptly, you may need to type ^Jstty sane^J to regain control. This probably won't echo, don't worry. You need Control-J, not just ENTER, because ENTER is really Control-M. Normally the tty canonical mode translates this (the ICRLF flag).

You should set VTIME=0 and VMIN=1. For fun, you might want to experiment with different settings of VTIME and VMIN and see how this affects interactive response.