I. Learning Objectives for Student
   A. Regular file reading and writing
   B. Text vs. binary regular files
   C. Printing of hex characters to view bytes values
   D. The errno value and str(errno) function
   E. Stevens-style of wrapping functions

II. Description of Program and Program Requirements
   A. Write a program whose executable name will be `encDec` and which conforms to the following "specification" (which is actually the help message).

      encDec performs an XOR-based encryption/decryption on the specified input file using the specified key file to produce the specified output file.

      encDec -i|--input <input file> -o|--output <output file> -k|--key <key file>
      or
      encDec [-h|--help]

   B. Essentially, the program reads a key and XORs that key with blocks of an plaintext (a.k.a., unencrypted) input file to produce an encrypted output file. If run again with the same key used to encrypt, an encrypted input file will be decrypted to yield the original plaintext input.

   C. Use a makefile to manage the selective compilation of your relocatable and executable objects files. Add `-DDEBUG` to the `gcc` compilation arguments when creating the `encDec_funcs.o` file (or both .o files, as it won't matter). When the `DEBUG` macro is set, the `encryptDecrypt` function will print out the hex characters of the byte being XOR'd, the hex characters of the byte in the key with which it is being XOR'd, and the result of the XOR in hex characters. See the `xorCrypt.c` code in the `InClassPrograms` directory for an example of how the macro is detected and acted upon.

   D. During development, you may find it beneficial to work with a smaller block size. To do so, simply change the "`#define BLOCKSIZE 16`" to a smaller value and recompile. As long as the key is at least as long as the BLOCKSIZE, your code will/must work. If the key is fewer bytes than the BLOCKSIZE, your code must error out.

   E. Exit with the following return code and an appropriate message to stderr under the following conditions. Except for the conditions specified below, the return code is 0. Here are my fprintf statements that you are welcome to use.
      - fprintf(stderr, "command line error (201): incorrect/unexpected argument entered\n");
      - fprintf(stderr, "command line error (202): one or more required arguments not entered\n");
      - fprintf(stderr, "command line error (203): no argument after '-i|--input'\n");
      - fprintf(stderr, "command line error (204): no argument after '-o|--output'\n");
      - fprintf(stderr, "command line error (205): no argument after '-k|--key'\n");
      - fprintf(stderr, "readKey error (206): key is fewer bytes than expected (%zu vs. %zu)\n", items_read, block_size);

- fprintf(stderr, "encrypt/decrypt error (207): unable to write all items read (%zu vs. %zu)\n", items_written, items_read);
- fprintf(stderr, "malloc error (%d): %s\n", errno, strerror(errno));
- fprintf(stderr, "fopen error (%d): %s\n", errno, strerror(errno));
- fprintf(stderr, "fread error (%d): %s\n", errno, strerror(errno));
- fprintf(stderr, "fwrite error (%d): %s\n", errno, strerror(errno));
- fprintf(stderr, "fclose error (%d): %s\n", errno, strerror(errno));

F. You must use Stevens-style wrapping of the following functions: fopen, fread, fwrite, fclose, and malloc.

G. It is recommended you create a directory structure of `cse109/encDec` under your home directory, and do your development work for this assignment in the `encDec` directory. Make sure you change the access privileges so no one else can see the directory's contents.

```
cd ~/cse109                                    ⇐ ~ is shorthand for $HOME
mkdir encDec
cd encDec
```

H. After accepting the assignment in GitHub Classroom, clone your repository using the `git clone` command. You will not need to modify the `encDec.c` or the `encDec_funcs.h` files.

I. You will need to create `encDec_funcs.c` and `makefile` text files (in the same directory) for this assignment.

J. When compiling to prepare for linking, use the following arguments to `gcc` (which should be in your mg alias and put into your makefile).

```
-g -c -Wall -Wextra -Wwrite-strings
```

K. When compiling to link, use the following argument to `gcc`.

```
-g
```

L. Check for memory leaks by running the valgrind tool as follows. You must replace the "<set of valid arguments>" string with a set of valid arguments.

```
valgrind --leak-check=yes encDec <set of valid arguments>
```

M. Work independently. Do not share code or look at each other's code. Discussing concepts/approaches is acceptable.

N. Post any questions/comments on Piazza. Enjoy!

III. For the checkpoint, due before midnight on Thursday, September 30, you need to do the following.

A. Write all the Stevens-style wrapped functions, including the error-checking portion within each. Use the man pages to learn the return value you need to be checking.

B. Write parseArgs and test for error conditions 201 - 205.

C. Write your makefile (or Makefile) to support the following.

1. `make clean` ⇒ removes all .o files and the `encDec` executable, if present
2. `make encDec.o` ⇒ compiles the relocatable object `encDec.o`
3. `make encDec_funcs.o` ⇒ compiles the relocatable object `encDec_funcs.o`
4. `make encDec` (or just `make`) ⇒ links the relocatable objects to create `encDec`