3. Write a program named `fcat` that "concatenates" any number of files by writing them to standard output, one after the other, with no break between files. For example, the following command will display the files `f1.c`, `f2.c`, and `f3.c` on the screen:

   ```
   fcat f1.c f2.c f3.c
   ```

   `fcat` should issue an error message if any file can't be opened. *Hint:* Since it has no more than one file open at a time, `fcat` needs only a single file pointer variable. Once it's finished with a file, `fcat` can use the same variable when it opens the next file.

W 4. (a) Write a program that counts the number of characters in a text file.

   (b) Write a program that counts the number of words in a text file. (A "word" is any sequence of non-white-space characters.)

   (c) Write a program that counts the number of lines in a text file.

   Have each program obtain the file name from the command line.

5. The `xor.c` program of Section 20.1 refuses to encrypt bytes that—in original or encrypted form—are control characters. We can now remove this restriction. Modify the program so that the names of the input and output files are command-line arguments. Open both files in binary mode, and remove the test that checks whether the original and encrypted characters are printing characters.

W 6. Write a program that displays the contents of a file as bytes and as characters. Have the user specify the file name on the command line. Here's what the output will look like when the program is used to display the `pun.c` file of Section 2.1:

```
Offset                  Bytes                    Characters
------  -----------------------------------  ----------
  0   23 69 6E 63 6C 75 64 65 20 3C  #include <
 10   73 74 64 69 6F 2E 68 3E 0D 0A  stdio.h>..
 20   0D 0A 69 6E 74 20 6D 61 69 6E  ..int main
 30   28 76 6F 69 64 29 0D 0A 7B 0D  (void)..{.
 40   0A 20 20 70 72 69 6E 74 66 28  .  printf(
 50   22 54 6F 20 43 2C 20 6F 72 20  "To C, or
 60   6E 6F 74 20 74 6F 20 43 3A 20  not to C:
 70   74 68 61 74 20 69 73 20 74 68  that is th
 80   65 20 71 75 65 73 74 69 6F 6E  e question
 90   2E 5C 6E 22 29 3B 0D 0A 20 20  .\n");..
100   72 65 74 75 72 6E 20 30 3B 0D  return 0;.
110   0A 7D                          .}
```

   Each line shows 10 bytes from the file, as hexadecimal numbers and as characters. The number in the `Offset` column indicates the position within the file of the first byte on the line. Only printing characters (as determined by the `isprint` function) are displayed; other characters are shown as periods. Note that the appearance of a text file may vary, depending on the character set and the operating system. The example above assumes that `pun.c` is a Windows file, so 0D and 0A bytes (the ASCII carriage-return and line-feed characters) appear at the end of each line. *Hint:* Be sure to open the file in `"rb"` mode.

7. Of the many techniques for compressing the contents of a file, one of the simplest and fastest is known as ***run-length encoding***. This technique compresses a file by replacing sequences of identical bytes by a pair of bytes: a repetition count followed by a byte to be repeated. For example, suppose that the file to be compressed begins with the following sequence of bytes (shown in hexadecimal):

   ```
   46 6F 6F 20 62 61 72 21 21 21 20 20 20 20 20
   ```

   The compressed file will contain the following bytes: