CS 1723, Data Structures
Spring Semester, 1998
Programming Assignment 4, Due September 25, 1998
Left Justify Text

The Assignment. For this assignment, you are to write a program that will produce text that is lined up at the left (left justified). Specifically, your program should read an integer (`linelen`) for the number of columns to use, and should read the name of a file for the text. The program should then read the file, and should arrange the words of the file into lines of length no longer than `linelen`. Here is a sample output of this program. Your program should produce the "ruler" across the top as shown and should include two runs, with widths 75 and 20, as shown.

```
runner% lj
75 lewis.text
         1         2         3         4         5         6         7
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345
---------------------------------------------------------------------------
What you have made me see is as plain as the sky, but I never saw it
before. Yet it has happened every day. One goes into the forest to pick
food and already the thought of one fruit rather than another has grown up
in one's mind. Then, it may be, one finds a different fruit and not the
fruit one thought of. One joy was expected and another is given. But this I
had never noticed before--that the very moment of the finding there is in
the mind a kind of thrusting back, or setting aside. The picture of the
fruit you have _not_ found is still, for a moment, before you. And if you
wished--if it were possible to wish--you could keep it there. You could
send your soul after the good you had expected, instead of turning it to
the good you had got. You could refuse the real good; you could make the
real fruit taste insipid by thinking of the other. C.S. Lewis, Perelandra,
1944
runner% lj
20 lewis.text
         1         2
12345678901234567890
--------------------
What you have made
     ... (lines missing)
than another has
grown up in one's
mind. Then, it may
be, one finds a
different fruit and
```

Details.

1. Read the number of columns (`linelen`) and the name of a file (`infilename`) using `scanf` and formats "`%i`" and "`%s`". Reading the file name, opening the file with that name, and reading from that file can look as follows:

```
FILE *infile;
scanf("%i %s", &linelen, infilename);
if ((infile = fopen(infilename, "r")) == NULL) {
   printf("Couldn't open file: %s\n", infilename);
```

```
    exit(1);
}
fscanf(infile, ... );
```

Getting the name of file wrong is common in programs, so an `fopen` should always check that the value returned is not `NULL` (which indicates that the file did not open).

2. This program regards the file of text as a sequence of words, where "word" means any sequence of non-whitespace characters. Thus successive words are separated by whitespace: characters that don't show up when printed, such as a blank (`' '`), a newline (`'\n'`), tab (`'\t'`), form feed (`'\f'`), and vertical tab(`'\v'`). The function `isspace()` returns true for whitespace characters, but you don't necessarily need it, because the function `scanf` ignores whitespace characters on input, so that `scanf` with a "`%s`" format will fetch successive words as strings with a `'\0'` at the end.

3. You can fetch word at a time, and make up a line of output with as many words as possible in it, with just one blank between each word, and not taking up more than `linelen` number of characters. Then the program outputs these lines. The result will be left justified. You should also output the initial scale exactly as shown.

4. There are two strategies for the basic program, that is for making up lines of length less than or equal to `linelen` (where each successive pair of words has exactly one blank between them). One strategy uses C string functions and works with strings as a whole. The other strategy regards the strings as arrays of char, and works with them character at a time. Either strategy, or even a mixed strategy, is valid.

5. Both `fscanf(infile, ...)` and `fgetc(infile)` return `EOF` at end-of-file. If you have trouble with `EOF`, you could use a special word, such as "`#END#`" at the end as sentinel.

What to turn in. Turn in a source listing and the results of at least two runs of the program. One with `linelen` 75 and the other with `linelen` 20. You may use any text that you choose. Try to find something interesting, but please nothing in "poor taste."