ES112.02/04
Fall 2003-2004 Term Project
Project #7: “Paragraph Formatting”

1 The Project

The goal of this project is to write a C program that takes a file containing a single paragraph of (unformatted) text, and outputs the same paragraph to another file after formatting it in a “justified” manner.

2 Justification

This is really a typesetting term, but you may have noticed it in word processing programs. One way or the other, this is best explained by example. Here is a paraghraph which is unformatted:

```
These functions return the number of input items assigned,
which can be fewer than provided for, or even zero, in the event
of a matching failure.  Zero indicates that, while there
was input available, no conversions were assigned; typically
this is due to an invalid input character, such as an alphabetic
character for a ‘%d’ conversion.  The value EOF is returned
if an input failure occurs before any conversion such as an
end-of-file occurs. If an error or end-of-file occurs after
conversion has begun, the number of conversions which were
successfully completed is returned.
```

Here is the same paragraph, fully justified to a 60-column width:

```
These functions  return the number of  input items assigned,
which can be  fewer than provided for, or  even zero, in the
event  of a  matching failure.   Zero indicates  that, while
there  was input  available, no  conversions  were assigned;
typically this is due to an invalid input character, such as
an alphabetic  character for  a ‘%d’ conversion.   The value
EOF  is  returned if  an  input  failure  occurs before  any
conversion  such as an  end-of-file occurs.  If an  error or
end-of-file occurs after conversion has begun, the number of
conversions which were successfully completed is returned.
```

Thus, justification means adjusting each line so that the text in it is exactly the wanted number of characters. To do this, enough words to fill the line are chosen, and additional spaces are added to make the line exactly the wanted length. The last line of the paragraph is single-spaced, and is left short if it happens to be short.

Note that punctuation marks are taken to be part of the words they are adjacent to. The end of a sentence (a period) is the preferred place to add a few extra spaces, since that looks better than large spaces in the middle of sentences.

3 Project Requirements

In this project, you should write a C program that when run, asks for the name of the input file. Once that is entered, it should try to open and read the file. If that fails, the error should be reported, and the program should ask the user for another filename. Once the file is successfully open, the user should be asked for the name of the output file. If that can be opened witout errors, (in case of errors, report them and ask for another filename) the contents of the input file (which is assumed to be a single paragraph of unformatted text) should be printed to the output file justified to 70 columns.

You can assume that at least two words will fit into 70 characters in the input file. Obviously, the first and last characters of any line in the output file must be non-whitespace characters. Anything surrounded by whitespace is considered to be a word (this way you do not have to worry about punctuation). Any whitespace should be ignored and replaced by as many spaces as required for the proper formatting of the line.

Note that you need not read the complete input before writing the output. What you should really do is open both files at once, read from the input file, and write to the output file simultaneously.

Once you have managed this, you can (optionally) ask the user for the number of columns for justification (this should be pretty straightforward.)