Homework V
(100 points)

This problem involves a common text processing computation — fully (left and right) justifying a document. That is, replacing the “whitespace” in a text document so that there are smooth left and right margins. In Haskell, text will simply consist of a String. For this problem, the term “whitespace” will refer to occurrences of spaces (' '), tabs ('\t'), and newlines or carriage returns ('\n'). The term “word” will denote a maximal length substring of non-whitespace characters. The term “line” will denote substrings enclosed by newline characters (or the beginning or end of string). Each string will be decomposed into a sequence of “words” separated by whitespace.

Write a Haskell definition for a function 'justify' that takes two arguments, an integer n>0 and a string 'text'. The resulting string should consist of the words from text with all whitespace replaced so that each line (except the last):

- is exactly n characters long (not counting the newline character),
- includes as many words of the document as possible,
- has a first character that is the first character of some word and last character that is the last character of some word, and
- has each pair of words in a line separated by a number of spaces differing from other pairs in that line by at most one.

If for any reason this reformatting cannot be achieved, an error message should be issued.

So for example

justify 15 "Now is the time for all good men to come to the aid of their country."

should yield

```
"Now is the time
 for  all  good
 men to come to
 the   aid   of
 their country."
```

while

justify 10 "abracadabra"

should yield an error message like "ERROR - width too small".

Note that to insert the special whitespace characters into a string in Haskell, the escape notations '\n' and '\t' are required. Also, to have the escape notations replaced by actual whitespace, it is necessary to use the 'putStr' function. For instance, "abc\ndef" returns "abc\ndef", but putStr "abc\ndef" returns

abc
def

The putStr function can only be used at the top-level since its result type is not String. Hence display of test cases will be shown by expressions of the form putStr (justify ...).

Special instructions

The solutions to this problem must include documentation (in-line comments, and a separate write-up as appropriate) that makes it clear both what general method you used in constructing the program, and how the details of the program accomplish that method. You need to run test cases that exercise **every** component of your code, and include documentation that justifies that your test data meets this condition. It is *not* the grader's responsibility to figure out how you wrote the program and whether it is correct — it is your responsibility to explain your program and convince the grader it is completely tested and correct. Full credit will not be awarded, even for (apparently) correct programs, unless you do so.

In addition to the "paper submission" of your tests and source code, you should submit your Haskell script file electronically using the 'submit' command to course id c111 in directory Hwk5.