CS140 - Assignment 6

Due: Sunday, 10/20 at 11:59pm

For this assignment, you may (and are encouraged to) work with a partner.

This assignment is exclusively coding. However, I strongly encourage you to develop your dynamic programming solution on paper first, like we did for the assignment last week. Once you're comfortable with your DP solution, then work on coding it up.

DP in Code

You decide that you looove latex so much that you're going to try and write your own version of the program. As a first start, you decide to write a program that will take a string of text as input and formats the text so that it is left-justified (all lines begin at the same left column) and the right margin is as even as possible (where "even as possible" is defined below).

For example, given the text:

```
Call me Ishmael.  Some
years ago, never mind how long precisely,
having little
or
```

```
no money in my purse, and nothing
particular to interest me on shore, I thought I
would sail
about a little and see the watery part of the
world.
```

Your system might output something like:

```
Call me Ishmael.  Some years ago, never
mind how long precisely, having little
or no money in my purse, and nothing
particular to interest me on shore, I
thought I would sail about a little
and see the watery part of the world.
```

You formulate this problem as follows. You are given a sequence of words $S = (w_1, \ldots, w_n)$ where w_i consists of $len(w_i)$ characters where each character is of the same width (a so-called "fixed-width" font such as Courier). (Punctuation symbols are considered to be regular characters.)

The words must be placed in the order in which they appear in the sequence, S. The maximum length of a line is L. That is, a line can accommodate up to L symbols, including the space that goes after each word on a line. (Of course, there does not necessarily need to be a space added after the last word placed on a line.)

If words $w_i, \ldots, w_j$ are placed on a line then the total length placed on that line, length(i, j), is:

$$\text{length}(i,j) = \left(\sum_{k=i}^{j-1} (\text{len}(w_k) + 1) \right) + \text{len}(w_j)$$

Notice that this accounts for the length of each of the words and the space immediately after the word. The last word is treated separately since it does not need a space afterwards.

Recall that length(i, j) must be at most L. The *slack* of that line is defined as $(L - \text{length}(i, j))$.

Your goal is to determine how to break the words in S sequentially on lines so as to *minimize the sum of the cubes of the slacks*, known as the *penalty* of the packing.

Write a program that given a string, determines and outputs the best line breaking scheme given the specification above.

Specification:

- Your submissions should be called `format.c`, `format.py`, or `Format.java` if you're using C, Python, or Java, respectively. We will use an autograder that relies on this naming convention.

  Your program should accept L (which is a positive integer) and the name of a file as input on the command line, e.g., `infile.txt` (this could be any filename, but I'm using it for example). We don't want to edit your code, so they need to be command-line parameters. For example:

  - If using C, your code might be compiled and executed as follows:

```
% gcc format.c -o format
% ./format L infile.txt
```

  - If using Python 3[1], your code might be executed as follows:

```
% python3 format.py L infile.txt
```

  - If using Java[2], your code might be compiled and executed as follows:

```
% javac Format.java
% java Format L infile.txt
```

- Your program should **print** out the overall penalty as a single number on a line by itself followed by the actually formatted text on the following lines.

  For example, if you called you program with $L = 15$ and the text "This is a test", you would get:

```
1
This is a test
```

- To determine the "words", just split up the string based on spaces.
- You may assume that no word is longer than L.

Submission

- In addition to your code, submit the output of your approach on the following four test cases below in a separate file called "results.txt". Make sure to include the overall penalty and make sure that your word-processor isn't splitting any lines that you didn't intend to have split.

1. $L = 10$

```
This is a sentence.
```

2. $L = 50$

```
Call me Ishmael.  Some years ago, never mind how long precisely,
having little or no money in my purse, and nothing
particular to interest me on shore, I thought I
would sail about a little and see the watery part of the world.
```

   (Note, your program should treat this as all one string.)

[1] See the `https://docs.python.org/3/howto/argparse.html` module.
[2] See `https://docs.oracle.com/javase/tutorial/essential/environment/cmdLineArgs.html`

3. $L = 20$

   Same text as previous (i.e. the Ishmael text).

4. $L = 80$

```
It was the best of times, it was the worst of times, it was the age of
wisdom, it was the age of foolishness, it was the epoch of belief,
it was the epoch of incredulity, it was the season of Light, it
was the season of Darkness, it was the spring of hope, it was the
winter of despair, we had everything before us, we
had nothing before us, we were all going direct to
heaven, we were all going direct the other way - in short, the
period was so far like the present period, that some of its noisiest
authorities insisted on its being received, for good or for evil,
in the superlative degree of comparison only.
```

I highly recommend writing some additional test cases and checking them by hand!