CS 10A

Word Processing

Emphasizing LISTS, Loops, Logic

Word Processing is the most ubiquitous software application for personal computers. There used to be a vibrant ecosystem with many systems from different vendors such as WordPro, WordStar, WordPerfect, and so called “turnkey systems” from makers such a Wang, which bundled the word processing software together with a custom computer. But it all collapsed into a Microsoft monopoly for Word, although niche programs such as Apple’s Pages do exist.

There are many parts to a word processor and for this assignment we will focus on one toy module called Text Justification. This is the reformatting of text so that the right margin of each line is the same. While modern word processors use graphic letters of different widths and array the characters along a real valued line, in the olden days, each letter was a width of one, and justification was accomplished by inserting extra spaces between words.

Your goal is given a FILE as a global string variable, break it into paragraphs, fill words onto a series of lines, then expand spaces on the line in order to have a justified paragraph, printing out each line along the way.

Your `MAIN` function should take a length (of desired lines), and use several subroutines and functions to accomplish the goal of printing out the whole document with justification.

One simple function, `printline(line,spaces)` is the workhorse. It takes a list of words and a list of integers representing the spaces after each word, and prints a justified line.

```
printline([‘This’,‘is’,‘a’,‘test.’],[3,1,2,0]) prints out:
This...is.a..test
```

First given the whole file as a global string variable called FILE, your main function splits it into paragraphs with the `string.split` method. I’ve included “\n\n” as a separator between paragraphs. Now each paragraph is a long multiline string. To process each paragraph, first split each **by whitespace** (e.g. spaces, tabs, newlines) also using the string split method (look up documentation). Now you have a list of words to process for each paragraph. Main uses a for loop and calls for each paragraph `printpara(words,length)`

`Printpara(words,length)` takes a long list of words representing a paragraph and the length of the justified line. While the len of words is non-zero, use `line=getline(words,length)` to strip out the first n words which don’t overflow the length. Create a numeric list the same length as line of all 1’s followed by a 0 and call it `spaces`. Spaces represents the number of spaces to print after each word. Now, if len(words) != 0 (final line), call `addspace(line,spaces,length)` to pad the spaces between words with spaces so the line is right justified. Finally, call `printline(line,spaces)` to print the justified line.

`Addspace(line,spaces,length)` first calculates the total space used so far by adding the length of each word in line and each number in spaces. Then, while total < length pick a spot using randint and increment spaces[spot] and increment total.

`Printline(line,spaces)` simply does a for loop using `i in range(len(line))` and prints each word `line(i)` followed by the number of spaces indicated by `spaces(i)`

`Getline(words,length)` assembles a line by adding words to a list, keeping track of each word's length + 1, until you either fill a line or run out of words. I'm including the code for this function:

```
def getline(words,length):
        ans=[]
        total=0
        while (length>total) and 0 != len(words):
            word=words.pop(0)
            total += len(word)+1  #add 1 for the space
            ans.append(word)
        #now we are one word too long
        if total > length:
            words.insert(0,ans.pop())
        return ans
```

A sample FILE constant to justify is included below with this assignment. Please show your justification with length 40 and length 60.

```
FILE='''I am typing a bit of nonsense in order to test my text
justification program. The goal is to split a file into paragraphs
using a unique separator,
and then for each paragraph, split it into lines and add spaces so the
each line prints the same length.

We are ignoring punctuation and carriage returns.
The only caveat is that if the last line only has one word in it, it
should be left justified.

The goal of the program is to be a primitive simulation of one of the
functions of a word processor, the most ubiquitous of software
applications which used to have a vibrant ecosystem of variants, but
which has collapsed into a monopoly governed by Microsoft word.
Other parts of word processing require paging files into buffers,
spell checking, display management and keyboard processing.'''
```

Extra Credit: modify main to take a filename and length and read in a file (use included getty.txt) rather than using the FILE constant:

```
main("c:\\temp\\getty.txt",60)
```

The output of main should look like this:

```
I am  typing a bit of nonsense in   order to test my    text
justification  program. The goal   is to split   a file into
paragraphs using a unique separator,  and then  for     each
paragraph, split it into lines  and  add spaces so the  each
line prints the same length.

We are ignoring punctuation   and carriage returns. The only
caveat is that if the last line only  has one word in it, it
should be left justified.

The  goal of  the program is to be a primitive simulation of
one  of the   functions  of   a   word   processor, the most
ubiquitous  of software  applications which  used to  have a
vibrant ecosystem of variants, but which has collapsed  into
a monopoly  governed by Microsoft word.  Other parts of word
processing    require paging   files  into   buffers,  spell
checking, display management and keyboard processing.
```