CS 100 Project Three – Fall 2018

Project Overview: In this project, you will use your knowledge of arrays and functions to write a program to format and print a paragraph, and to report the printing cost. The program first asks for a formatting code. A formatting code is a string with the first character being 'L', 'R' or 'C', and the remaining characters being digits to make up an integer. "L15", "R20" and "C25" are three examples. The integer specifies the output column width. 'L', 'R' and 'C' represent left-justified, right-justified and centered, respectively. The program then asks for a paragraph to be formatted. For simplicity, assume that a paragraph contains no punctuation marks or digits. It is just a sequence of words (consisting of both lower-case and upper-case letters), separated by one or more white spaces. The paragraph ends with <CTRL-D> (i.e. end-of-file).

Assume that the formatted (left-justified, right-justified or centered) paragraph will be printed by a dot-matrix printer. The printing cost will include the number of dots printed, the number of spaces printed, and the number of newlines printed. The following two arrays list the cost (in dots) to print each of 26 upper-case letters and each of 26 lower-case letters. You may copy these two arrays to your program.

```
// the number of dots for 'A' (at 0) through 'Z' (at 25)
int dotsUpper[26]={
16, 19, 13, 18, 18, 14, 16,
17, 11, 12, 14, 11, 21, 19,
16, 15, 18, 18, 15, 11, 15,
13, 20, 11, 11, 15
};
// the number of dots for 'a' (at 0) through 'z' (at 25)
int dotsLower[26]={
12, 14,  9, 14, 14, 12, 18,
14,  9, 10, 12, 10, 16, 12,
12, 14, 14,  8, 13, 10, 12,
 9, 14,  9, 16, 13
};
// based on https://www.urbanfonts.com/fonts/DotMatrix.font
```

In the formatted printout, the original word order shall be maintained. Each output line shall hold as many words as possible, and two consecutive words in the same output line shall be separated by only one space. The additional spaces shall be added to the end to reach the output column width if 'L' (left-justified) is specified. However, the spaces at the end will never be printed because you just need to print a newline at the end to achieve the same effect. The additional spaces shall be added to the beginning if 'R' (right-justified) is specified. If 'C' (centered) is specified, the additional spaces shall be added to the beginning and to the end evenly. However, if the number of spaces to be added is odd, you shall add the extra space to the end. For example, if the output column width is 25 and you have a line of 20 actual characters of text to print, you allocate 2 spaces at the beginning and 3 at the end. Again, the spaces at the end will never be printed.

You can assume the formatting code and the paragraph will be valid as specified. In addition, the specified output column width will always be at least as large as the longest word in the paragraph.

Three sample executions of the program are shown at the end of this document, with the program prompts/output in blue and the user input in red. For ease of output verification by you and the grader, please end each prompt with a newline. Also, please point out the column marks before and after the formatted paragraph as shown in the sample execution. However, the printing cost shall not include the cost of printing these column marks.

Testing: Whenever the input involves multiple lines such as a paragraph, it is a good idea to test the program using input redirection. Otherwise, the input echo and output will be mixed together, and it is very hard to read them. To do this, first use vim to create a data file, say **case1.dat**, and insert a formatting code and a paragraph, as shown below.

```
L15
The quick brown fox
jumps over
the lazy old dog
```

Please make sure every line including the last line ends with a newline. Once you have the test file **case1.dat**, you can test the program using

```
./a.out < case1.dat
```

You need to create more test cases to test your program. To verify whether your program works correctly with a test case, you can post the test case and its result to Piazza to see if others agree with your result. (**However, posting any part of C code from the project on Piazza is prohibited.)** To avoid errors caused by copy and paste, you can use output redirection to save the output into a file. For example, if you run your program using

```
./a.out < case1.dat > case1out.dat
```

All the output including the two input prompts will be saved into the file named **case1out.dat**.

What You Need To Do

- Create a directory **project3** on your machine. In that directory, create a file named **printing.c**
- In **printing.c**, write the code needed to solve the problem stated above. Make sure that your program:
  - Has a header block of comments that includes your name and a brief overview of the program.
  - Has at least three useful functions in addition to the main function. For each additional function, put its signature at the top of your file and give a brief description.
- When you are ready to submit your project, compress your **project3** directory into a single (compressed) zip file, **project3.zip**. See the **Basics** document on Blackboard if you don't remember how to do it.
- Once you have a compressed zip file named **project3.zip**, submit that file to Blackboard.

Project 3 is due at 5:00pm on Friday, October 5. Late projects are not accepted.

A project shall be completed individually, with no sharing of code or solutions.
All submissions will go through MOSS (Measure Of Software Similarity) for similarity check.
The University of Alabama's Code of Academic Conduct will be rigorously enforced.

Three sample executions of the program

```
Enter a formatting code:
L15
Enter a paragraph, ending with <CTRL-D>:
The quick brown fox jumps over the lazy old dog
123456789012345
The quick brown
fox jumps over
the lazy old
dog
123456789012345

Dots printed: 465
Spaces printed: 6
Newlines printed: 4

Enter a formatting code:
C25
Enter a paragraph, ending with <CTRL-D>:
The quick brown fox jumps over the lazy old dog
12345678901234567890123 45
The quick brown fox jumps
  over the lazy old dog
12345678901234567890123 45

Dots printed: 465
Spaces printed: 10
Newlines printed: 2

Enter a formatting code:
R20
Enter a paragraph, ending with <CTRL-D>:
The quick brown fox jumps over the lazy old dog
12345678901234567890
 The quick brown fox
 jumps over the lazy
              old dog
12345678901234567890

Dots printed: 465
Spaces printed: 22
Newlines printed: 3
```