CS 100 Project Three – Spring 2019

Project Overview: In this project, you will write a C program named **`sign.c`** to print out a big sign. The program will read a formatting code followed by a message and then use a dot matrix font and the formatting code to print out a big sign that displays the message.

The dot matrix font to be used for this project is from http://sunge.awardspace.com/glcd-sd/node4.html and you can download the font file named **`font5x7.h`** from Blackboard. The font file contains an array named **`Font5x7`** that shows how to print every printable character from `' '` (the space, ASCII code=32) to `'}'` (the right brace, ASCII code=125). We ignore the last two characters (-> and <-) in the font because they do not exist in the ASCII table. The font does not include `'~'` (the tilde, ASCII code=126), but you are assured that no tilde will appear in the message.

Each character in the font can be printed based on a 5x7 dot matrix. For example, for the capital A, its 5x7 dot matrix is shown below at the left. The value of 1 or 0 at each location determines whether something (usually a dot) should be printed. There are 7 bits in each of the 5 columns. These 7 bits form an integer. Therefore, each dot matrix can be represented by 5 integers. For example, for the capital A, these 5 integers are 1111110 (or 0x1E in hexadecimal), 0010001 (or 0x11), 0010001 (0x11), 0010001 (0x11), and 1111110 (0x1E). In the **`font5x7.h`** file, each printable character has a row of 5 integers in hexadecimal that represents its 5x7 dot matrix. For example, for the capital A, these 5 integers are 0x7E, 0x11, 0x11, 0x11, and 0x7E. In this project, for each printable character, you will print its 5x7 dot matrix on a 8x8 matrix as shown below at the right. It means you will leave 3 blank columns between two characters horizontally and one blank row between two characters vertically. Instead of printing a dot, you will print a specified character based on the formatting code, to be explained later.

0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1
7E	**11**	**11**	**11**	**7E**

0	1	1	1	0	0	0	0
1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	0
1	1	1	1	1	0	0	0
1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0

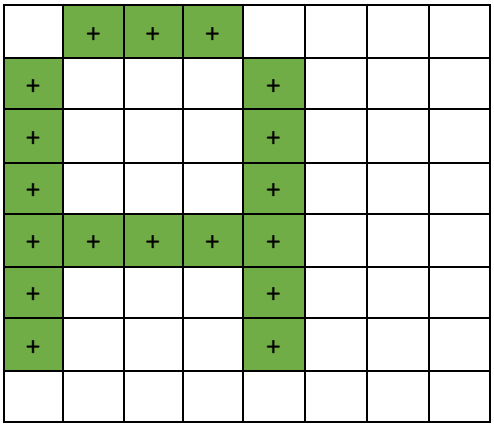
The formatting code consists of two characters in their own line. **The message** consists of one or more lines. Each line contains only printable ASCII characters listed in **`font5x7.h`** (i.e., from the space to the right brace) and it is terminated by a newline. The message is terminated with EOF (end-of-file).

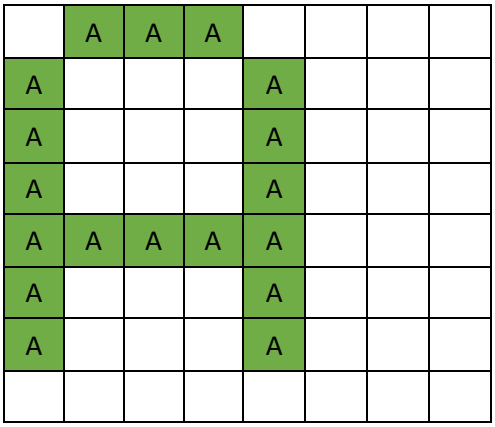
The first character of the formatting code specifies how many characters should be printed in each line of the sign, and it can be `'C'`, `'W'` or `'L'`.

- If the first character of the formatting code is `'C'`, each line of the sign contains only one character. The program shall print one space between two words in the output. (A word here is a sequence of non-whitespace characters that can be read in by `scanf` with the `"%s"` format.) Since you can print only one character in each output line, you will print the space in its own line or simply print out a blank line.
- If the first character of the formatting code is `'W'`, each line of the sign contains only one word.
- If the first character of the formatting code is `'L'`, each line of the message will be printed directly as an output line in the sign. You can use the `fgets` function to read a line, and you can assume that each input line will contain no more than 40 characters.

The smallest thing you can print at this stage of the course is a single character using `printf` with the `"%c"` format. **The second character of the formatting code** specifies what character to print in lieu of the dot in the dot matrix.

- If the second character is a printable character other than `'~'` (the tilde, ASCII code=126), you will print that character when printing a dot in the dot matrix. For example, if the second character is `'+'`, the capital A will be printed as shown below at the left.
- If the second character is `'~'`, you will use the character you are about to print in lieu of the dot. For example, if the second character is `'~'`, the capital A will be printed as shown below at the right. Note that the font defined in **`font5x7.h`** does not specify how to print `'~'` (the tilde), and no `'~'` will appear in the message either.





Please see the sample executions at the end for more details on how the formatting coding works with the message. You can assume the formatting code and the message will always be valid as specified above. For ease of output verification by you and the grader, please terminate each prompt with a newline. Since the message can contains multiple lines, 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 **test2.dat**, and insert a formatting code and a message, as shown below.

```
W@
ROLL     TIDE
     ROLL!
```

Again please make sure every line including the last line ends with a newline. Once you have the test file **test2.dat**, you can test the program using

```
./a.out < test2.dat
```

Three sample executions of the program are shown at the end of this document. Three test files (**test1.dat, test2.dat, test3.dat**) have been created and used in the sample executions.

What You Need To Do

- Create a directory named **`project3`** on your machine. Download **`font5x7.h`** into the **`project3`** directory, and create a file named **`sign.c`** in the **`project3`** directory.
- In **`sign.c`**, write the code needed to solve the problem stated above. Make sure that your program:
  - Has a line of **`#include "font5x7.h"`** in the `include` block.
  - Has a header block of comments that includes your name and a brief overview of the program.

- 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, February 22. 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

test1.dat	test2.dat	test3.dat
C+ ROLL TIDE	W@ ROLL TIDE ROLL!	L~ ROLL TIDE ROLL!

```
./a.out < test1.dat
Enter formatting code:
Enter message:
++++
+   +
+   +
++++
+ +
+  +
+   +

 +++
+   +
+   +
+   +
+   +
+   +
 +++

+
+
+
+
+
+
+++++

+
+
+
+
+
+
+++++
```

```
+++++
  +
  +
  +
  +
  +
  +

 +++
  +
  +
  +
  +
  +
 +++

+++
+  +
+   +
+   +
+   +
+  +
+++

+++++
+
+
++++
+
+
+++++


./a.out < test2.dat
Enter formatting code:
Enter message:
@@@@      @@@     @         @
@    @    @    @    @         @
@    @    @    @    @         @
@@@@     @    @    @         @
@ @      @    @    @         @
@   @    @    @    @         @
@    @     @@@     @@@@@    @@@@@

@@@@@     @@@     @@@      @@@@@
  @         @      @   @     @
  @         @      @    @    @
  @         @      @    @    @@@@
  @         @      @    @    @
  @         @      @   @     @
  @       @@@     @@@      @@@@@
```

```
@@@@      @@@     @        @           @
@    @    @    @    @        @           @
@    @    @    @    @        @           @
@@@@      @    @    @        @           @
@ @       @    @    @        @           @
@  @      @    @    @        @
@   @      @@@     @@@@@    @@@@@       @
```

```
./a.out < test3.dat
Enter formatting code:
Enter message:
RRRR      OOO     L        L                TTTTT     III     DDD      EEEEE
R   R    O   O    L        L                  T        I      D  D     E
R   R    O   O    L        L                  T        I      D   D    E
RRRR     O   O    L        L                  T        I      D   D    EEEE
R R      O   O    L        L                  T        I      D   D    E
R  R     O   O    L        L                  T        I      D  D     E
R   R     OOO     LLLLL    LLLLL              T       III     DDD      EEEEE
```

```
RRRR      OOO     L        L          !
R   R    O   O    L        L          !
R   R    O   O    L        L          !
RRRR     O   O    L        L          !
R R      O   O    L        L          !
R  R     O   O    L        L
R   R     OOO     LLLLL    LLLLL      !
```