

Assignment 3 - Seven Segment Displays

[Start Assignment](#)

Due Friday by 11:59pm **Points** 100 **Submitting** a file upload

Introduction

For this assignment you will develop a C program display an integer number in the style of a seven segment digital display, using only the space, underscore and vertical bar characters. For example, the numbers 10245 and 36978 would be displayed as:

```
  _ _ _ _ _  
  | | | | |  
  | | | | |  
  | | | | |  
  | | | | |  
  | | | | |  
  | | | | |
```

The number to be displayed will be specified on the command line when the program is run, for example:

```
> ./Assg3 10245
```

```
  _ _ _ _ _  
  | | | | |  
  | | | | |  
  | | | | |  
  | | | | |  
  | | | | |  
  | | | | |
```

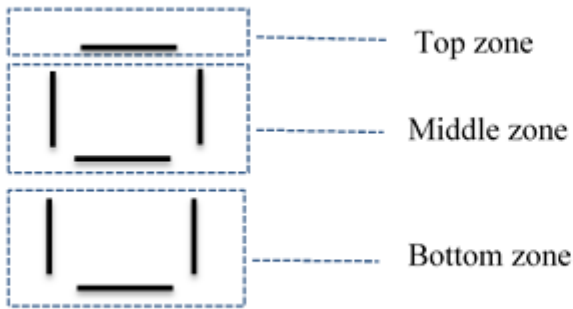
Your program must check that there is a command-line argument and that it is an integer number, terminating with an error message otherwise.

In optional extensions to this exercise, you can extend the program to:

1. Handle negative integers
2. Enable scaling of the display.

Solving the Problem

The first step toward solving this problem is to recognize that the digital display of each digit has seven segments arranged in three horizontal zones, as shown below.



Each zone for each digit can be displayed as three characters, chosen from space, underscore and vertical bar. The digits should be displayed as follows:



Submit your program through the Canvas web site.

Programming Style

In order to encourage good programming practices, you are required to use at least six functions, in addition to the main function:

1. A function to display the top zone characters for all the digits in the number.
2. A function to display the top zone characters for one digit.
3. A function to display the middle zone characters for all the digits in the number.
4. A function to display the middle zone characters for one digit.
5. A function to display the bottom zone characters for all the digits in the number.
6. A function to display the bottom zone characters for one digit.

Bonus Section

A correctly working program, constructed in the way described above, using the coding standards specified below, is sufficient for full points on this assignment.

There are several extra credit extensions to the assignment.

1. (5 bonus points) Enable the display of negative numbers, using a leading minus sign. So, for example, the number -10245 would display as follows, using two underscore characters for the minus sign, aligned with the middle level of the digits.

```
> ./Assg3 -10245
```

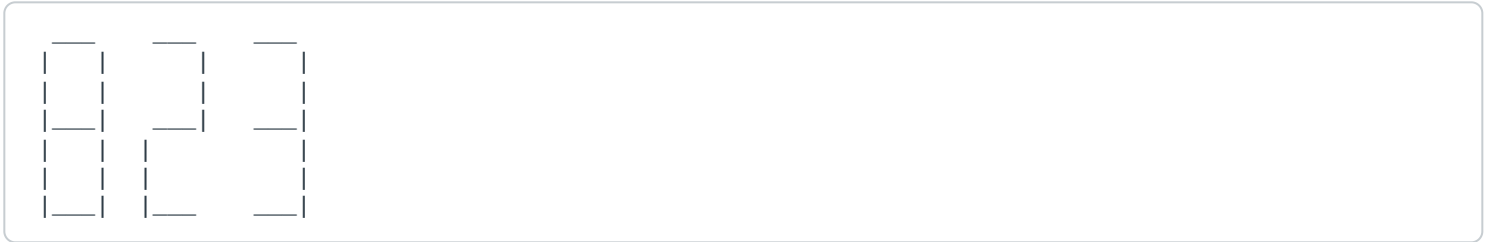
```
-  _ _ _ _ _
|_|_|_|_|_|
|_|_|_|_|_|
```

2. (5 bonus points) Enable the numbers to be displayed in scaled form, with the scale specified by the user in a second command line argument for the program.

The display produced for the program so far is with scale equal to 1. A scale of 2 must double the height and width of the digits, a scale of 3 must triple their size, and so on up to a scale of 5. For example, to display the digit 8, with scale 3, use the command:

```
> ./Assg3 823 3
```

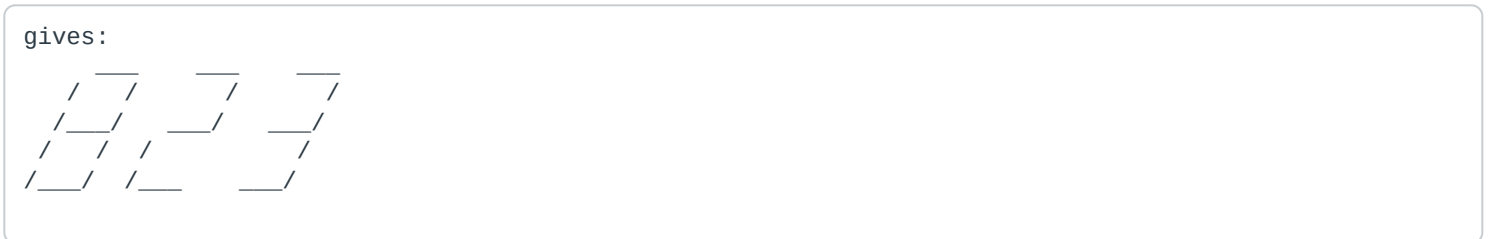
This results in the following output.



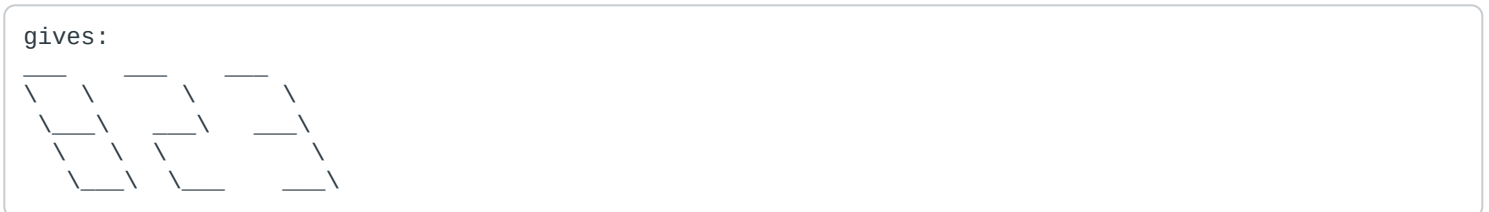
The scaled numbers look very cramped unless we leave some spacing between the digits. The number of spaces between the digits should be one less than the scale factor. For the first part of the assignment, the scale factor is 1.

3. (5 Bonus Points) Print the numbers with a forward or backward slant (as specified by an input parameter). Slanted numbers will default to double height. Note that the numbers end up overlapping.


```
> ./Assg3 823 f
```



```
> ./Assg3 823 b
```



4. (5 bonus points) Instead of a seven segment display present the numbers as roman numbers using the following formats:

 Roman Numbers
The following command:

```
> ./Assg3 823 R
```

Displays:

 823

Note that this could be displayed on a 14-segment display, although some of the letter shapes may vary slightly.

 14- Segment Display

5. (5 bonus points) All command options are fully functional at the same time. For example,

```
> ./Assg3 823 b R 1
```

would specify that 823 be displayed in Roman numbers slanted backwards at standard scale (overrides the default 2 for slanted numbers).

Backslanted roman numbers:  backslant romans

Coding Standards

1. Use meaningful names that give the reader a clue as to the purpose of the thing being named.
2. Avoid the repeated use of numeric constants. For any numeric constants used in your program, assign their value to a variable and then use that variable wherever the value is needed.
3. Use comments at the start of the program to identify the purpose of the program, the author and the date written.
4. Use comments at the start of each function to describe the purpose of the procedure and the purpose of each parameter to the procedure and a description of its return value (if any).
5. Use comments at the start of each section of the program to explain what that part of the program does.
6. Use consistent indentation.