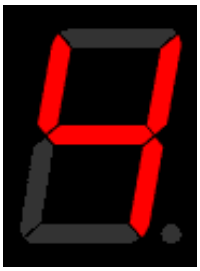
CS 105 (C++)

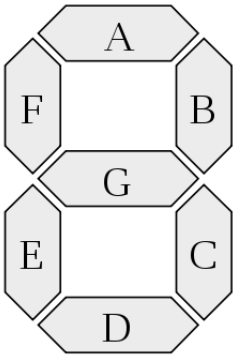
Assignment 2: Seven-Segment Display



I. Overview

In this assignment, you will be working with a simulated seven-segment display.

A seven-segment display is composed of seven elements which can be individually switched on or off to produce representations of numbers or other symbols. According to Wikipedia, these segments are named as shown below.



You will be creating two executables (called "a2a" and "a2b") which will be used in combination with an executable (called "a2c") which you will compile from a source file which I provide, to transform a single input digit character into its representation on a simulated seven-segment display in the following manner.

- The first executable (a2a) takes its single digit character of input, and produces a string of 7 `'0'` or `'1'` characters, indicating which of the segments A through G should be off or on (respectively) in the final display. So, for example, if the input to a2a is `'7'`, the segments A, B, and C should be on, so the output of a2a will be the string of characters `"1110000"`. You can assume that the input to a2a will always be one of the digit characters `'0'` through `'9'`.

- The second executable (a2b) takes the output of a2a and packs it into a single character bit-by-bit. For example, given the input `"0000101"`, a2b will produce a character with the integer value 5. Note that many such characters are non-printing (i.e., they do not produce visual output), but they will still work properly when used as described in this assignment.

  Important Note: Unlike a2a which is only expected to handle one of ten possible inputs, a2b should be written in a general way, so that any input string of seven `'0'` or `'1'` characters will be handled correctly.

- The executable a2c (download source here and compile as described in Section III, below) takes the single packed character generated by a2b and produces a simulated seven-segment display with segments on or off as described by the bits of the character.

Along with the UNIX command `echo` (which simply sends its argument to output), these three executables can be strung together in a pipeline (with the output of one being passed to the input of the next) using pipes ("`|`"). For example, to run the pipeline on the character '2', you could use the command

```
echo 2 | a2a | a2b | a2c
```

to produce the following output.

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

II. Grading

- Minimum Requirements
  - a2a must correctly transform a single digit character passed to its input into a string of seven '0' and '1' characters.
  - a2b must correctly transform *any* string of seven '0' and '1' characters into a single packed character with bit values encoding the values of the input string.
  - Your work must be submitted in files with the following names:

    `a2a.c`

    `a2b.c`
  - These files must compile on a department UNIX machine with the following commands (as described in Section III, below):

    `cc a2a.c -o a2a`

    `cc a2b.c -o a2b`
  - Before evaluation, your code must be submitted via `turnin`, using the following command on a

department UNIX machine:

```
turnin --submit dlessin a2 a2a.c a2b.c
```

- Graded Elements
  Incorporate at least one example of each listed element into your program.
  - `switch` statement.
  - String constants.
  - Bitwise operator `|`, `&`, `^`, or `~`.
  - Shift operator << or >>.

III. The More You Know

The following are some additional items that may be very important for you to know about this assignment.

- So far, we've only had the compiler produce the default executable name (`a.out`), but for this project, you'll need to specify your executable names using the `-o` option to `cc`. For example, if you want to compile `a2a.c` to produce the executable `a2a`, use the following command:

```
cc a2a.c -o a2a
```