# CSC 230 Assignment 2

10/20

Due Sunday, September 29, 2019 at 11:55pm

Late submissions will not be graded.

## Overview

For integers $n$ and $k$, the *binomial coefficient* $\binom{n}{k}$ (read 'n choose k') is the number of ways to choose $k$ objects from a collection of $n$ objects. One way to visualize and compute the binomial coefficients is via Pascal's Triangle[1], which is constructed using the identities below.

$$\binom{n}{0} = 1$$

$$\binom{n}{n} = 1$$

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1} \qquad \text{for } 1 < k < n$$

The rules above can be used to conveniently compute binomial coefficients in a table (starting from $n = 0$), by computing each row's values using the values in the previous row.

The table below shows the first ten rows of Pascal's Triangle ($n = 0, 1, \ldots, 9$) as a 2D array (one of many ways that it could be stored in memory). All of the blank entries can be left undefined (or set to zero) to fit the triangle into a square array.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | | | | | | | | | |
| 1 | 1 | 1 | | | | | | | | |
| 2 | 1 | 2 | 1 | | | | | | | |
| 3 | 1 | 3 | 3 | 1 | | | | | | |
| 4 | 1 | 4 | 6 | 4 | 1 | | | | | |
| 5 | 1 | 5 | 10 | 10 | 5 | 1 | | | | |
| 6 | 1 | 6 | 15 | 20 | 15 | 6 | 1 | | | |
| 7 | 1 | 7 | 21 | 35 | 35 | 21 | 7 | 1 | | |
| 8 | 1 | 8 | 28 | 56 | 70 | 56 | 28 | 8 | 1 | |
| 9 | 1 | 9 | 36 | 84 | 126 | 126 | 84 | 36 | 9 | 1 |

1. https://www.mathsisfun.com/pascals-triangle.html

Your task for this assignment is to write an AVR Assembly program for the ATMega2560 that allows the user to navigate an invisible cursor (or index) through Pascal's Triangle using the black buttons on the LCD Shield (which is mounted on top of our AVR board). At each step, the program will display the value of the triangle at the current location of the cursor/index in binary using the LED lights. The program will implement the following specification.

1. **User input (buttons):**
   - When the program starts (at the RESET signal), the cursor will be positioned at index $[0, 0]$ of the table above and display the value 1 in binary on the LEDs (as described below).
   - When the user presses the UP button, the position $[i, j]$ will change as follows.
     - If $i = j$, nothing will happen.
     - Otherwise, the program will change to position $[i - 1, j]$ in the table.
   - When the user presses the DOWN button, the position $[i, j]$ will change as follows.
     - If $i = 9$, nothing will happen.
     - Otherwise, the program will change to position $[i + 1, j]$ in the table.
   - When the user presses the LEFT button, the position $[i, j]$ will change as follows.
     - If $j = 0$, nothing will happen.
     - Otherwise, the program will change to position $[i, j - 1]$ in the table.
   - When the user presses the RIGHT button, the position $[i, j]$ will change as follows.
     - If $i = j$, nothing will happen.
     - Otherwise, the program will change to position $[i, j + 1]$ in the table.
2. **Program output (LEDs):**
   At all times, the program will track the current location $[i, j]$ (row $i$, column $j$) in the table, and the LED lights will display the corresponding table value in binary format, with the least significant bit on PORTL (pin 42) and the most significant bit on PORTB (pin 52). For example, for $[i, j] = [9, 4]$ the LEDs will display $0b111000$. Note that we have only 6 LEDs and can effectively **display only the lowest 6 bits** of a given binary number, so for this assignment, you are expected to ignore any higher order bits. As mentioned above, when the program starts, the initial position of the index is $[0, 0]$ and the LEDs will display the binary value $0b000001$, which means

only the LED on pin 42 (PORTL) will be on and the rest will be off.

3. **Pascal's Triangle (data/processing):**
   You may compute or store the table values in memory using any format you want. There are several ways to do this including (1) creating a 2D array representation (data structure in memory) as shown above and filling it in with a loop or with pre-computed data and (2) using a recursive function to compute values of the table on-the-fly.

This problem is composed of several sub-problems (hint: functions) that can be completed and tested separately before combining them into the the final solution.

## Submission

Submit your solution via conneX. When doing so, verify that your file is actually uploaded correctly and is not corrupted. You can do so by navigating back to the Assignments section, then downloading your `a2.asm` submission (which you just uploaded), opening it in an editor and visually verifying it's contents.

It must be possible to build and run your program on the equipment provided in the labs (using the same procedure discussed in the lab sessions), otherwise your solution will not be graded.

The solution is worth 6% of your final grade and **must be your individual work**. You may discuss the assignment with your fellow students, but you must write your own code from scratch. Sharing code in any way (or receiving shared code), either electronically or over the shoulder of another student, will be considered plagiarism, even if the code is modified after being shared.

**Late submissions will not be graded.**