Tutorial 3
Bit by Bit

Learning Objectives

After this tutorial, you will be able to:

- Use bitwise operations to get and store information in a single bit position
- Retrieve and manipulate data from multi-dimensional arrays

While it's possible that we may not have worked with multi-dimensional arrays in the lectures at this point of the course, for the purposes of this tutorial they should be quite familiar to how we have worked with them in previous courses. Try it out!

Note that this tutorial will require the use of Bitmasks. You may reference either the lecture slides or the course notes for more information on the bitmasking operations required to complete the tutorial.

Tutorial

In order to receive 1 out of your 2 tutorial marks, you must complete everything in the "Tutorial" section.

1. Download the file `T03.tar` from the tutorial module on Brightspace.
2. Implement the functions `getBit()`, `setBit()`, and `clearBit()` as prototyped.
    a. These will require using the bitmask operations that we discussed in class.
    b. For a challenge and to test your understanding, see if you can remember or work out how to implement the operations before looking them up
3. Using the functions that you just wrote, implement the function `printBits()` as prototyped. The output of `printBits('A')` should be:
    `0100 0001`
4. Write code to accomplish the following, where specified in the provided source code:
    a. Clear the bit position 6 from all values in `arr`
    b. Set the bit position 3 for all values in `arr`
    c. Output all values in `arr`

Exercises

In order to receive full marks for the tutorial, you should at least get a start on one of the following exercises, but you **do not need to complete it** and the instructions are left intentionally vague to allow for experimentation. These are provided to give you some practice with working in C and looking up some documentation [Link]. You may also use the course notes.

You may wish to draw some ideas on paper or a program such as paint for Question 3 to help work out indices.

1. Write a function `printIntBits(int c)` which prints the bits of an integer parameter.
2. Write a function `printIntHex(int c)` which prints the hexadecimal representation of an integer parameter. Don't forget to precede the number with '`0x`' to signify that it is a hex value in the print.
3. Suppose we have an array `arr[m][n]` which we would like to represent as a one-dimensional array `arrFlat[m*n]` (for example, `arr[5][5]` would be represented as `arrFlat[25]`)
    a. Write a function `idx(int j, int k, int n)`, which returns a unique and valid index in the one-dimensional array for each valid pair `(j, k)` of indices in the `m` x `n` 2D array.
    b. Write the inverse functions `idx_m(int i, int n)` and `idx_n(int i, int n)` which give a unique, valid pair of indices in the `m` x `n` 2D array for each valid index in the 1D array.

Note: If implemented correctly, `idx(idx_m(a,n), idx_n(a, n), n) == a`, as long as `a` is a valid index.

c. Write similar functions for a 3D to 1D transformation.

Saving and Submission

In-Person Attendance: Submission is optional, as you can be checked off in class.
Asynchronous: If you are completing the work on your own at home, make sure to follow the submission requirements.

1. Simply submit your `T03.c` file with the questions completed. Make sure to remove any debugging code that you included to simplify the grading process.
    a. If you are more familiar with packaging and submitting tar files, you may do this as well
    b. Make sure not to submit a `T03.c` file that is only the base, provided code.
2. For **full marks** you should have completed all problems in the Tutorial section (and they should be seen upon running the program) and **started** to work on (or completed) at least one of the exercises, beyond just having a function prototype.
3. For **part-marks** you should have completed the “Tutorial” section of the tutorial (no attempt made toward the exercises). The TA retains discretion to provide part-marks for submissions that come *very close* to completion of the tutorial section, but you should plan to complete the whole section as these marks are not guaranteed for incomplete submissions.