In this assignment, you will be dealing with loops and bit-wise operations.

In this assignment, you will program in assembly, a program that will match the following pseudocode (**use the NIOS II simulator, NOT THE DE0-CV board**):

```
int x and y = get_input_from_switches; // see note below

/* write code here that will calculate the hamming distance
Between x and y … see some examples below  */

display the hamming distance on the LEDs

}
```

Input:

When getting X and Y from the switches, you will be getting both X and Y at the same time, in parallel, because X will come from bits SW[9..5] and Y will come from SW[4..0]. So for example, if the user intends X = 0x1B and Y = 0x09, then the switches would be set to:

1 1 0 1 1 0 1 0 0 1 // Just one example of possible X and Y's that could be input

You will have to figure out how to separate this one value into two separate variables X and Y, so that you can save (and use) these values independently in your program. You will probably want to make use of bit-wise operators such as AND/OR and shifts to accomplish the separation of the numbers.

Hamming Distance:

The definition of the hamming distance between X and Y is defined as the number of bits in X and Y that do not match.

For example, if X = 1 1 0 1 1 and Y = 0 1 0 0 1, then the hamming distance between X and Y is 2, because the number of bits that differ between them is 2. Furthermore, it may be convenient to think about taking the XOR between and Y:

```
X = 1 1 0 1 1
Y = 0 1 0 0 1
    1 0 0 1 0  // this is just X XOR Y
```

Note, that the hamming distance between X and Y is simply the number of 1's that is in the result X XOR Y.

This will be somewhat challenging to do, but basically, just XOR X and Y together, and then count up the number of 1's in the result, and that is the answer. You may find that shifting and comparisons may help. Do a few hand examples to see.

Output:

The output on the LEDs should be a number that ranges from 0 to 5, meaning that you need 3 bits, i.e. 3 LEDs. LEDR[2..0].

Submission: (no late work accepted, under any circumstances)

Also, prior to the due date and time (see the date specified on Moodle), **upload the single NIOS II assembly program.** It will be named userid-210-HWA.s, where userid is your userid. **Make sure you check two things afterwards:**

1. That the file was actually uploaded correctly to Moodle.
2. **That when you download your submission from Moodle, that you can save it in a temporary location on your laptop and make sure that it will run in the NIOS II simulator that we're using in class.**