Goal:	A stronger understanding of unsigned and signed (2's complement) binary representation.

PART I: Background

File Decimal2Binary.c contains the skeleton of a C function with two parameters:

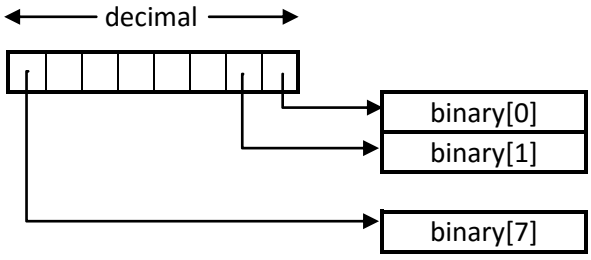
```
void Dec2Bin(int decimal, int binary[8]) ;
```

The function is intended to convert a decimal integer (the first parameter, "decimal") into a corresponding sequence of eight 0's and 1's stored as an array of ints (the second parameter, "binary"). The least-significant bit of "decimal" should be copied into "binary[0]". Your task is to complete the implementation of the function. You may NOT modify the number or type of parameters, or the return type of the function.

Note:

1. The value of the first parameter (decimal) may be any valid 8-bit integer. If the value is signed, the range will be -128 ≤ decimal ≤ +127, or if the value is unsigned, the range will be 0 ≤ decimal ≤ 255.
2. You *could* use the remainder of integer division by two to get the least-significant bit of "decimal" (e.g., decimal % 2), store it in the first element of the array (subscript position 0), and then repeat the process using the quotient (e.g., decimal / 2) until you have all eight bits. This works great for unsigned values, but <u>not</u> for negative signed values because the sign of the remainder will be the same as the sign of the dividend. A better approach is to use C's bit-wise "AND" operator (&) instead of the remainder operator (%), and C's right-shift operator (>>) instead of integer division (/).

Hint: Don't worry about the range. Simply consider the variable "decimal" to be an 8-bit container; your job is to copy each bit held in that variable into a different element of the array called "binary".



PART II: Preparation

1. Download the ZIP file called “Lab Assignments.zip” from the course website on Camino.
2. Unzip the file to your desktop. This should create a folder called “Lab Assignments”. Open the folder.
3. Find and double click on the file called “COEN20.eworkspace”. This will open the EmBitz Integrated Development Environment (IDE) and display the projects for all the lab assignments.
4. If step 3 did not open EmBitz, find the program on the Start Menu and open it. Once EmBitz is open, click on “File > Open” in the upper left-hand corner. In the dialog window that opens, find the pull-down menu in the bottom right and select “EmBitz workspace files”. Then in the middle of the dialog window, navigate to your “Lab Assignments” folder, select the file “COEN20.eworkspace”, and click on “Open”.

PART III: Creating Your Solution

1. Find the project (lab assignment) name in the “Management” panel on the left side of the screen. Make sure that the name is in **boldface**, which indicates that the project is *Active*. If not, right-click on it and select “Activate project”.
2. Expand the project by clicking on the “+” sign immediately to the left of its name. Do the same for any subgroups found within it.
3. Double-click on “Decimal2Binary.c” to open the file. Complete the implementation of the source code.
4. To compile the program, right-click on the project name and select “Build”. If there are any error or warning messages displayed, correct the source code of the function and recompile.

   Hint: Function key F7 is a short-cut for “Build” for the Active project.

5. Connect the STM32F4 Discovery board to a USB port on your computer. This provides both power and a download connection to the device. To download the program to the board, click on “Debug” → “Start/stop Debug Session”.

   Hint: Function key F8 is a short-cut for “Debug” → Start/stop Debug Session”.

6. To run the program, click on “Debug” → “Run”. When the program begins to run it will display the first test case and pause. Press the left button to sequence through all the test cases. Verify that your program behaves as expected.
7. To end the debug session, click on “Debug” → Start/stop Debug Session” again.
8. Demonstrate your working program to the Teaching Assistant.
9. Upload your final version of file Decimal2Binary.c to Camino.