EXERCISE 3
MULTI-PROCESS SEQUENTIAL LOGIC IN A 2-DIGIT HEX COUNTER WITH PWM MODULE AND LED

Dual digit hexadecimal counter

In this exercise, we will extend the previous exercise to design a 2-digit, bi-directional counter that will use hexadecimal representation. Therefore, the counter range goes from `00` to `FF`. The requirements are the same than in the previous exercise with some small modifications:

- ✓ Counting now goes from `00` to `FF` and from `FF` to `00` depending on the down/up input.
- ✓ The overflow is generated when the counter reaches `FF` or `00`.
- ✓ The counter now has two outputs (two different signals, one for each of the output digits), while the data input is a single eight-bit array. (Only include in testbench, no need to implement on the Zedboard.)

In case you are not familiar with the hexadecimal number system, the table below shows the equivalency with the decimal and binary systems.

Hexadecimal	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

You are required to write the module architecture separated in **two processes**: one for the combinatorial logic and another for frequency divider (the clock signal).

Task1:

- Write a test bench which covers all important behaviours of the counter in a sequential manner: reset, enable, load, counting down, counting up.
- Simulate the counter to check they operate correctly.

Task2 (grade 4 or 5):

- Implement to Zedboard with a PWM (**pulse width modulation**) . The output should be mapped to the brightness of LED.
- For the brightness that are represented by RGB values different than 0 or 255, you will need to **use pulse width modulation** to emulate an intermediate value. To do that, create a new top module that maps the three outputs of your driver (which take values from 0 to 255) into a squared signal with the corresponding duty cycle.