Week 4 Error Correcting Code and BitMasking

Question 1

The bitwise operations AND, OR, NOT and XOR are used to do bit-masking, that is, to set **(made 1)** or reset **(made 0)** particular bits on a byte (or word).

Find the appropriate bitmask(s) M and bitwise operator(s) for any byte A for the following cases showing all your working out and intermediate steps:

1. Reset bit 7 leaving the rest untouched.
2. Make sure that bit 0 is set and only this bit is set in the byte.
3. Flip the MSB and bit 3 leaving the other bits untouched.
4. Reset bits 2 and 6, all other bits are set.

Question 2

Calculate the number of parity bits needed to detect and correct a single-bit error in a string of **8, 16, 32** and **64** bits.

Question 3

Determine the single-bit error correction **Hamming** code using **even**-parity for the 7-bit ASCII character "!"

1. How many Hamming parity bits are required to cover all 7 bits?
2. Encode this character into its own 11-bit even Hamming code.
3. Express that result as a single hexadecimal string.

Question 4

Using the **even**-parity **Hamming** code for "!" from Question 3:

1. Flip P4 and show it can be corrected.
2. Flip P4 and D3, show these **cannot be corrected**.

Question 5

Repeat using **even**-parity **SECDED** for ! from Question 3:

1. Flip P4 and show it can be corrected.
2. Flip P4 and D3, show these **can be detected but corrected**.

Question 6

Data has been encoded using an **odd**-parity **SECDED** code. The binary code was then retrieved as 0111 0110.

1. Has an error occurred? Explain your answer (and show your working).
2. If there was an error, either correct it reporting the correct binary string or explain why it could not be corrected.