Blockchain and Cryptocurrency

Assignment # 02

Hamza Shahid

20P-0117

CS-7A

Question 1:

Write a Solidity program to illustrate the functionality of the given Bitwise operators. Additionally, provide a concise explanation, consisting of a maximum of two lines, for each of these operators.

Explanation:

1. Left Shift (<<):
Moves the bits of the left operand to the left by a number of positions specified by the right operand, filling the vacated positions with zeros.

2. Right Shift (>>):
Moves the bits of the left operand to the right by a number of positions specified by the right operand, filling the vacated positions based on the sign bit for signed integers or with zeros for unsigned integers.

3. OR (|):
Performs a bitwise OR operation on each corresponding pair of bits. Resulting bit is 1 if at least one of the corresponding bits of the operands is 1.

4. AND (&):
Performs a bitwise AND operation on each corresponding pair of bits. Resulting bit is 1 only if both corresponding bits of the operands are 1.

5. Exclusive OR (^):
Performs a bitwise XOR (Exclusive OR) operation on each corresponding pair of bits. Resulting bit is 1 if the corresponding bits of the operands are different; otherwise, it's 0.

Certainly! Below is a simple Solidity program illustrating the functionality of the given bitwise operators along with concise explanations:

```solidity
// Solidity program demonstrating Bitwise Operators

pragma solidity ^0.8.0;

contract BitwiseOperatorsExample {

    function leftShift(uint8 a, uint8 b) public pure returns (uint8) {
        return a << b;
    }
    function rightShift(uint8 a, uint8 b) public pure returns (uint8) {
        return a >> b;
    }

    function bitwiseOR(uint8 a, uint8 b) public pure returns (uint8) {
        return a | b;
    }

    function bitwiseAND(uint8 a, uint8 b) public pure returns (uint8) {
        return a & b;
    }

    function bitwiseXOR(uint8 a, uint8 b) public pure returns (uint8) {
        return a ^ b;
    }
}
```

Question 2:

Explain how is it possible for int8 to represent values from -128 to 127.

Answer:

The `int8` data type represents signed 8-bit integers. In a binary system, an 8-bit representation allows for a total of 2^8 possible combinations, which is 256.

For signed integers, one bit is typically used to represent the sign (positive or negative), and the remaining 7 bits are used for the magnitude of the number.

For the range from -128 to 127:

- The leftmost bit (most significant bit) is the sign bit.
  - If it is 0, the number is positive.
  - If it is 1, the number is negative.

- The remaining 7 bits are used for the magnitude.

This binary representation is achieved through a concept called Two's complement, which is a way of representing signed integers in binary. In Two's complement, negative numbers are represented by the positive number's Two's complement.

For example, if you have the binary representation `10000001`, it represents the value -127. The leftmost bit is 1, indicating it's negative, and the remaining bits represent the magnitude using Two's complement.

So, the range of `int8` from -128 to 127 is achieved by allocating one bit for the sign and 7 bits for the magnitude in a binary representation.