###################### Questions ######################

1. Write a function that counts the number of couples of set bits in a byte.
For example, 01110110 has 3 couples of set bits: 01110110, 01110110, 01110110.

2. Write three different implementations for a function that swaps 2 ints.

3. Write a function that counts the number of set bits in a long, using a loop. The loop must be written so that the number of times that it executes is exactly the number of set bits!
For example, if a long is 8 bytes, and the number is 0000000000000000000000000000000000000000000000000000000001100000, then the loop will execute only twice (and not 64 times)!*

4. If you have a signed char c as follows, 11111110 and you perform c >>=1, what would the signed char c look like ?

5. What is the difference between logical shift and arithmetic shift?
Is Cs >> operator logical or arithmetic? Explain.

###################### Answers ######################
1. **Count Couples of Set Bits in a Byte:**

```
int countCouplesOfSetBits(unsigned char byte)
{
    int count = 0;
    for (int i = 0; i < 7; i++)
    {
        if ((byte & (3 << i)) == (3 << i))
    {
            count++;
        }
    }
    return count;
}
```

2. **Swap Two Integers - Three Implementations:**

**Implementation 1: Using a Temporary Variable**
```
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

**Implementation 2: Using Addition and Subtraction**
```
void swap(int *a, int *b)
{
    *a = *a + *b;
    *b = *a - *b;
    *a = *a - *b;
}
```

**Implementation 3: Using XOR**
```
void swap(int *a, int *b)
```

```
{
    *a = *a ^ *b;
    *b = *a ^ *b;
    *a = *a ^ *b;
}
```

3. **Count Set Bits in a Long Using Loop:**

```
int countSetBitsInLong(long num)
{
    int count = 0;
    while (num)
    {
        count++;
        num &= (num - 1); // Clears the least significant set bit
    }
    return count;
}
```

4. **Signed Char Right Shift:**

If you have a signed char `c` with the value `11111110` and you perform `c >>= 1`, the resulting value will be `11111111`. The right shift (`>>`) operation on signed data types usually preserves the sign bit while shifting.

5. **Difference Between Logical Shift and Arithmetic Shift:**

- Logical Shift: In a logical shift, zeros are shifted in from the left, regardless of the sign. Its often used for unsigned types. In C, the logical right shift is performed with `>>`.
- Arithmetic Shift: In an arithmetic shift, the sign bit is shifted in from the left, which preserves the sign of the number. Its often used for signed types. In C, the arithmetic right shift is performed with `>>`.

Cs `>>` operator performs an arithmetic right shift on signed values and a logical right shift on unsigned values. This behavior allows the shift operation to work as expected for both signed and unsigned types.