→ Terminologies in Binary numbers:

7 6 5 4 3 2 1ST 0TH ← Indexing in Binary numbers.

1 0 1 1 1 0 . 1 1

MSB (Most significant Bit)

LSB (Least Significant Bit)

eg: 0TH Bit = 1
4TH Bit = 1
6TH Bit = 0

→ Set => 1
→ Unset => 0

How many sets are there in this Binary no: 6 (Bc, 6 1's are here)

How many unsets are there in this Binary no: 2 (Bc, 2 0's are here).

Q) Suppose we are given this binary no: 101 and we have to check if its 1ST bit is set or unset.

Ans: 1 0 1
↳ 1ST Bit

∴ 1 0 1
& 0 1 0
0 0 0

To check this, we will take (&) of this binary with a binary which has only 1ST bit as set.

→ If the resultant is all 0's then 1st bit was unset else it was set.

e.g: 0 0 0 1 0 1 1
& 0 0 0 1 0 0 0
0 0 0 1 0 0 0

Check for 3rd bit here.

∴ Yes, 3rd bit is set.

NOTE: If we subtract 1 from the binary of power of 2's (2^n) it will give all 1's

e.g: 4 → 100
− 1
011 → 3

e.g: 8 → 1000
− 1
111 → 7

e.g: 16 → 10000
− 1
1111 → 15

→ Function to print decimal no. in Binary:

```
VOID PRINTBINARY(INT NUM)
{
  FOR (INT i=10;  i >= 0;  i--)
  {
```

(2>>10) = 0 (2>>9) = 0 (2>>8) = 0 (2>>7) = 0
(2>>6) = 0 (2>>5) = 0 (2>>4) = 0 (2>>3) = 0
(2>>2) = 0 (2>>1) = 1 (2>>0) = 0

```
    cout << (( num >> i ) & 1);
  }
  cout << "\n";
}
```

FOR num = 2 => 10

2 >> 1 => 1

2 >> 0

```
int main()
{
```

→ Printing Binary of 9 in 11 digits.

```
int a = 9;
printBinary(a);   // 00000001001
```

→ check if 3rd bit is set or unset

```
int i = 3;
if((a & (1 << i)) != 0)
{
    cout << "SET BIT" << "\n";
}
else
{
    cout << "UNSET BIT" << "\n";
}
```

→ Inverting a Binary number eg:
00000001001 → 11111110110

```
printBinary(~a);   // ~ (This symbol is called tilda) It inverts the binary number.
```

→ Setting a bit

```
printBinary(a | (1 << 1));   // setting 1st bit
```

→ 00000001001 | 10 → 000000 01011

→ unsetting a bit

```
PRINTBINARY(a & (~(1<<3))); // unsetting 3RD Bit
```

→ 00000001001 & 0111 →
00000000001

→ Toggling a bit 1 → 0 or 0 → 1

```
PRINTBINARY(a ^ (1<<2)); // toggling 2ND Bit
```

→ 00000001001 ^ 100 →
00000001101

```
PRINTBINARY(a ^ (1<<3)); // toggling 3RD Bit
```

→ 00000001001 ^ 1000 →
00000000001

→ Finding bit count (count of set bits)

```
INT CT = 0;
FOR (INT i=31; i>=0; i--)
{
    IF ((a & (1<<i)) != 0)
    {
        CT++;
    }
}
COUT << CT << "\n"; // 2 (as two 1's in 1001)
```

→ inbuilt function for bit count

```
COUT << __BUILTIN_POPCOUNT(a) << "\n"; // 2
```

→ inbuilt function for bit count of long long integer

```
COUT << __BUILTIN_POPCOUNTLL((1LL<<35)-1) << "\n"; // 35
```

```
    RETURN 0;
}
```