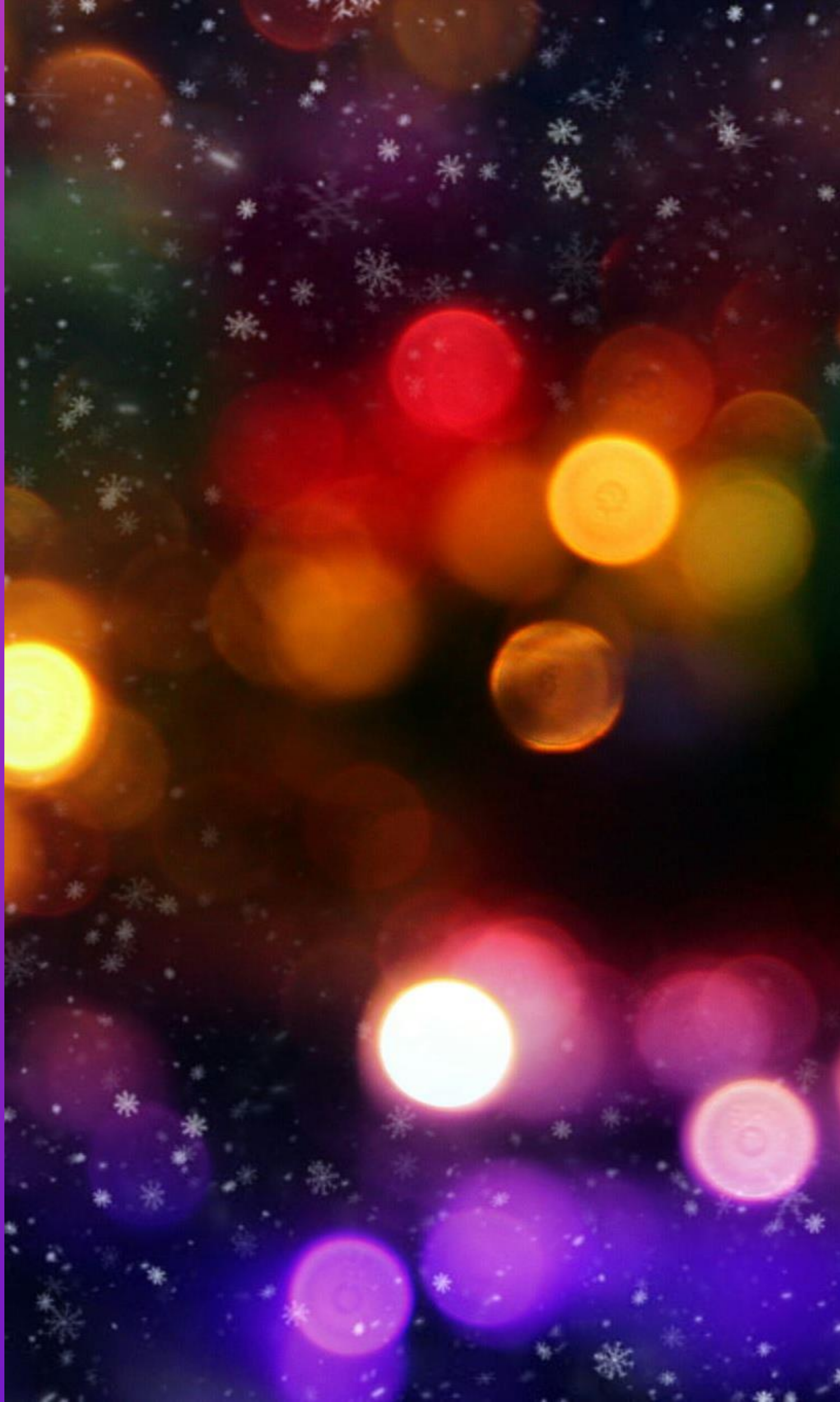
MNNIT COMPUTER CODING CLUB

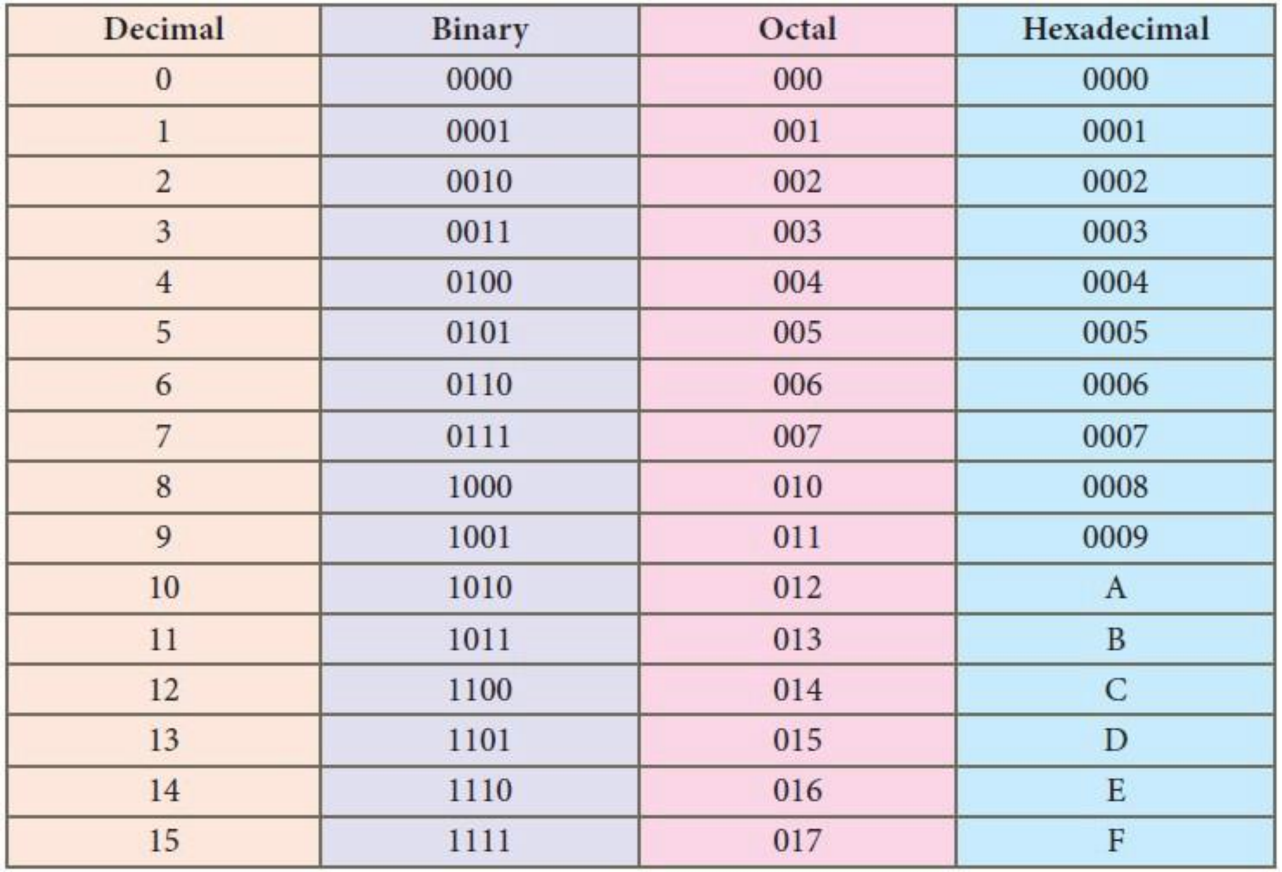
CLASS-8

BASICS OF C



NUMBER SYSTEMS

- Binary
  - Base 2
  - Digits used : 0, 1
- Decimal
  - Base 10
  - Digits used : 0 to 9
- Octal
  - Base 8
  - Digits used : 0 to 7
- Hexadecimal
  - Base 16
  - Digits used: 0 to 9, Letters used : A- F
- Base n



Decimal	Binary	Octal	Hexadecimal
0	0000	000	0000
1	0001	001	0001
2	0010	002	0002
3	0011	003	0003
4	0100	004	0004
5	0101	005	0005
6	0110	006	0006
7	0111	007	0007
8	1000	010	0008
9	1001	011	0009
10	1010	012	A
11	1011	013	B
12	1100	014	C
13	1101	015	D
14	1110	016	E
15	1111	017	F

DECIMAL TO OTHER BASE SYSTEM

- **Step 1** – Divide the decimal number to be converted by the value of the new base.
- **Step 2** – Get the remainder from Step 1 as the rightmost digit (least significant digit) of the new base number.
- **Step 3** – Divide the quotient of the previous divide by the new base.
- **Step 4** – Record the remainder from Step 3 as the next digit (to the left) of the new base number.
- Repeat Steps 3 and 4, getting remainders from right to left, until the quotient becomes zero in Step 3.
- The last remainder thus obtained will be the Most Significant Digit (MSD) of the new base number.

OTHER BASE SYSTEM TO DECIMAL SYSTEM

- **Step 1** – Determine the column (positional) value of each digit (this depends on the position of the digit and the base of the number system).
- **Step 2** – Multiply the obtained column values (in Step 1) by the digits in the corresponding columns.
- **Step 3** – Sum the products calculated in Step 2. The total is the equivalent value in decimal.

BITWISE OPERATORS

- Bitwise AND (&)
- Bitwise OR (|)
- Bitwise XOR (^)
- One's Complement (~)
- Bitwise Left Shift (<<)
- Bitwise Right Shift (>>)

BITWISE OPERATORS

- Bitwise AND (&)

Bit of operand1	Bit of operand2	Resulting Bit
0	0	0
0	1	0
1	0	0
1	1	1

a	0000 1010	= 10 (Decimal)
b	0001 1100	= 28 (Decimal)
a & b	0000 1000	= 8 (Decimal)

BITWISE OPERATORS

- Bitwise OR (|)

Bit of operand1	Bit of operand2	Resulting Bit
0	0	0
0	1	1
1	0	1
1	1	1

<table>
<tr><td>a</td><td>0000 1010</td><td>= 10 (Decimal)</td></tr>
<tr><td>b</td><td>0001 1100</td><td>= 28 (Decimal)</td></tr>
<tr><td>a | b</td><td>0001 1110</td><td>= 30 (Decimal)</td></tr>
</table>

BITWISE OPERATORS

- Bitwise XOR (^)

Bit of operand1	Bit of operand2	Resulting Bit
0	0	0
0	1	1
1	0	1
1	1	0

```
a             0000 1010       = 10 (Decimal)
b             0001 1100       = 28 (Decimal)
______________________________________________
a ^ b         0001 0110       = 22 (Decimal)
```

BITWISE OPERATORS

- One's Complement (~)

Bit of operand1	Resulting Bit
0	1
1	0

a	0000 1010	= 10 (Decimal)
~a	1111 0101	= 245 (Decimal)

*For 8-bit number, remember integer is either 32 bits or 64 bits depending on the OS

BITWISE OPERATORS

- Bitwise Left Shift (<<)

a 0001 1010 1000 0001

a 6,785 (Decimal)

a << 5 00011 0101 0000 001**0 0000**

00000 New bits added

00011 Lost bits

0101 0000 001**0 0000** = 20,512 (Decimal)

BITWISE OPERATORS

- Bitwise Right Shift (>>)

a 0001 1010 1000 0001

a >> 5 **0000 0**000 1101 0100 00001

00000 New bits added
00001 Lost bits
0000 0000 1101 0100 = 212 (Decimal)

Given **N**, if we write all numbers from **1** to **N** (both inclusive) in binary what is the count of 1s I have written.

For example, if N=3,

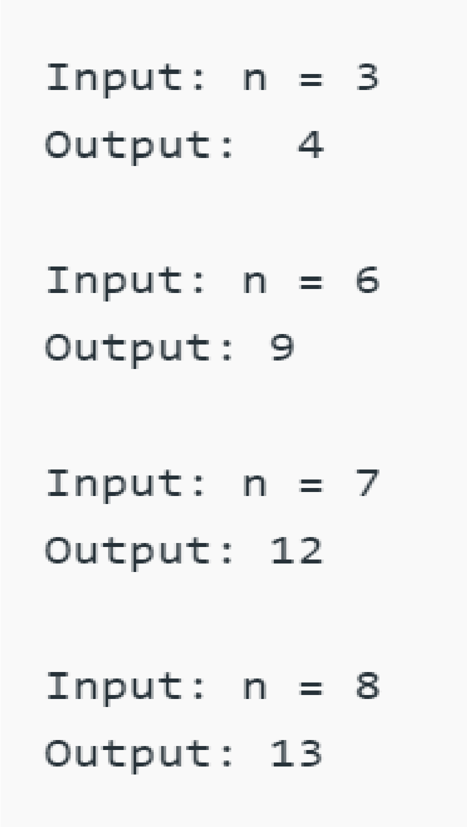
I will write down:

1

10

11

Therefore, a total of 4 ones.



```
Input: n = 3
Output:  4

Input: n = 6
Output: 9

Input: n = 7
Output: 12

Input: n = 8
Output: 13
```

Few problems you can try yourself :

https://www.hackerrank.com/challenges/sum-vs-xor/problem

https://codeforces.com/problemset/problem/1208/A

https://codeforces.com/problemset/problem/1421/A