Number System (Binary & Decimal)

Date: .....
Page: .....

→ Number System - A method to represent numberic values or quantities using different digits.

→ Types of number system
i) Binary number system (Base = 2)
ii) octal " " (" = 8)
iii) Decimal " " (" = 10)
iv) Hexdecimal " " (" = 16)

→ Binary number System
It uses only 2 digits (i.e. 0 & 1) { 0 = off (false) }
It has base 2 { 1 = ON (True) }

→ octal Number System
It uses 8 digits from 0 - 7.
It has base 8

→ Decimal Number System
It uses 10 digits from 0 - 9
It has base 10.

→ Hexodecimal Number System
It uses digits from 0 - 9 & alphabet from A - F
It uses total 16 digits & it has base 16.

→ Base - It is the no. of symbols (digits) that a number system uses.

Date: .....
Page: .....

→ Decimal To Binary conversion

i) Division Method.

a) Divide decimal number by 2.

b) Store remainder. (That will be a bit in binary number)

c) Repeat above steps with the quotient, until quotient is less than 2.

d) Reverse the bits so obtained.

Eg:- $(10)_{10} \rightarrow (?)_2$

		Remainder	
2 | 10			
2 | 5	-	0	↑
2 | 2	-	1	Reverse
2 | 1	-	0	
0	-	1	

$\therefore (10)_{10} \rightarrow (1010)_2$

→ To represent the binary representation of 10 $(1010)_2$ in integer. we multiply each binary bits by 10 to the power of bits place value.

1 0 1 0

- 0 → one place (1) = 10^0
- 1 → Ten place (10) = 10^1
- 0 → Hundred place (100) = 10^2
- 1 → Thousand place (1000) = 10^3

$1010 = 1 \times 10^3 + 0 \times 10^2 + 1 \times 10^1 + 0 \times 10^0$

$= 1000 + 0 + 10 + 0$

$= 1010.$ (Now it is a Integer number, not a bits of binary representation)

Date: .....
Page: .....

→ Code of Division Method To convert Decimal To Binary

```
# include <iostream>
# include <cmath>   (To use power function)
using namespace std;

int decimalToBinary (int num) {
    int binaryNum = 0;  (Store binary numbers (bits))
    int Power = 0;  (Indicate Power of 10, which is
                     initialize with 0)
    while (num > 0) {
        int bit = num % 2;  (Getting last digit or remainder)
        binaryNum += bit * Pow (10, power);
        num = num / 2;  (Remove last digit (quotient))
        Power ++;  (Increase 'power' after each iteration)
    }
    return binaryNum;
}
```

→ DRY RUN

Num = 10, Binary representation : 1010

Convert Binary representation in Integer.

integer = 0; power = 0; (Power of 10)

1st Iteration :-

bit = num % 2 => 10 % 2 = 0

integer = integer + (bit * 10^{power})

= 0 + (0 * 10^0)

= 0 + 0 = (0)

num = num / 2 => 10 / 2 = (5)

Power ++ = 0 + 1

= (1)

Date: .....
Page: .....

2nd iteration :-

bit = num %% 2 => 5 %% 2 = 1 (Remainder)

integer = integer + (bit * 10^{power})

= 0 + (1 * 10^1)

= (10)

num = num / 2 => 5/2 = (2)

power ++ = 1 + 1 = (2)

3rd iteration :-

bit = num %% 2 => 2 %% 2 = 0

integer = 10 + (0 x 10^2)

= 10 + 0 = (10)

num = 2/2 = (1)

power ++ = 2 + 1 = (3)

4th iteration :-

bit = num %% 2 => 1 %% 2 = 1

integer = 10 + (1 x 10^3)

= 10 + 1000

= (1010)

num = 1/2 = 0

power ++ = 3 + 1 = 4.

5th iteration :-

while (num > 0) condition become false, & control exist from loop & return integer (i.e. 1010).

ii) Bitwise Method. (This method is fast)

a) obtain bit with bitwise AND (&) operator (i.e. N & 1)

b) Right shift Number (N) by 1. (N = N >> 1) → To remove last digit of N.

c) Repeat above step till N > 0.

d) Reverse the bits so obtained.

Date: .....
Page: .....

→ Code of Bitwise Method.

```
int decimalToBinary (int num) {
    int binaryNum = 0;
    int power = 0;
    while (num > 0) {
        int bit = (num & 1);    Getting last digit
        binaryNum += bit * pow(10, power);
        num = num >> 1;    Removing last digit
        power++;
    }
    return binaryNum;
}
```

→ Binary To Decimal conversion.

i) Multiply each digits by its place value

i.e. 1 0 1

- one = 1×10^0
- Ten = 0×10^1
- Hundred = 1×10^2

ii) Add up all place values.

iii) Sum is the decimal number

Eg:- $(1010)_2 \rightarrow (?)_{10}$

1 0 1 0

- one place = $2^0 \Rightarrow 0 \times 2^0 = 0$
- Ten " = $2^1 \Rightarrow 1 \times 2^1 = 2$
- Hundred place = $2^2 \Rightarrow 0 \times 2^2 = 0$
- Thousand " = $2^3 \Rightarrow 1 \times 2^3$ (3 → Power = 8; 2 → Base; 1 → digit)

Add up all place values

= 0 + 2 + 0 + 8 = (10)

Date: .....
Page: .....

Eg:- $(10111)_2 = (?)_{10}$

$= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$= 16 + 0 + 4 + 2 + 1$

$= 23$

$\therefore (10111)_2 = (23)_{10}$

→ code of Binary To Decimal conversion.

```
# include <iostream>
# include <cmath>
using namespace std;

int binaryToDecimal (int num) {
    int decimalNum = 0;
    int power = 0;
    while (num != 0) {
        int bit = num % 10;
        decimalNum += bit * pow (2, power);
        num /= 10;
        power++;
    }
    return decimalNum;
}
```

→ DRY RUN.

$(1010)_2 \rightarrow (?)_{10}$

1st iteration :-

num = 1010, power = 0, decimalNum = 0

bit = num % 10 ⇒ 1010 % 10 = 0 (Remainder)

decimalNum += bit x pow (2, power)

$= 0 + 0 \times 2^0 = 0$

Date: .....
Page: .....

num = num / 10 ⇒ 1010 / 10 = 101 (quotient)

Power ++ ⇒ 0 + 1 = 1.

2nd iteration :-

num = 101, decimalNum = 0, Power = 1

bit = 101 % 10 = 1

decimalNum = 0 + 1 × 2^1

= (2)

num = 101 / 10 = 10

Power ++ = 1 + 1 = 2

3rd iteration :-

num = 10, decimalNum = 2, Power = 2

bit = 10 % 10 = 0

decimalNum = 2 + 0 × 2^2

= (2)

num = 10 / 10 = 1

Power ++ = 2 + 1 = 3

4th iteration :-

num = 1, decimalNum = 2, Power = 3

bit = 1 % 10 = 1

decimalNum = 2 + 1 × 2^3

= 2 + 8 = (10)

num = 1 / 10 = 0

Power ++ = 3 + 1 = 4

5th iteration :-

~~(while != 0)~~ while (num != 0) condition become false & control exist from loop & return decimalNum (10).

∴ $(1010)_2 = (10)_{10}$