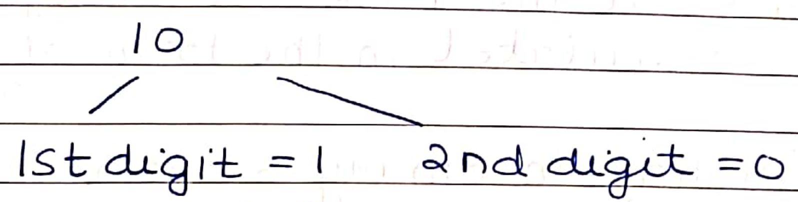
however we can use for loop.

Number system

Method to represent numeric values or quantities using different digits.
There are digits from 0-9.



Digits are arranged & we get a quantity.
Like 1 & 0 digits are used to make 10 which is a number/quantity

Decimal system

This system has base 10. It uses digits from 0 to 9.
Base can be defined as no. of symbols / digits a number system uses.

Binary system

This system has base 2. This uses only 2 symbols namely 0 and 1. We can represent any quantity with these 2 symbols. These symbols are also known as bits.

CPU does calculations, storage in memory all are done in binary system.

0	1	1	0	1	0

This have 6 grids & hence it is known as 6 bit CPU.

0 → nothing. Power is $\overset{\to 0V}{0}$ (No power)
1 → something is there. Voltage is 5V (Power is there)

int a = 5 ;
integer occupy 4 bytes of space or 32 bits. At the backend everything is stored or calculated in the form of bits.

Counting in binary system

0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
⋮	

Any integer can be written in the binary form.

Decimal to binary conversion

(i) This is done by the division method.

1) Divide number by 2.
2) Store remainder.
3) Repeat the above steps until quotient is less than 2.
4) Reverse the bits so obtained.

Ex→ Convert 10 into binary form

Division	Remainder
10/2 → 5	0
5/2 → 2	1
2/2 → 1	0
1/2 → 0	1

Read like this

10 is stored as 1010 in the binary form.

Code

```
int decToBinary (int n) {
        int binaryNumber = 0;
        int i = 0;
        while (n>0) {
            int bit = n % 2;
            // Number from digits question
            binaryNumber = bit * pow(10, i)
                              + binaryNumber;
            n = n/2;
            i++;
        }
return binaryNumber;
}
```

Note → To use pow function, we need to include cmath header file.

(ii) This can also be done via bitwise method

1) Obtain bit with bitwise AND operation i.e n & 1.

2) Right shift n by 1.   n = n >> 1
3) Repeat above steps till n > 0
4) Reverse bits so obtained.

Ex→ Convert 10 in the decimal form.

N = 10 → 1010

* 1010 & 0001 = 0000 → 0

* 1010 >> 1 → 101

Now 101 & 1 →

```
101
001
---
001  → 1
```

* 101 >> 1 → 10

Now 10 & 1 →

```
10
01
--
00  → 0
```

* 10 >> 1 → 1

Now 1 & 1 →

```
1
1
-
1  → 1
```

Now same logic will be applied on making a number from digits or bits.

Code

```
int decToBin (int n) {

        int binaryNumber = 0;
        int i = 0;
        while (n > 0) {
              int bit = n & 1;
        binaryNumber = bit * pow (10, i) +
                                  binaryNumber ;
        n = n >> 1 ;
        i ++ ;
}
return binaryNumber ;
}
```

Note → It is better to use the bitwise method as it is faster operation.

Binary to decimal conversion

1) Multiply each digit with its place value

1 2 3 → 3×10^0

1×10^2 ← 1

2 ↳ 2×10^1

These are the place values in case of decimals.

Place values in case of binary numbers.

------ 8 ←   → 2

1 0 1 0

↳ 4   ↳ 1

2) Add up all the place values.

3) Sum is decimal number.

Place value = Digit * $(\text{Base})^i$

1 0 1 0

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

8 + 0 + 2 + 0 = 10 } Decimal number

Code

```
int bin To Decimal (int bin) {
        int ans = 0;
        int i = 0;
        while (bin > 0 ) {
            int bit = bin % 10 ;
            ans = ans + bit * pow (2, i);
            i++;
            bin = bin /10;
        }
return ans;
}
```

Note- Also we can extract the bit in above question by using bitwise AND.

```
int bit = bin & 1;
```