Bitwise Operators & Loops

1) Bitwise Operators:

Bitwise Operator works on Binary Digits. There are mainly six bitwise operators.

1) AND (&)
2) OR (|)
3) NOT (~)
4) XOR (^)
5) Left Shift (<<)
6) Right Shift (>>)

2) Bitwise AND (&) Operator:

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Ex- a = 5 (101), b = 7 (111)

```
  a
& b
---
```

```
  101
& 111
-----
  101
```

Result = $(101)_2 = 5$

3) Bitwise OR (|) Operator:

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Ex- a = 5 (101), b = 7 (111)

```
  a
| b
---
```

```
  101
| 111
-----
  111
```

Result = $(111)_2 = 7$

4) Bitwise NOT (|) Operator:

X	~X
1	0
0	1

Ex- a = 2 (10)

~a = (01)

1's compliment (10)

2's compliment 10
+1
11

Result = $-(11)_2 = -3$

5) Bitwise XOR Operator:

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Ex- a = 2 (10), b = 4 (100)

a
^ b

010
^ 100

110

Result = $(110)_2 = 6$

6) Bitwise Left Shift (<<) Operator:

Ex- 5 << 1

(0101) << 1 = (1010) Result = $(1010)_2 = 10$

7) Bitwise Right Shift (>>) Operator:

Ex- 5 >> 1

(0101) >> 1 = (0010) Result = $(0010)_2 = 2$

8) Increment & Decrement Operators:

Increment & Decrement operators are used to increase & decrease the value of a variable by 1.

1) Post Increment.

```
int p=5;
cout<< p++<<endl; // 5
cout<< p<<endl; // 6
```

2) Post Decrement.

```
int q=5;
cout<<q--<<endl; //5
cout<<q<<endl; //4
```

3) Pre Increment

```
int r=5;
cout<<++r<<endl; //6
cout<<r<<endl; //6
```

4) Pre Decrement.

```
int s=5;
cout<<--s<<endl; //4
cout<<s<<endl; //4
```

9) For Loop:

C++ for loop is a repetition control structure that allows us to write a loop that is executed a specific number of times. For loop is an entry-controlled loop.

```
for(int i=1; i<=n; i++){
  cout<<i<<endl;
}
```

10) Multiple Conditions in For Loop:

```
for(int i=1,j=10; i<=10 && j>=1; i++, j--){
  cout<<i<<" "<<j<<endl;
}
```

11) Question 1:

Write a program to find the sum of first n natural number using for loop.

Solution:

```
int sum = 0;
for(int i=1; i<=n; i++){
  sum += i;
}
cout<<"Sum is "<<sum<<endl;
```

12) Question 2:

Write a program to print fibonacci series using for loop.

Solution:

```
int a = 0;
int b = 1;
cout<<a<<" "<<b;
for(int i=1; i<=n; i++){
  int c = a + b;
  cout<<" "<<c<<" ";
  a = b;
  b = c;
}
```

13) Question 3:

Write a program to check if number is prime or not.

Solution:

```
bool flag = true;

for(int i=2; i<n; i++){
  if(n%i == 0){
    flag = false;
    break;
  }
}

if(flag){
  cout<<n<<" is Prime Number :)"<<endl;
}
else{
  cout<<n<<" is Not Prime Number :("<<endl;
}
```

14) Break & Continue Statement:

Break Statement leads to an immediate exit of the innermost enclosing loop. Continue Statement skips the current iteration of the loop and continues with next iteration.

15) Scope of Variables:

In general, the scope is defined as the extent up to which something can be worked with. In programming also, the scope of variable is defined as the extent of the program code within which the variable can be accessed or declared or worked with. There are mainly two types of scopes in variables.

1) Local Variables
2) Global Variables

Local Variables: Variables defined within a function or block are said to be local to those functions.

Global Variables: Global Variables can be accessed from any part of the program.

16) Operator Precedence:

Operator precedence determines how operators are parsed concerning each other. Operators with higher precedence become the operands of operators with the lower precedence.

Note: We need not to worry about the operator precedence we need to use Bracket only.

17) Question 4: (Leetcode 1281)

Subtract the product and sum of digits of an integer.

Solution:

```
int Fun(int n){
  int p = 1;
  int s = 0;
  while(n>0){
    int lastDigit = n%10;
    p *= lastDigit;
    s += lastDigit;
    n = n/10;
  }
  return (p-s);
}
```

18) Question 5: (Leetcode 191)

Find the number of 1 Bits.

Solution:

```
int Fun(int n){
  int count = 0;
  while(n>0){
    if(n&1){
      count++;
    }
    n = n>>1;
  }
  return count;
}
```

19) Do While Loop:

Do While loop the condition is tested at the end of the loop body. Do While loop is exit controlled. Do While loop execute at least once.

```
int i=1;
do{
  cout<<i<<endl;
  i++;
}
while(i<=n);
```