ASSIGNMENT:

OPERATORS

Bitwise Operators:

During computation, mathematical operations like: addition, subtraction, multiplication, division, etc

are converted to bit-level which makes processing faster and saves power.Bitwise operators are used in C programming to perform bit-level operations.

Operators	***Meaning of operators***
&	***Bitwise A***
|	***Bitwise OR***
^	***Bitwise exclusive OR***
~	***Bitwise complement***
<<	

Shift left

>>

Shift right

Bitwise AND operator:

Bitwise AND operator is denoted by the single ampersand sign (&). Two integer operands are written on both sides of the (&) operator. If the corresponding bits of both the operands are 1, then the output of the bitwise AND operation is 1; otherwise, the output would be 0.

```
#include <stdio.h>
int main()
{
  int a=6, b=14;  // variable declarations
  printf("The output of the Bitwise AND operator a&b is %d",a&b);
  return 0;
}
```

Output of Bitwise AND operator and b is 6

Bitwise OR operator:

The bitwise OR operator is represented by a single vertical sign (|). Two integer operands are written on both sides of the (|) symbol. If the bit value of any of the operand is 1, then the output would be 1, otherwise 0.

```
#include <stdio.h>
int main()
{
  int a=23,b=10;  // variable declarations
  printf("The output of the Bitwise OR operator a|b is %d",a|b);
  return 0;
}
```

Output of Bitwise OR operator a|b is 31

Bitwise exclusive OR operator:

Bitwise exclusive OR operator is denoted by (^) symbol. Two operands are written on both sides of the exclusive OR operator. If the corresponding bit of any of the operand is 1 then the output would be 1, otherwise 0.

```
#include <stdio.h>
int main()
{
  int a=12,b=10;  // variable declarations
  printf("The output of the Bitwise exclusive OR operator a^b is %d",a^b);
  return 0;
}
```

Output of Bitwise exclusive OR operator a^b is 6

Ternary Operator:

The ternary operator for decision making in place of longer if and else conditional statements.

1. *The ternary operator take three arguments*
2. *The first is a comparison argument*
3. *The second is the result upon a true comparison*
4. *The third is the result upon a false comparison*

Syntax:

```
condition ? value_if_true : value_if_false
```

Example:

```
int a = 10, b = 20, c;
```

Output :10

```
c = (a < b) ? a : b;

printf("%d", c);
```