TASK-OPERATORS

BITWISE OPERATOR:

1.) & (bitwise AND) in C takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.
2.) | (bitwise OR) in C takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 if any of the two bits is 1.
3.) ^ (bitwise XOR) in C takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.
4.) << (left shift) in C takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.
5.) >> (right shift) in C takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.
6.) ~ (bitwise NOT) in C takes one number and inverts all bits of it.

Example:

```
#include<stdio.h>
main()
{    int a=100,b=30;
     printf( "%d&%d:",a&b);
     printf( "%d|%d:",a|b);
     printf( "%d^%d:",a^b);
     printf( "%d<<%d:",a<<b);
     printf( "%d>>%d:",a>>b);
     printf( "%d~%d:",a~b);
     return 0;

}
```

TERNERY OPERATOR:

Syntax: variable= expression 1? expression 2: expression 3

If expression 1 is true then expression 2 gets executed else expression 3 is executed.

Example:

```
#include<stdio.h>
main()
{ int x,y;
int max=x>y?x:y;
printf("Maximum of %d and %d is %d",x,y,max);
return 0;
}
```