GO - BITWISE OPERATORS

The Bitwise operators supported by Go language are listed in the following table. Assume variable A holds 60 and variable B holds 13, then:

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	A & B will give 12 which is 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	$A\|B$ will give 61 which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	A^B will give 49 which is 0011 0001
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 0000 1111

Example

Try the following example to understand all the bitwise operators available in Go programming language:

```go
package main

import "fmt"

func main() {

   var a uint = 60 /* 60 = 0011 1100 */
   var b uint = 13 /* 13 = 0000 1101 */
   var c uint = 0

   c = a & b       /* 12 = 0000 1100 */
   fmt.Printf("Line 1 - Value of c is %d\n", c )

   c = a | b       /* 61 = 0011 1101 */
   fmt.Printf("Line 2 - Value of c is %d\n", c )

   c = a ^ b       /* 49 = 0011 0001 */
   fmt.Printf("Line 3 - Value of c is %d\n", c )

   c = a << 2      /* 240 = 1111 0000 */
   fmt.Printf("Line 4 - Value of c is %d\n", c )

   c = a >> 2      /* 15 = 0000 1111 */
   fmt.Printf("Line 5 - Value of c is %d\n", c )
}
```

When you compile and execute the above program it produces the following result:

```
Line 1 - Value of c is 12
Line 2 - Value of c is 61
Line 3 - Value of c is 49
```

```
Line 4 - Value of c is 240
Line 5 - Value of c is 15
```