LISP - OPERATORS

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. LISP allows numerous operations on data, supported by various functions, macros and other constructs.

The operations allowed on data could be categorized as:

- Arithmetic Operations
- Comparison Operations
- Logical Operations
- Bitwise Operations

Arithmetic Operations

The following table shows all the arithmetic operators supported by LISP. Assume variable **A** holds 10 and variable **B** holds 20 then:

Show Examples

Operator	Description	Example
+	Adds two operands	$+AB$ will give 30
-	Subtracts second operand from the first	$-AB$ will give -10
*	Multiplies both operands	$*AB$ will give 200
/	Divides numerator by de-numerator	$/BA$ will give 2
mod,rem	Modulus Operator and remainder of after an integer division	$modBA$will give 0
incf	Increments operator increases integer value by the second argument specified	$incfA3$ will give 13
decf	Decrements operator decreases integer value by the second argument specified	$decfA4$ will give 9

Comparison Operations

Following table shows all the relational operators supported by LISP that compares between numbers. However unlike relational operators in other languages, LISP comparison operators may take more than two operands and they work on numbers only.

Assume variable **A** holds 10 and variable **B** holds 20, then:

Show Examples

Operator	Description	Example
=	Checks if the values of the operands are all equal or not, if yes then condition becomes true.	$=AB$ is not true.
/=	Checks if the values of the operands are all different or not, if values are not equal then condition becomes true.	$/=AB$ is true.

>	Checks if the values of the operands are monotonically decreasing.	$> AB$ is not true.
<	Checks if the values of the operands are monotonically increasing.	$< AB$ is true.
>=	Checks if the value of any left operand is greater than or equal to the value of next right operand, if yes then condition becomes true.	$>= AB$ is not true.
<=	Checks if the value of any left operand is less than or equal to the value of its right operand, if yes then condition becomes true.	$<= AB$ is true.
max	It compares two or more arguments and returns the maximum value.	$maxAB$ returns 20
min	It compares two or more arguments and returns the minimum value.	$minAB$ returns 20

Logical Operations on Boolean Values

Common LISP provides three logical operators: **and, or,** and **not** that operates on Boolean values. Assume **A** has value nil and **B** has value 5, then:

Show Examples

Operator	Description	Example
and	It takes any number of arguments. The arguments are evaluated left to right. If all arguments evaluate to non-nil, then the value of the last argument is returned. Otherwise nil is returned.	$andAB$ will return NIL.
or	It takes any number of arguments. The arguments are evaluated left to right until one evaluates to non-nil, in such case the argument value is returned, otherwise it returns **nil**.	$orAB$ will return 5.
not	It takes one argument and returns **t** if the argument evaluates to **nil.**	$notA$ will return T.

Bitwise Operations on Numbers

Bitwise operators work on bits and perform bit-by-bit operation. The truth tables for bitwise and, or, and xor operations are as follows:

Show Examples

p	q	p and q	p or q	p xor q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

```
Assume if A = 60; and B = 13; now in binary format they will be as follows:
```

```
A = 0011 1100
B = 0000 1101
-----------------
A and B = 0000 1100
A or B = 0011 1101
A xor B = 0011 0001
not A  = 1100 0011
```

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

Operator	Description	Example
logand	This returns the bit-wise logical AND of its arguments. If no argument is given, then the result is -1, which is an identity for this operation.	*logandab*) will give 12
logior	This returns the bit-wise logical INCLUSIVE OR of its arguments. If no argument is given, then the result is zero, which is an identity for this operation.	*logiorab* will give 61
logxor	This returns the bit-wise logical EXCLUSIVE OR of its arguments. If no argument is given, then the result is zero, which is an identity for this operation.	*logxorab* will give 49
lognor	This returns the bit-wise NOT of its arguments. If no argument is given, then the result is -1, which is an identity for this operation.	*lognorab* will give -62,
logeqv	This returns the bit-wise logical EQUIVALENCE *alsoknownasexclusivenor* of its arguments. If no argument is given, then the result is -1, which is an identity for this operation.	*logeqvab* will give -50