LISP - PREDICATES

Predicates are functions that test their arguments for some specific conditions and returns nil if the condition is false, or some non-nil value is the condition is true.

The following table shows some of the most commonly used predicates:

Predicate	Description
atom	It takes one argument and returns t if the argument is an atom or nil if otherwise.
equal	It takes two arguments and returns **t** if they are structurally equal or **nil** otherwise
eq	It takes two arguments and returns **t** if they are same identical objects, sharing the same memory location or **nil** otherwise
eql	It takes two arguments and returns **t** if the arguments are **eq**, or if they are numbers of the same type with the same value, or if they are character objects that represent the same character, or **nil** otherwise
evenp	It takes one numeric argument and returns **t** if the argument is even number or **nil** if otherwise.
oddp	It takes one numeric argument and returns **t** if the argument is odd number or **nil** if otherwise.
zerop	It takes one numeric argument and returns **t** if the argument is zero or **nil** if otherwise.
null	It takes one argument and returns **t** if the argument evaluates to nil, otherwise it returns **nil.**
listp	It takes one argument and returns **t** if the argument evaluates to a list otherwise it returns **nil.**
greaterp	It takes one or more argument and returns **t** if either there is a single argument or the arguments are successively larger from left to right, or **nil** if otherwise.
lessp	It takes one or more argument and returns **t** if either there is a single argument or the arguments are successively smaller from left to right, or **nil** if otherwise..
numberp	It takes one argument and returns **t** if the argument is a number or **nil** if otherwise.
symbolp	It takes one argument and returns **t** if the argument is a symbol otherwise it returns **nil**.
integerp	It takes one argument and returns **t** if the argument is an integer otherwise it returns **nil**.
rationalp	It takes one argument and returns **t** if the argument is rational number, either a ratio or a number, otherwise it returns **nil**.
floatp	It takes one argument and returns **t** if the argument is a floating point number otherwise it returns **nil.**
realp	It takes one argument and returns **t** if the argument is a real number otherwise it returns **nil**.
complexp	It takes one argument and returns **t** if the argument is a complex number otherwise it returns **nil.**
characterp	It takes one argument and returns **t** if the argument is a character otherwise it returns **nil**.

stringp	It takes one argument and returns **t** if the argument is a string object otherwise it returns **nil**.
arrayp	It takes one argument and returns **t** if the argument is an array object otherwise it returns **nil**.
packagep	It takes one argument and returns **t** if the argument is a package otherwise it returns **nil.**

Example 1

Create a new source code file named main.lisp and type the following code in it.

```
(write (atom 'abcd))
(terpri)
(write (equal 'a 'b))
(terpri)
(write (evenp 10))
(terpri)
(write (evenp 7 ))
(terpri)
(write (oddp 7 ))
(terpri)
(write (zerop 0.0000000001))
(terpri)
(write (eq 3 3.0 ))
(terpri)
(write (equal 3 3.0 ))
(terpri)
(write (null nil ))
```

When you execute the code, it returns the following result:

```
T
NIL
T
NIL
T
NIL
NIL
NIL
T
```

Example 2

Create a new source code file named main.lisp and type the following code in it.

```
(defun factorial (num)
   (cond ((zerop num) 1)
      (t ( * num (factorial (- num 1))))
   )
)
(setq n 6)
(format t "~% Factorial ~d is: ~d" n (factorial n))
```

When you execute the code, it returns the following result:

```
Factorial 6 is: 720
```