Lisp: An Introduction

- LISP stands for LISt Programming
- Developed by John McCarthy, MIT, in 1958
    - It was first implemented by Steve Russell on an IBM 704 computer
    - It is particularly suitable for Artificial Intelligence programs, as it processes symbolic information efficiently.
- Common LISP originated during the decade of 1980 to 1990
    - It is machine-independent.
    - It uses iterative design methodology .
    - It has easy extensibility.
    - It allows to update the programs dynamically .
    - It provides high level debugging.
    - It provides advanced object-oriented programming.

ENVIRONMENT SETUP

- CLISP is the GNU Common LISP multi-architectural compiler used for setting up LISP in Windows.
- To run a .lisp or .lsp file,

  simply use: clisp hello.lisp

PROGRAM STRUCTURE

- LISP expressions are called symbolic expressions or S-expressions. The S-expressions are composed of three valid objects :
  - Atoms
  - Lists
  - Strings
- Any S-expression is a valid program.
- LISP programs run either on an interpreter or as compiled code.
- The interpreter checks the source code in a repeated loop, which is also called the Read- Evaluate-Print Loop (REPL).
- It reads the program code, evaluates it, and prints the values returned by the program.

Basic building blocks of LISP

The basic building blocks of LISP are the atom, list and the string. These three are the only valid objects in LISP. They are called symbolic-expressions or s-expressions.

•*Atom:* It is a number or string of contiguous characters, including numbers and special characters, e.g. bill, 1200, a100, rock* etc.

•*List:* It is a sequence of zero or more symbolic-expressions (atoms and/or other lists) enclosed within parentheses,e.g. (a b c), (a, (j,i), (e,g,h)), (), (sat,sun) etc.

•*String:* It is a group of characters enclosed in double quotation marks,e.g. "hello LISP", "^&^%%fbdfdbfds" etc.

Which elements always return their own values in LISP

There are three types of special elements are that they are constant and always evaluate to themselves, returning their own value. These are numbers , the letter t (for logical true) and nil (for logical false). So, if we evaluate any one of these, the answer is the element itself:

```
->60
60
->t
T
->nil
 NIL
```

BASIC SYNTAX

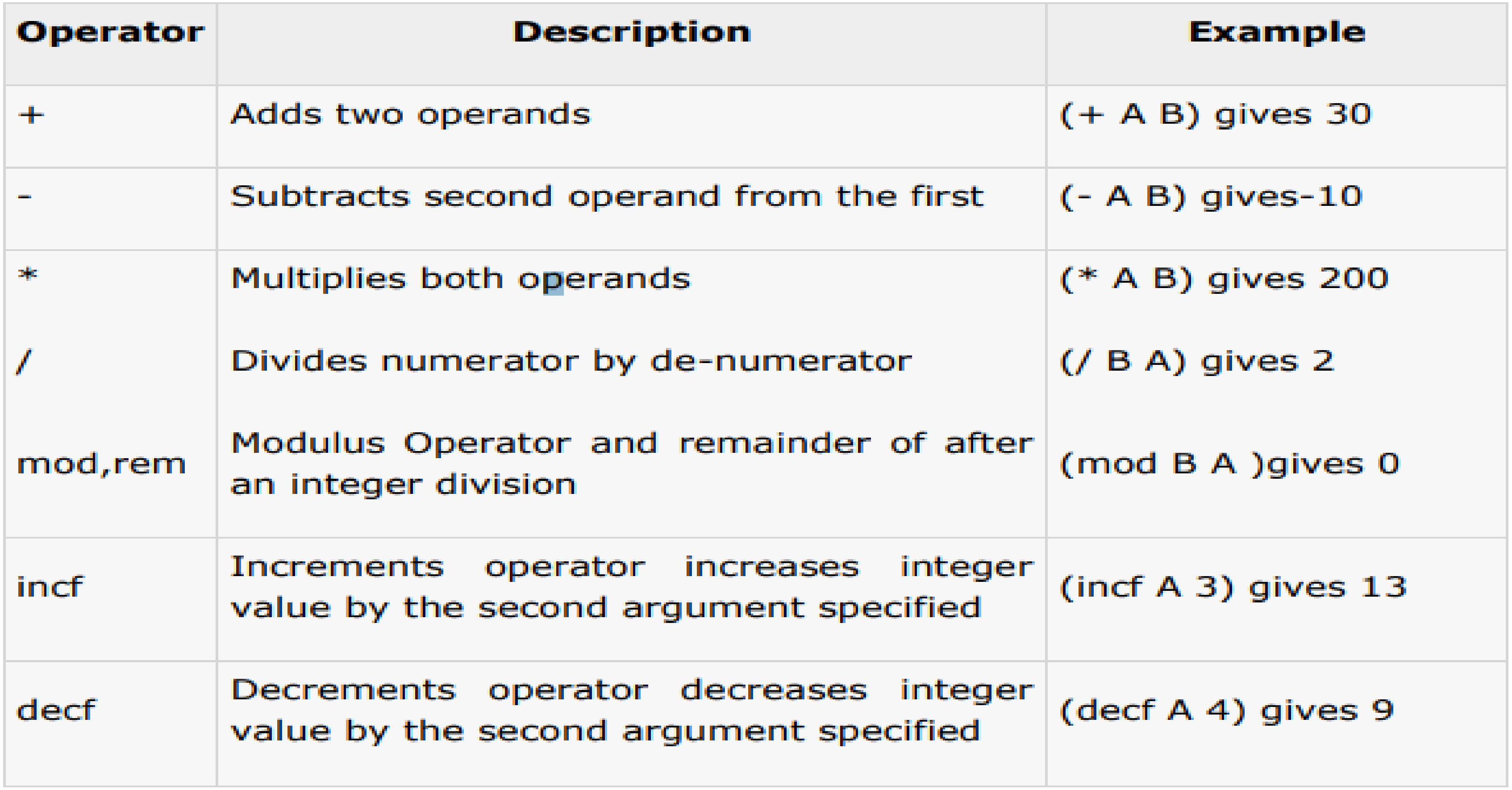
- LISP programs are made up of three basic elements:
  - atom
  - list
  - string
- An atom is a number or string of contiguous characters. It includes numbers and special characters.
- The following examples show some valid atoms:
  - hello-from-thapar-university
  - name
  - 123008907
  - *hello*
  - Block#221
  - abc123

BASIC SYNTAX

- A list is a sequence of atoms and/or other lists enclosed in parentheses.
- The following examples show some valid lists:
    - (i am a list)
    - (a (a b c) d e f g h)
    - (father tom (susan bill joe))
    - (sun mon tue wed thur fri sat)
    - ( )
- A string is a group of characters enclosed in double quotation marks.
- The following examples show some valid strings:
    - " I am a string“
    - "a ba c d efg #$%^&!"
    - "Please enter the following details:“
    - "Hello from ‘Thapar University'! "

Arithmetic Operations

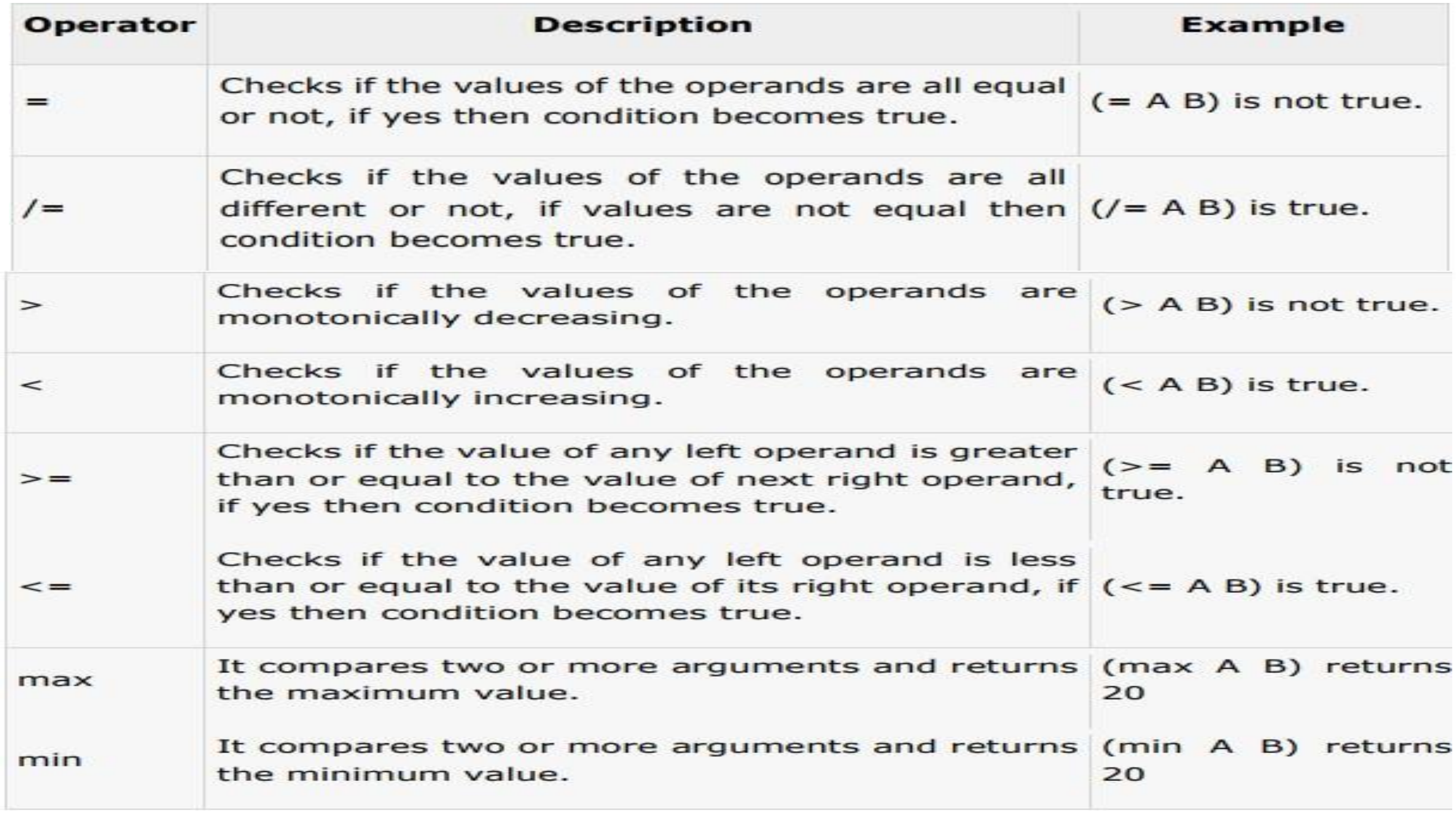
The following table shows all the arithmetic operators supported by LISP. Assume variable A = 10 and variable B = 20 then:



Operator	Description	Example
+	Adds two operands	(+ A B) gives 30
-	Subtracts second operand from the first	(- A B) gives-10
*	Multiplies both operands	(* A B) gives 200
/	Divides numerator by de-numerator	(/ B A) gives 2
mod,rem	Modulus Operator and remainder of after an integer division	(mod B A)gives 0
incf	Increments operator increases integer value by the second argument specified	(incf A 3) gives 13
decf	Decrements operator decreases integer value by the second argument specified	(decf A 4) gives 9

Comparison Operations

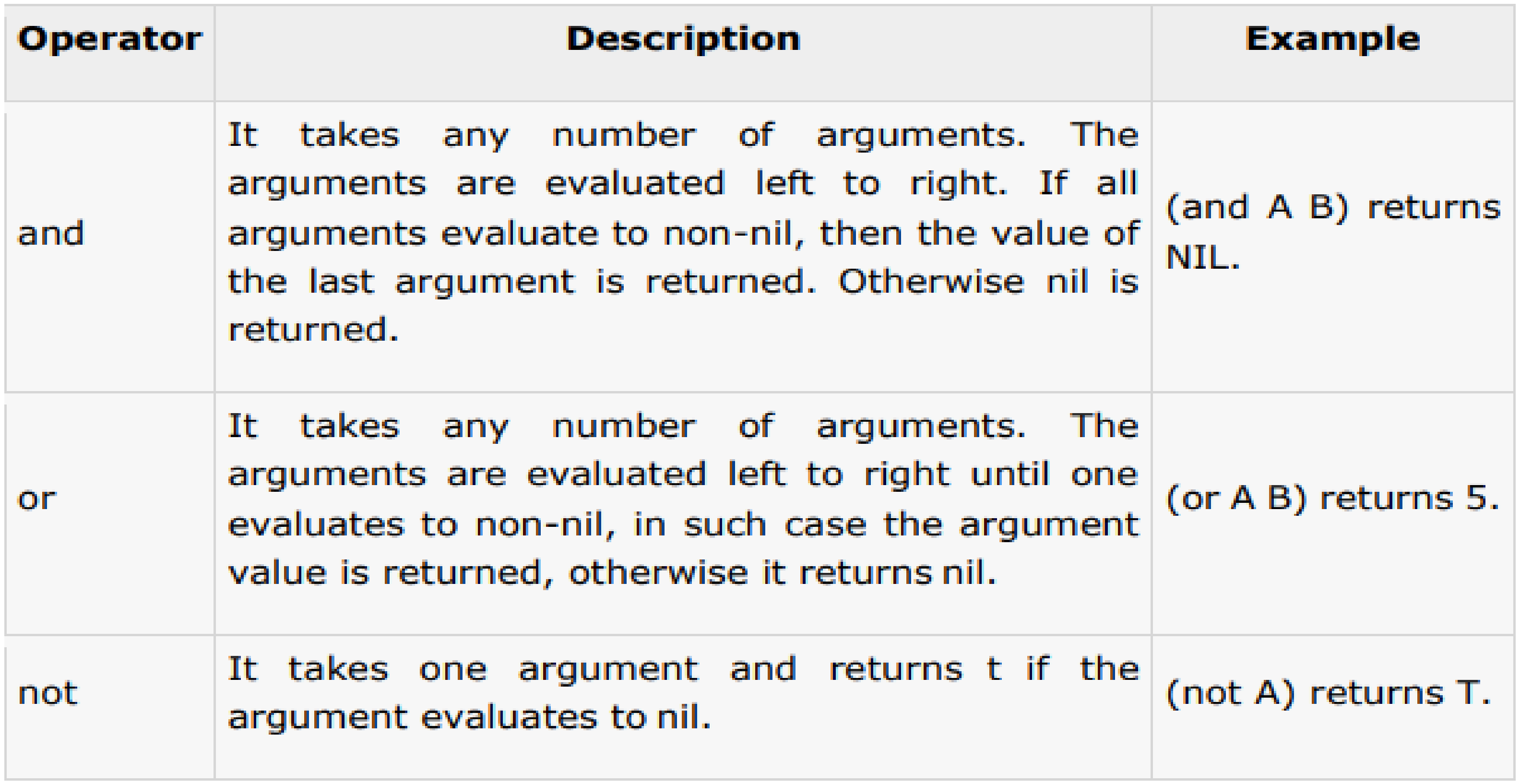
Following table shows all the relational operators supported by LISP that compares between numbers. Assume variable A = 10 and variable B = 20, then:



Operator	Description	Example
=	Checks if the values of the operands are all equal or not, if yes then condition becomes true.	(= A B) is not true.
/=	Checks if the values of the operands are all different or not, if values are not equal then condition becomes true.	(/= A B) is true.
>	Checks if the values of the operands are monotonically decreasing.	(> A B) is not true.
<	Checks if the values of the operands are monotonically increasing.	(< A B) 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.	(>= A B) 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.	(<= A B) is true.
max	It compares two or more arguments and returns the maximum value.	(max A B) returns 20
min	It compares two or more arguments and returns the minimum value.	(min A B) returns 20

Logical Operations on Boolean Values

Common LISP provides three logical operators: and, or, and not that operate on boolean values. Assume A = nil and B = 5, then:



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.	(and A B) returns 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.	(or A B) returns 5.
not	It takes one argument and returns t if the argument evaluates to nil.	(not A) returns T.

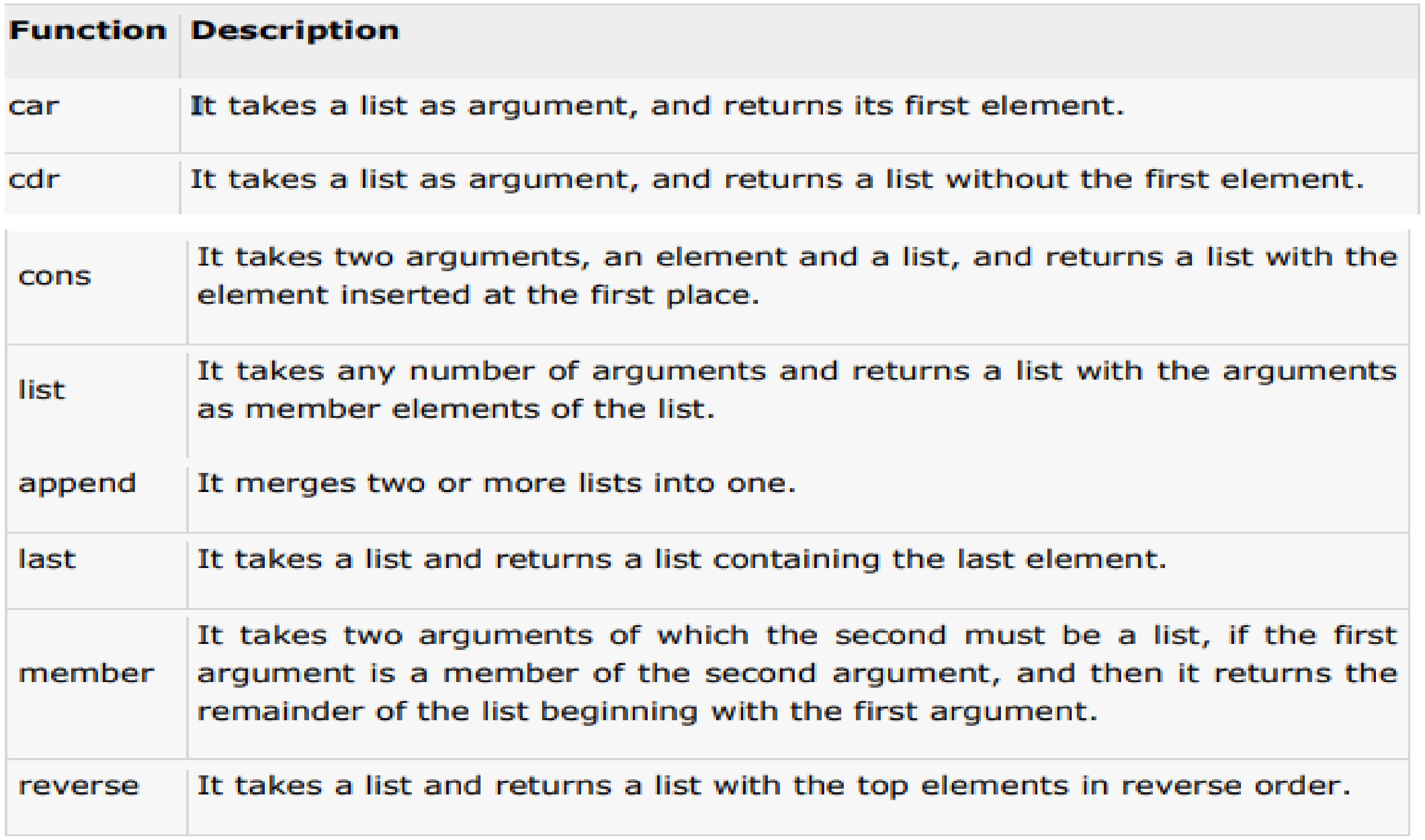
Pre define Functions

1. Boolean

	Description	
Atom	Returns True if argument is an atom	(atom 8) returns T
LISTP	Returns True if argument is an atom	(listp list(2 3) returns T
ZEROP	Returns True if argument is zero	
ODDP	Returns True if argument is an odd number	
EVENP	Returns True if argument is an even number	

List Manipulating Functions

The following table provides some commonly used list manipulating functions.



Function	Description
car	It takes a list as argument, and returns its first element.
cdr	It takes a list as argument, and returns a list without the first element.
cons	It takes two arguments, an element and a list, and returns a list with the element inserted at the first place.
list	It takes any number of arguments and returns a list with the arguments as member elements of the list.
append	It merges two or more lists into one.
last	It takes a list and returns a list containing the last element.
member	It takes two arguments of which the second must be a list, if the first argument is a member of the second argument, and then it returns the remainder of the list beginning with the first argument.
reverse	It takes a list and returns a list with the top elements in reverse order.

Input output Functions in LISP

	Description	Syntax
read	Used to take input from the user	(write (+3 (read)))
write	Returns True if argument is an atom	(write(+ 2 3))
print		(print “Hello”)
princ		(princ “Hello”)

User Defined Function

```
(defun f1(a b c)
(+ a b c))
```

Calling of this function

```
(f1 10 20 30)
```

Condition in LISP

```
(defun f2(a b)
( if(> a b)
    (print “ a is greater than b”)
    (print “a is less than b”)
)
)
```

Calling of this function

```
(f2 10 20)
```

Thank You