LISP
listp

- The function `listp` returns true if its argument is a list:

- `(listp '(a b c))`
- `(listp 123)`
- `(listp a)`
IF

- IF usually takes three arguments: a test expression, a then expression, and an else expression.

- `(if (listp 27) (+ 1 2) (+ 5 6))`
The logical operators and and or resemble conditionals. Both take any number of arguments, but only evaluate as many as they need to in order to decide what to return. If all its arguments are true (that is, not nil), then `and` returns the value of the last one:

```
(and t (+ 1 2))
```
Lab 1

- Q2
Special Form Function: **DEFUN**

**DEFUN**:

To bind names to expressions

e.g., `(DEFUN square (x) (* x x))

Example use: `(square 5)`
(defun sum-greater (x y z) (> (+ x y) z))
(defun test()
  (format t "CMPS 3500 Lab01 Name: santa claus~%")
  (format t "02. ") (dot2 5) (format t "~%")
)
(test)
FORMAT

- **format** produces formatted output by outputting the characters of control-string and observing that a tilde introduces a directive. The character after the tilde, possibly preceded by prefix parameters and modifiers, specifies what kind of formatting is desired. Most directives use one or more elements of args to create their output.

- If destination is a string, a stream, or t, then the result is nil. Otherwise, the result is a string containing the `output.'

- (format t “Here is my answer” )
- (format t "~%")
Lisp Loops

- **loop**
  - The **loop** construct is the simplest form of iteration provided by LISP. In its simplest form, it allows you to execute some statement(s) repeatedly until it finds a **return** statement.

- **loop for**
  - The **loop for** construct allows you to implement a for-loop like iteration as most common in other languages.

- **do**
  - The **do** construct is also used for performing iteration using LISP. It provides a structured form of iteration.
loop

- The loop construct is the simplest form of iteration provided by LISP. In its simplest form, it allows you to execute some statement(s) repeatedly until it finds a return statement.

- `(loop (s-expressions))`
(setq a 10)
(loop
  (setq a (+ a 1))
  (format t "~D" a)
  (format t "~%")
  (when (> a 17) (return a))
  )
(setq a 10)
(loop
 (setq a (+ a 1))
 (write a)    (format t "~D" a)
 (terpri)     (format t "~%")
 (when (> a 17) (return a))
 )
Loop for

- (loop for loop-variable in <a list>)
  - do (action)
  - )

- (loop for x in '(tom dick harry))
  - do (format t " ~s" x)
  - )
Loop for

- (loop for loop-variable in <a list>
  - do (action)
  - )

- (loop for a from 10 to 20
  - do (print a)
  - )
DO

- `(do ((variable1 value1 updated-value1)
-        (variable2 value2 updated-value2)
-        (variable3 value3 updated-value3)
-        ...)
-       (test return-value)
-       (s-expressions)
-       )`
(do ((x 0 (+ 2 x))
   (y 20 (- y 2)))
   ((= x y))
   (format t "~% x = ~d  y = ~d" x y))
)
Lab 1

- Q1
car / cdr

- The car of a list is the first element,
- The cdr is everything after the first element.

- `(car '(a b c))`
- `(cdr '(a b c))`
Lab 1

- Q3