Special Forms

- So far:
  Only considered functions
  which have a fixed number of args,
  and
  which "evaluate" these args

Eg: $\boxed{\texttt{CONS}}$: 2 args, both evaluated

`(CONS t (CONS 5 nil))` $\leadsto$ `(t 5)`

[NOT: `(t (CONS 5 nil))`]

$\boxed{\texttt{CAR}}$: 1 arg, which is evaluated

`(CAR (CONS 5 nil))` $\leadsto$ `5`

[NOT: `CONS`]

- Need other "function-like" things
  Do not "evaluate" their args, or
  Take n-args, any n

`Quote` “Function” Motivation

- Q: How to form list $\boxed{\text{(B)}}$?

  A: Apply `CONS` to

  form `?1` which evaluates to $\boxed{\text{B}}$ and

  form `?2` which evaluates to $\boxed{\text{nil}}$.

- Q: What FORM evaluates to $\boxed{\text{nil}}$?

  A: the atom `nil` —

  The value of `nil` is $\boxed{\text{nil}}$.

- Q: What FORM evaluates to $\boxed{\text{B}}$?

  A: ??

  Not the atom `B`.

  After `(setq B 5)`,

  the value of `B` is `5`.

`Quote` "Function"

- Answer: the form `(QUOTE B)`.

In gen'l: value of `(QUOTE` x`)` is x

for any s-expr x.

Value of	`(QUOTE B)`	is
Value of	`(QUOTE (foo A B))`	is
Value of	`(QUOTE (CONS 5 nil))`	is
Value of	`(CAR (QUOTE (A B)))`	is
Value of	`(QUOTE nil)`	is

$\rightarrow$ `(QUOTE B)`
`B`

$\rightarrow$ `(QUOTE (foo A B))`
`(foo A B)`

$\rightarrow$ `(QUOTE (CONS 5 nil))`
`(CONS 5 nil)`

$\rightarrow$ `(CONS 5 nil)`
`(5)`

$\rightarrow$ `(CAR (QUOTE (A B)))`
`A`

$\rightarrow$ `(QUOTE nil)`
`nil`

Comments on `Quote` "Function"

- `QUOTE` is NOT a (std) function:

  A function must EVALUATE its args;

  `QUOTE` does not.

- Abbreviation:

  `'B` for `(QUOTE B)`

```
→ (QUOTE B)
B

→ 'B
B

→ '(foo A B)
(foo A B)

→ '(CONS 5 nil)
(CONS 5 nil)

→ (CONS 5 nil)
(5)

→ (CAR '(A B))
A

→ 'nil
nil
```

Examples of `Quote` “Function”

(LAMBDA (x) x)	of	B	is	B
(LAMBDA (x) (QUOTE x))	of	B	is	x

→ <u>((LAMBDA (x) x) (QUOTE B))</u>

B

→ <u>((LAMBDA (x) (QUOTE x)) (QUOTE B))</u>

X

→ <u>((LAMBDA (X) X) 'B)</u>

B

→ <u>((LAMBDA (x) 'x) 'B)</u>

X

`List` "Function"

- Consider forming list

  `(A B t 5)`

- Can write as

  `(CONS 'A (CONS 'B (CONS t (CONS 5 nil))))`

- Shorthand:

  `(LIST 'A 'B t 5)`

- In gen'l:

  (LIST e_1 ...e_n) evaluates to ($\mathtt{s}_1$ ...$\mathtt{s}_n$)

  where each e_i is a form

  which evaluates to $\mathtt{s}_i$

  `List` can have any number of arguments!

  ... so is NOT a (std) function.

Examples of List "Function"

```
→ (LIST t 5)
(t 5)
→ (LIST (QUOTE BAC))
(BAC)
→ (LIST 'BAC)
(BAC)
→ (LIST 'PLUS 3 4)
(PLUS 3 4)
→ (LIST '(PLUS 3 4))
((PLUS 3 4))
→ (LIST (PLUS 3 4))
(7)
→ (CAR (LIST 'A 'B))
A
→ (CDR (LIST 'A 'B))
(B)
→ (QUOTE (LIST 'A 'B))
(LIST (QUOTE A) (QUOTE B))
```

Using `List` to Form λ-FORMs

→ <u>`(LIST 'LAMBDA '(x) (LIST 'CAR 'x))`</u>

`(LAMBDA (x) (CAR x))`

→ <u>`((LAMBDA (fn) (LIST 'LAMBDA '(x) (LIST fn 'x))) 'CAR)`</u>

`(LAMBDA (x) (CAR x))`

→ <u>`((LAMBDA (fn) (LIST 'LAMBDA '(x) (LIST fn 'x))) 'CDR)`</u>

`(LAMBDA (x) (CDR x))`

Other examples: p22.

CONDitional Expression #I
Absolute Value function

“If $x > 0$ then x else $-x$.”

Can write as

```
(LAMBDA (x)
   (COND ((> x 0) x     )
         ( t      (- x))) )
```

... of 5 is 5

... of -8 is 8

COND itional Expression II
Completely Defined EQuality Test

If	[not (atom x)]	then	nil
ElseIf	[not (atom y)]	then	nil
Else	(EQ x y).		

Can write as

```
(LAMBDA (x y)
   (COND ((NULL (ATOM x)) nil)
         ((NULL (ATOM y)) nil)
         ( t               (EQ X Y))) )
```

... of	A	and	A	is
... of	A	and	B	is
... of	A	and	(A)	is
... of	(A)	and	B	is
... of	(A)	and	(A)	is

EQuality Test w/Error Values

If	[not (atom x)]	then	error1
ElseIf	[not (atom y)]	then	error2
Else	(EQ x y).		

Can write as

```
(LAMBDA (x y)
   (COND ((NULL (ATOM x)) 'error1)
         ((NULL (ATOM y)) 'error2)
         ( t               (EQ X Y))) )
```

...	of	A	and	A	is
...	of	A	and	B	is
...	of	A	and	(A)	is
...	of	(A)	and	B	is
...	of	(A)	and	(A)	is

COND "Function"

```
(COND (⟨p1⟩ ⟨q1⟩)
      (⟨p2⟩ ⟨q2⟩)
         ...
      (⟨pn⟩ ⟨qn⟩) )
```

where each of the $\langle p_i \rangle$s and $\langle q_i \rangle$s are forms.

Evaluates like

```
If      ⟨p1⟩ then ⟨q1⟩
ElseIf  ⟨p2⟩ then ⟨q2⟩
ElseIf  ...
ElseIf  ⟨pn⟩ then ⟨qn⟩
Else    undefined.
```

Evaluating CONDitional Expressions

To **evaluate** $\sigma \stackrel{def}{=}$ `(COND (⟨p1⟩ ⟨q1⟩)`
`(⟨p2⟩ ⟨q2⟩) ...`
`(⟨pn⟩ ⟨qn⟩) )`

1. Evaluate $\langle \mathtt{p}_1 \rangle$

If non-**`nil`**,

Then evaluate $\langle \mathtt{q}_1 \rangle$ — this is σ's Value

Else

2. Evaluate $\langle \mathtt{p}_2 \rangle$

If non-**`nil`**,

Then evaluate $\langle \mathtt{q}_2 \rangle$ — this is σ's Value

Else

...

n. Evaluate $\langle \mathtt{p}_n \rangle$

If non-**`nil`**,

Then evaluate $\langle \mathtt{q}_n \rangle$ — this is σ's Value

Else

The Form σ is *undefined*.

COND "Function" (con't)

- Usually use $\langle \mathtt{p}_n \rangle = \mathtt{t}$
  so always defined.

[If $\langle \mathtt{p}_1 \rangle, \ldots, \langle \mathtt{p}_{n-1} \rangle$ all false,
then value of `(COND (⟨p1⟩ ⟨q1⟩) ··· ( t ⟨qn⟩) )`
is EVAL[$\langle \mathtt{q}_n \rangle$].]

(In FranzLisp: EVAL[$\langle \mathtt{p}_i \rangle$] = nil $\rightsquigarrow$
EVAL[(COND ($\langle \mathtt{p}_1 \rangle$ $\langle \mathtt{q}_1 \rangle$) ... ($\langle \mathtt{p}_n \rangle$ $\langle \mathtt{q}_n \rangle$))]
= nil)

- COND is not a function
  as its forms not all evaluated.

[If $\langle \mathtt{p}_1 \rangle$ is nil,
then $\langle \mathtt{q}_1 \rangle$ is not evaluated;
otherwise none of
$\langle \mathtt{p}_2 \rangle, \langle \mathtt{q}_2 \rangle, \ldots, \langle \mathtt{p}_n \rangle, \langle \mathtt{q}_n \rangle$
is evaluated.]

- Allows . . .

COND itional Expressions III

- Error-free Quotient:

```
  (LAMBDA (Y X)
    (COND ((eq X 0)   'Error-Db0)
          ( t        (Divides Y X) ) ))
```

  [Never ERROR/*undefined*].

- NULL Predicate

```
(LAMBDA (X)
  (COND (X nil )
        (t t   ) ) )
```

... of	nil	is	
... of	t	is	
... of	7	is	
... of	'(A B C)	is	

Boolean Utilities

- NOT $\approx$ NULL

  (Use NULL for Lists, NOT for Negation)

- (AND $\langle v_1 \rangle$ ... $\langle v_n \rangle$)

  Returns $\begin{cases} \texttt{NIL} & \text{if any term is } \texttt{NIL} \\ \text{or ...} & \text{value of last form.} \end{cases}$

```
(COND   ((NOT ⟨v1⟩)   nil)
              ...
        ((NOT ⟨vn⟩)   nil)
        ( t           ⟨vn⟩) )
```

- (OR $\langle v_1 \rangle$ ... $\langle v_n \rangle$)

  Returns $\begin{cases} \texttt{NIL} & \text{if all terms are } \texttt{NIL} \\ \text{or ...} & \text{value of first non}\texttt{NIL}\text{ form.} \end{cases}$

```
(COND   (⟨v1⟩   ⟨v1⟩)
           ...
        (⟨vn⟩   ⟨vn⟩)
        ( t     nil ) )
```

Examples of AND, OR

```
→ george
undefined variable ...blah blah blah

→ (and t nil)
nil

→ (and t nil george)
nil

→ (or t nil)
t

→ (or t nil george)
t

→ (and (atom 'tom) (null nil))
t

→ (or (atom '(a b)) (atom 'fred))
t

→ (or (and (listp 'tom) (atom 'tom))
        (or (atom '(a b)) (listp '(a))))

t
```

AND, OR — (con't)

```
→ (and 'fred 'george)
george
→ (or 'fred 'george)
fred
→ (or nil (list 5 4))
(5 4)
→ (and nil (divides 1 0))
nil
→ (and (not (eq n 0)) (divides 10 n))
nil
→ (or (eq n 0) (divides 10 n))
t
→ (or (eq m 0) (divides 10 m))
5
→ (or (eq m 0) (eq m 0))
nil
```