

CMPUT 325

Midterm Review

What you should know!

- Lisp
 - How to read Lisp Syntax
 - How to program in Lisp
 - Create Data/Nested Lists
 - Define functions
 - How Lisp is evaluated

What you should know.. continued

- Lambda Calculus
 - Evaluation order
 - Church encodings
 - Booleans and numbers

What you should know.. continued

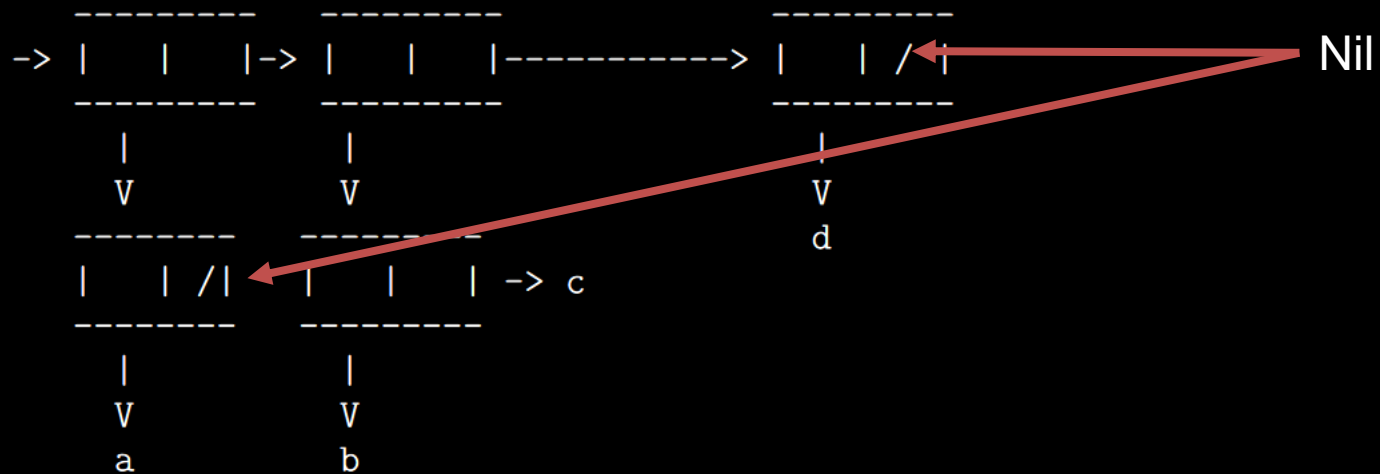
- Context/Closure Interpreter
 - How contexts are created
 - How closures are created

What you should know.. continued

- SECD
 - Compiling Simplified Lisp to SECD instructions
 - Running code on the SECD machine

Practice Midterm Q1a

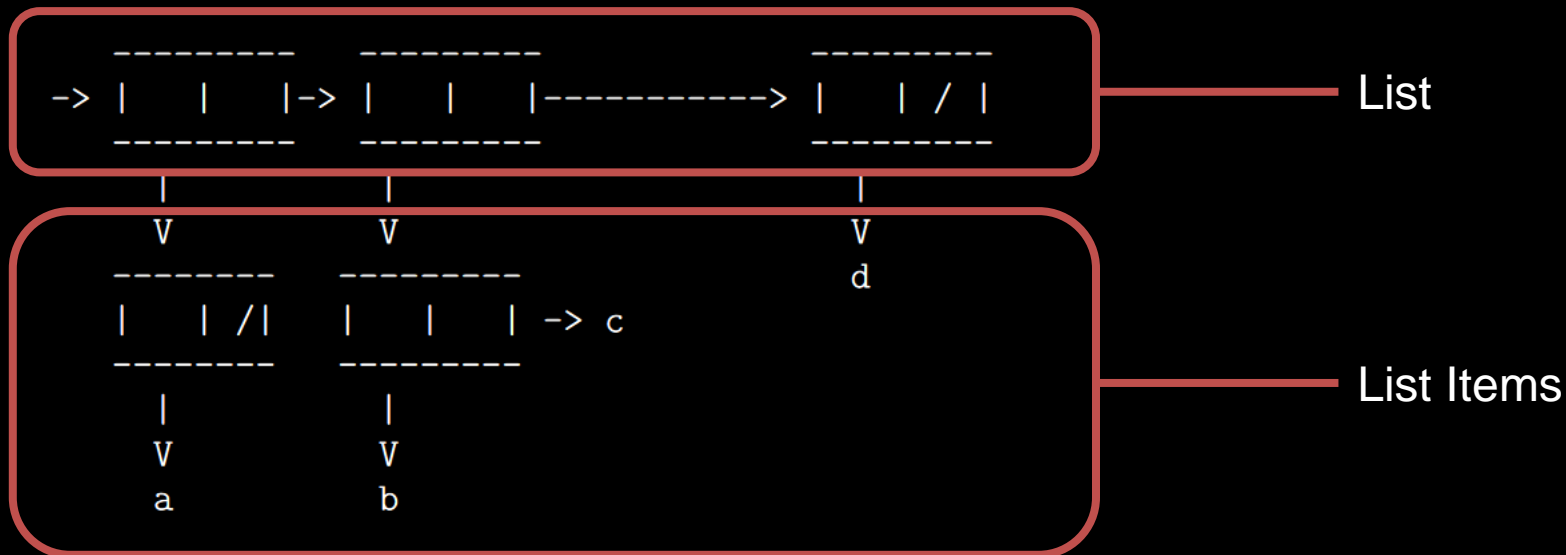
Show the simplest S-expression that is stored internally by the following structure



Practice Midterm Q1a

Show the simplest S-expression that is stored internally by the following structure

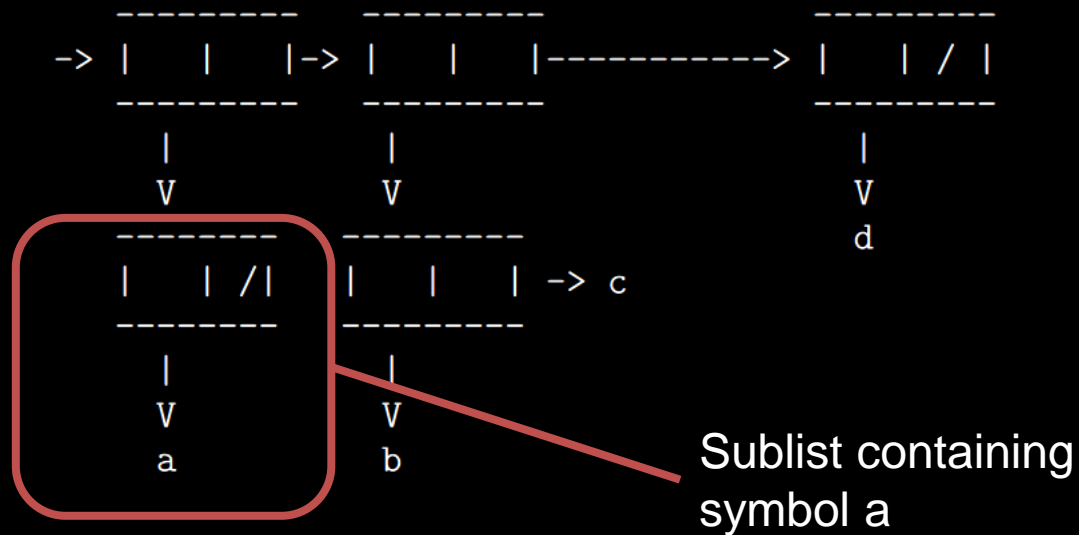
Answer looks like:
(☐ ☐ ☐)



Practice Midterm Q1a

Show the simplest S-expression that is stored internally by the following structure

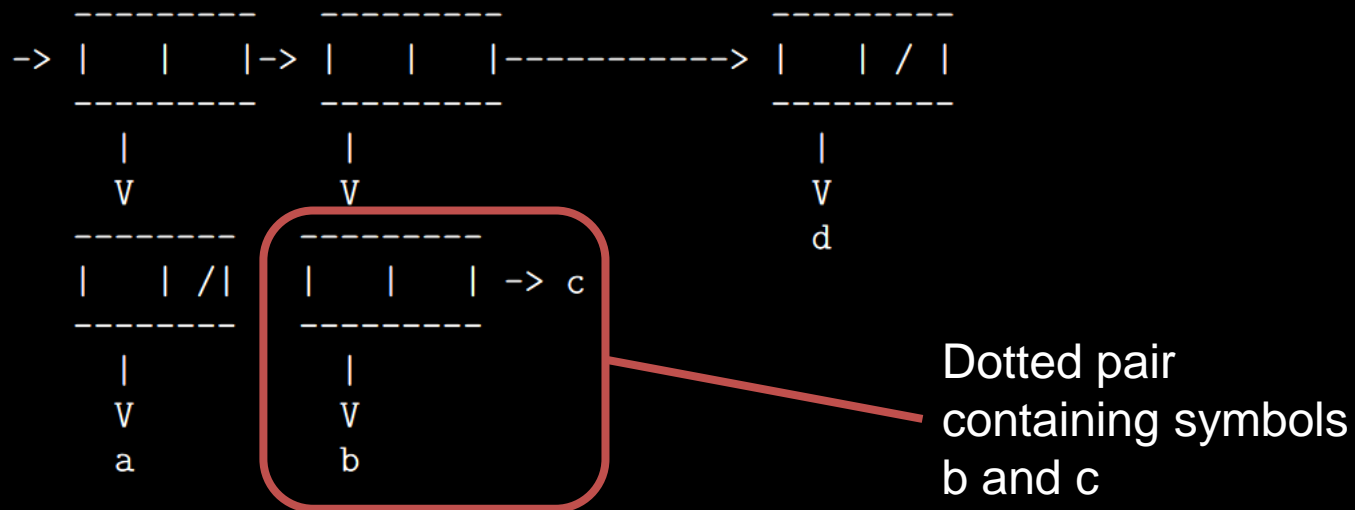
Answer looks like:
((a) □ □)



Practice Midterm Q1a

Show the simplest S-expression that is stored internally by the following structure

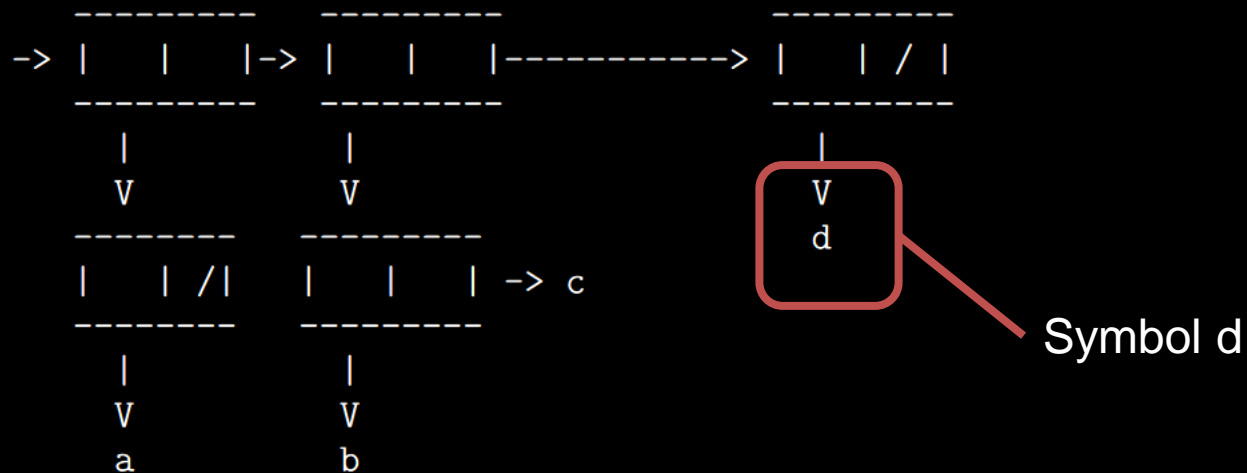
Answer looks like:
((a) (b . c) □)



Practice Midterm Q1a

Show the simplest S-expression that is stored internally by the following structure

Answer looks like:
((a) (b . c) d)



Practice Midterm Q1b

Draw the machine level representation of the following expression

```
(defun max (X Y)
  (if (> X Y)
      X
      Y))
```

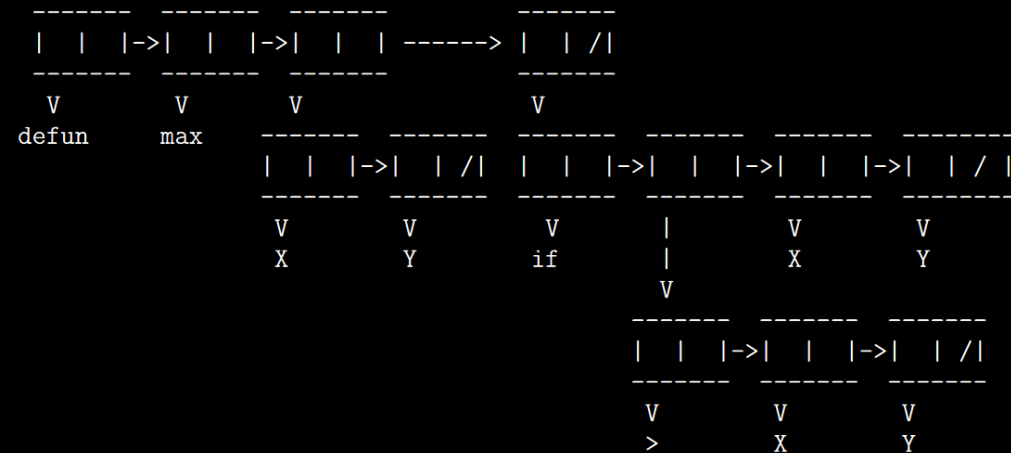
- List containing
 - defun
 - max
 - List (X Y)
 - List (if (> X Y) X Y)

Practice Midterm Q1b

Draw the machine level representation of the following expression

```
(defun max (X Y)
  (if (> X Y)
      X
      Y))
```

- List containing
 - defun
 - max
 - List (X Y)
 - List (if (> X Y) X Y)



Practice Midterm Q1c

Lisp code to return sublist ($> X Y$) of the following

```
(set 'a  
    '(defun max  
        (X Y)  
        (if (> X Y)  
            X  
            Y))  
)
```

- Fourth item
 - (if ($> X Y$) X Y)
- Second item of that
 - ($> X Y$)
- Answer:
 - (cadr (caddr a))
 - ↑ Drop 1 and take first (second)
 - ↑ Drop 3 and take first (fourth)

Practice Midterm Q2

Simplify

$F T F a (NOT F b c)$

- Tips
 - Brackets can be helpful
 - Booleans take two arguments
- Answer:
 - $((F T F) a ((NOT F) b c))$
 - $(F a ((NOT F) b c))$
 - $((NOT F) b c)$
 - $(T b c)$
 - b

Practice Midterm Q3

Define a lambda expression for

| X | Y | OP | X | Y |
|---|---|----|---|---|
| T | T | | T | |
| T | F | | F | |
| F | T | | F | |
| F | F | | T | |

- If X is true then OP returns y
- If X is false then OP return negation of Y
- Answer:
 - $(\lambda xy \mid xy(\text{NOT}y))$

Practice Midter Q4

Compile the following to SECD

$(- (* 5 4) (+ 2 4))$

- SECD evaluates arguments right to left
 - Code for $(+ 2 4)$
 - Code for $(* 5 4)$
 - Code for $-$
- Answer:
LCD 4 LCD 2 +
LCD 4 LCD 5 *
-

Context and Closure based Interpreter

- Class notation
 - $\text{eval}[e, n, v]$
 - e is an expression
 - n is a name list
 - v is a value list
- Alternate notation
 - $\text{eval } e \text{ in CT}$
 - e is an expression
 - CT is a context
- Think of n and v as being a context CT
 - $n = (x \ y \ z \ a \ b \ x)$
 - $v = (1 \ 2 \ 3 \ 4 \ 5 \ 6)$
 - $\text{CT} = \{x \rightarrow 1, y \rightarrow 2, z \rightarrow 3, a \rightarrow 4, b \rightarrow 5, x \rightarrow 6\}$

Closures

- Lambda function together with a context
- Written as $[(\text{lambda } \textit{params body}), \text{CT}]$
- Evaluation:
 eval
 (lambda params body)
 in
 CT
 -> $[(\text{lambda } \textit{prams body}), \text{CT}]$

Evaluating Closure Applications

```
eval
  (f e1 ... en)
in
  CT0 = { f -> [(lambda (x1 ... xn) body), CT] ... }
->
eval
  body
in
  {x1 -> (eval e1 in CT0), ..., xn -> (...)} U CT
```

Practice Midterm Q5

```
((lambda (f x y) (f (f x y) z))  
 (lambda (w v) (+ w v)) 3 5)
```

- Evaluation contexts for
 - $(f (f x y) z)$
 - $(f x y)$
 - First call $(+ w v)$
 - Second call $(+ w v)$

- $CT1 =$
 $\{f \rightarrow [(\lambda (w v) (+ w v)), CT0]$
 $, x \rightarrow 3$
 $, y \rightarrow 5\} \cup CT0$
- $CT1$
- From $(f x y)$
 - eval $(+ w v)$
in $\{w \rightarrow 3, v \rightarrow 5\} \cup CT0$
 - evaluates to 8
- From $(f (f x y) z)$
 - eval $(+ w v)$
in $\{w \rightarrow 8, v \rightarrow 4, CT0\}$

Practice Midterm Q6a

Lisp Program

```
(defun f (L1 L2)
  (if (null L1)
      (let ((s (g L2))) (- 0 s))
      (+ (car L1) (f (cdr L1) L2))))
(defun g (L)
  (cond
    ((null L) 0)
    ((null (cdr L)) (car L))
    (t (+ (cadr L) (g (cddr L))))))
```

Results for

- `(g '(1 2 3))`
→ `(+ 2 (g '(3)))`
→ `(+ 2 3)`
→ `5`

Practice Midterm Q6b

Lisp Program

```
(defun f (L1 L2)
  (if (null L1)
      (let ((s (g L2))) (- 0 s))
      (+ (car L1) (f (cdr L1) L2))))
(defun g (L)
  (cond
    ((null L) 0)
    ((null (cdr L)) (car L))
    (t (+ (cadr L) (g (cddr L))))))
```

Results for

- `(g '(1 2 3 4))`
-> `(+ 2 (g '(3 4)))`
-> `(+ 2 (+ 4 (g nil)))`
-> `(+ 2 (+ 4 0))`
-> `6`

Practice Midterm Q6

Lisp Program

```
(defun f (L1 L2)
  (if (null L1)
      (let ((s (g L2))) (- 0 s))
      (+ (car L1) (f (cdr L1) L2))))
(defun g (L)
  (cond
    ((null L) 0)
    ((null (cdr L)) (car L))
    (t (+ (cadr L) (g (cddr L))))))
```

Results for

- $(f \text{ '(1 2 3) '(2 4)})$
→ $(+ 1 (f \text{ '(2 3) '(2 4)}))$
→ $(+ 1 (+ 2 (f \text{ '(3) '(2 4)})))$
→ $(+ 1 (+ 2 (+ 3 (f \text{ nil '(2 4)}))))$
 $(g \text{ '(2 4)})$
 → $(+ 4 (g \text{ nil}))$
 → 4
→ $(+ 1 (+ 2 (+ 3 (- 0 4))))$
→ 2

Practice Midterm Q6

Lisp Program

```
(defun f (L1 L2)
  (if (null L1)
      (let ((s (g L2))) (- 0 s))
      (+ (car L1) (f (cdr L1) L2))))
(defun g (L)
  (cond
    ((null L) 0)
    ((null (cdr L)) (car L))
    (t (+ (cadr L) (g (cddr L))))))
```

Results for

- `(f '(5 3 6) '(1 4 2))`
→ `(+ 5 (+ 3 (+ 6 (f nil '(1 4 2)))))`
 `(g '(1 4 2))`
 → `(+ 4 (g '(2)))`
 → `(+ 4 2)`
 → `6`
→ `(+ 5 (+ 3 (+ 6 (- 0 6))))`
→ `8`

Practice Midterm Q7-Q11

- Q7-Q8: More Lisp Evaluation
- Q9-Q11: Lisp Programming
- Questions?