UCLA Samueli School of Engineering

CS161 WEEK1 DISCUSSION 1C

Danfeng G
Contributed by Yewen W's previous course materials

Agenda

1. ADMINISTRATIVE ISSUES
2. LISP

About CS161 Dis1C

- TA: Danfeng(Lyle) Guo
- Contact: lyleguo101@gmail.com
  - Please indicate 'CS161' in the title
- Discussion session: 2-4 pm Friday
- Office hour: 4-6 pm Wednesday
- CampusWire
- Everything will be uploaded later on CCLE

About CS161 Dis 1C

WHAT DO WE DO IN DISCUSSION SESSION

1. Only key parts of lectures
2. Try some problems
3. Answer questions

Any questions?

LISP

Get prepared for all your homeworks

LISP

TO RUN LISP

1. LISP online: https://jscl-project.github.io
2. SEASnet
   - Open your SEASnet account. Set your ucla vpn if your are not in campus wifi.
   - Login: ssh -X lnxsrv.seas.ucla.edu -l yourSEASaccount
   - Copy file: scp –r localpath yourSEASaccount@lnxsrv.seas.ucla.edu:path
   - Run: clisp myfile.lsp (or just use clisp to open the interface)

LISP

A FEW THINGS ABOUT LISP

LISP tutorial: https://www.tutorialspoint.com/lisp/

Basic types: numbers, chars, symbols

Data structures: lists, strings

Atom: anything not a list + ‘nil’

Expression: (operator arg1 arg2 …)

eg: (+ 1 2), (sum_list ' (1 2 3))

LISP

BASIC OPERATIONS

Math operator: +, -, *, /

Comparison operator: =, >, <, /=, >=, <=

Logical operator: and, not, or

Print values: print, format

LISP

EQUALITY

- For numbers: =
    - (= 3 3)
    - (= a b)
- For object identities: eql
    - (eql 3 3)
    - (eql 'q 'q)
    - (eql (list 3 4) (list 3 4))
- For lists and strings: equal
    - (equal (list 'a 'b 'c) (list 'a 'b 'c))

LISP

VARIABLES

- Global variable:
  - defparameter: Eg: (defparameter name 'John)
  - defvar: Eg: (defvar name 'John)
  - defvar assigns values only once
- Local variable
  - let: (let ((var1 val1) (var2 val2) ..) (expr))
  - Eg: (let ((a 1) (b 1)) (+ a b))

LISP

LIST

- Create a list
  - Use quote: '(1 2 3)
  - Use list: (List 1 2 3)

LISP

LIST

- cons: append one element to the front of list (cons elm1 lst2)
  - elm1 will be an element of lst2
  - lst2 can be a number/nil/list
  - Be clear on this because lots of bugs come from it.
- append: concatenate two lists (append lst1 lst2)
  - Both arguments should be lists
- car: return the first element of a list (car lst)
- cdr: return the remaining parts except for the first one (cdr lst)
  - It returns a list of elements
  - What if (cdr '(1))?

LISP

SOME CODING PRACTICES

What will be the results of

1. (cons '(1 2) '(3 4))
2. (list 2 (cons 3 nil))
3. (append 3 ’(4 5))
4. (car (cons 5 ’(6 7)))
5. (cdr (append ’(1 2) ’(3 4)))

LISP

FUNCTION

- defun:
    - defun functionName (arg1 arg2 ..) (to do)
    - Eg: (defun hello (name) (format nil "Hello, ~A" name))

UCLA Samueli School of Engineering

LISP

FLOW CONTROL

if (test expr) (then expr) (else expr)

Eg: (if (> 3 1) (print “yes”) (print “no”))

```
cond (
        ((case1) todo)
        ((case2) todo)
        …
        …
        (t todo)
)
```

LISP

LOOP

```
(loop for x in '(1 2 3 4 5) do (print x) )
```

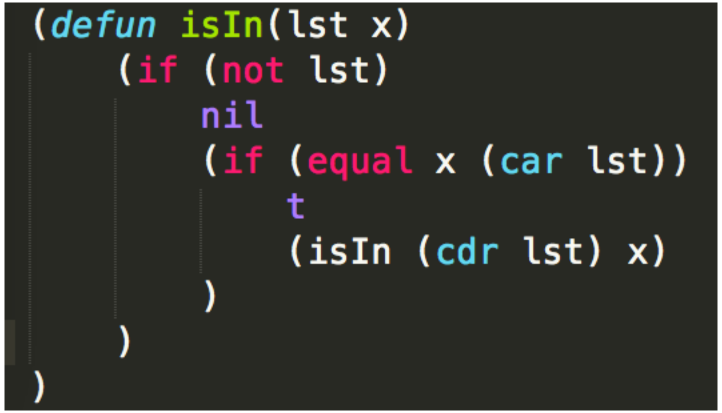
We do not suggest it for this course

LISP

RECURSION

Please use recursion instead of loop as much as you can

Eg: check whether an element is in list



```
(defun isIn(lst x)
    (if (not lst)
        nil
        (if (equal x (car lst))
            t
            (isIn (cdr lst) x)
        )
    )
)
```

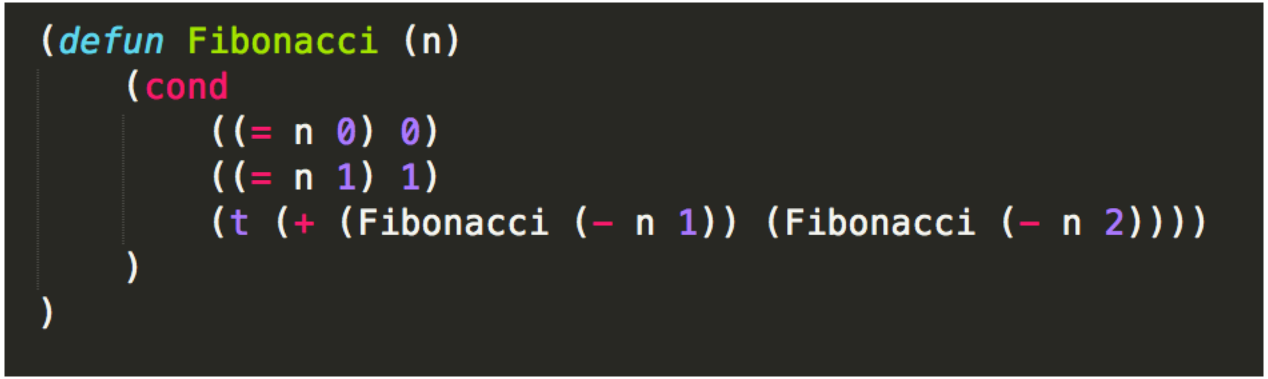
LISP

PRACTICE!

Implement Fibonacci(n) to generate the n-th Fibonacci number

$F(0) = 0$, $F(1) = 1$

$F(n) = F(n - 1) + F(n - 2)$, for $n > 1$.



```
(defun Fibonacci (n)
    (cond
        ((= n 0) 0)
        ((= n 1) 1)
        (t (+ (Fibonacci (- n 1)) (Fibonacci (- n 2))))
    )
)
```

HW1

SOME ISSUES TO CONSIDER

- They are almost all about recursion
- Sometimes make use of previous answers
- Take care of ‘nil’. It is both an item and a list

Q&A

UCLA Samueli School of Engineering