CS 161 Discussion Week 1

TA: Honghua Zhang

10/01/2021

Basic Info

- Syllabus and Slides: CCLE
- Campuswire:
  - https://campuswire.com/p/G2EF3B1E3
  - Access code: 0628
- Office Hour:
  - Friday 9:00 – 11:00 AM
  - Zoom Link: https://zoom.us/j/8934041037?pwd=cDM3MnN0VEdlSzBrVEZHUmF4cm5wZz09
  - If expired, refer to CCLE for the up-to-date link.

Grading

- Grading
  - Homework 20%
  - Midterm 35%
  - Final 40% (multiple-choice only)
- Late Policy
  - -25% of total score each day lateFinal 40%

CLISP

- SEASNet linux server
- To setup on your own machine:
  - https://clisp.sourceforge.io
- Online
  - https://jscl-project.github.io/

Atom

```
30          ; => 30
“Hello!”    ; string

t           ; denoting true
nil         ; false; the empty list: ()
```

any non-NIL value is true!

Atom

```
9999999999999999999999 ; integer
#b111                  ; binary => 7
#x111                  ; hexadecimal => 273
3.14159so             ; single
3.14159do              ; double
1/2                    ; ratios
#C(1 2)                 ; complex numbers
```

Basic arithmetic operations

```
(+ 1 1)                ; => 2
(- 8 1)                ; => 7
(* 10 2)               ; => 20
(expt 2 3)             ; => 8
(mod 5 2)              ; => 1
(/ 35 5)               ; => 7
(/ 1 3)                ; => 1/3
(+ #C(1 2) #C(6 -4))   ; => #C(7 -2)
```

Booleans and Equality

```
(not nil)              ; => T
(and 0 t)              ; => T
(or 0 nil)             ; => 0 (T)
(and 1 ())             ; => NIL
     empty list
```

Lists

- Linked-list data structures
  - Struct node
    - Val
    - Next pointer

- Made of CONS pairs

```
(cons 1 2)                         ; => '(1 2)
(cons 3 nil)                       ; => '(3)
(cons 1 (cons 2 (cons 3 nil)))     ; => '(1 2 3)
(list 1 2 3)                       ; => '(1 2 3)
(cons 4 '(1 2 3))                  ; => '(4 1 2 3)
(cons '(4 5) '(1 2 3))             ; =>  ?
```

Lists

```
(cons 1 (cons 2 (cons 3 nil)))     ; => '(1 2 3)
(list 1 2 3)                       ; => '(1 2 3)
(cons 4 '(1 2 3))                  ; => '(4 1 2 3)
(cons '(4 5) '(1 2 3))             ; => '((4 5) 1 2 3)

(append '(1 2) '(3 4))             ; => '(1 2 3 4)
(append 1 '(1 2))                  ; ERROR!
(concatenate 'list '(1 2) '(3 4))  ; => '(1 2 3 4)

(car '(1 2 3 4))                   ; => 1
(cdr '(1 2 3 4))                   ; => '(2 3 4)
car and cdr should be used for list
```

Functions

- Define a function

```
(defun hello (name) (format nil "Hello, ~A" name))
```

- Call the function

```
(hello "Bob")             ; => "Hello, Bob"
```

Control Flow

```
(if (equal *name* "bob")   ; test expression
    "ok"                   ; then expression
    "no")                  ; else expression
```

- Chains of tests: cond

```
(cond ((> *age* 20) "Older than 20")
      ((< *age* 20) "Younger than 20")
      (t "Exactly 20"))

(cond ((> *age* 20) "Older than 20")
      ((< *age* 20) "Younger than 20")) ; NIL when *age*=20
```

Programming Practice!

- Factorial
- compute list length
- find kth element
- delete kth element

Factorial

```
(defun factorial (n)
  (if (< n 2)
    1                  ; returns 1 when n<2
    (* n (factorial (- n 1)))   ; when n>=2
   )
)

(factorial 5)                   ; => 120
```

Compute list length

```
'((a b) (c (d 1)) e)  => 3

(defun listlength (x)
     (if (not x) ; base case: empty list
          0
          (+ (listlength (cdr x)) 1)
     )
)      '(1 2 3 4) -> '(2 3 4)
```

Find kth element (top-level)

```
(defun find_kth (k x)
       (if  (= k 1)
             (car x)
             (find_kth (- k 1) (cdr x))
       )
)
```

How do we find kth element in the flattened list?

```
3, '((a b) (c (d 1)) e)   => c
```

Delete kth element

```
(defun delete_kth (k x)
       (if  (= k 1)
             (cdr x)
             (cons (car x)
                     (delete_kth (- k 1) (cdr x))
             )
       )
)
```