CS 230
Programming Languages

10 / 27 / 2022

Instructor: Michael Eckmann

Today's Topics

- Questions? / Comments?
- More Scheme

Scheme

- Some key things we learned last time
- define
  - Examples:
  - (define games 162)
  - (define (square x) (* x x))
- car, cdr
  - Examples:
  - (car ‘(a b c)) ; ‘a
  - (cdr ‘(a b c)) ; ‘(b c)
- cons
  - Example:
  - (cons ‘a ‘(b c)) ; ‘(a b c)
- null?
  - Examples:
  - (define list1 ‘(a b c))
  - (null? list1) ; #f
  - (null? ‘()) ; #t
- quote or ‘

Functional Programming

- Scheme
  - Numeric “predicate” functions return a Boolean
  - #T or #F
  - =, <>, >, <, >=, <=, EVEN?, ODD?, ZERO?

  - Symbolic Atoms and Lists “predicate” functions
  - EQ?, LIST?, NULL?, EQUAL?

Functional Programming

- Scheme
  - There is a difference between
    - =
    - eq?
    - equal?
  - = is used for numeric comparison
  - eq? is used for symbol/atom comparison
  - equal? is used for symbol/atom or list comparison

Functional Programming

- Scheme
  - IF takes three parms

(IF predicate then_expression else_expression)

  - COND – is like switch / case statements

(COND
(predicate1 expression { expression })
(predicate2 expression { expression })
...
(predicaten expression { expression })
(ELSE expression { expression })
)

Functional Programming

- Let's write a function named linsearch that determines whether an atom is in a particular list.
- e.g. (linsearch 'B '(A B C)) should return #T and

(linsearch 'X '(A B C)) should return #F.)

Functional Programming

- solution

```
(DEFINE (linsearch atm lis)
   (COND
         ((NULL? lis) #F)
         ((EQ? atm (CAR lis)) #T)
         (ELSE ( linsearch atm (CDR lis)))
   )
)
```

Functional Programming

- when writing functions dealing with a list lis,
  - if you want to go through the elements one at a time,
    - use (car lis) to get the first
    - and recurse on (cdr lis)
    - also, you'll need to tell your function when to stop, that is, when the list is null, so (null? lis) should be the first thing you check.
- In general recursive functions need to
  - 1st check the base case (the case that will be true when the recursion should stop)
  - and in the recursive step(s) make sure that the function is working towards the base case (What do I mean here?)

Functional Programming

- Let's write functions to:
  - compute a fibonacci number
    - we'll write it with ifs
    - we'll write it with cond
  - power – raise the value of the first parameter to the second.

Functional Programming

- Let's write functions to:

memberdeep - of an atom in a list (or any of the “sublists”)

- Let's write this so that it checks for membership of the atom in the list as before, or in any list within the list.

atomsonly – determine if a list is made up of only atoms (no sublists).

– Fix power to handle negative exponents

Functional Programming

- Let's write functions to:
  - remove the first occurrence of an element of a list