;;Constructor: make new objects

;;leaf constructor for simple binary tree

(define leaf

(lambda (x) x))

(define interior-node2

(lambda (symbol tree1 tree2)

(append (cons symbol (list tree1)) (list tree2))))

;;interior node will a list with three things inside (a symbol and 2 binary tree)... => list can take a lot of parameters and turn them into a list

(define interior-node

(lambda (symbol tree1 tree2)

(list symbol tree1 tree2)))

;; check if the leaf is a number ==> boolean true or false

(define leaf?

(lambda (x)

(number? x)))

;;left-son

(define lson

(lambda (binTree)

(car (cdr binTree))))

;;right-son

(define rson

(lambda (binTree)

(car (cdr (cdr binTree)))))

;;contents-of only gets the symbol

(define contents-of

(lambda (binTree)

(car binTree)))

(define testTree

(interior-node 'red

(interior-node 'bar

(leaf 26)

(leaf 12))

(interior-node 'red

(leaf 11)

(interior-node 'quux (leaf 117) (leaf 14)))))

(define count

(lambda (binTree)

(if (leaf? binTree)

1

(+ (count (lson binTree)) (count (rson binTree)) ))))

(define count-leaves

(lambda (binTree)

(count-leaves-helper binTree 0)))

(define count-leaves-helper

(lambda (binTree num)

(cond

((leaf? binTree)

(leaf num))

((leaf? (lson binTree))

(interior-node

(contents-of binTree)

(leaf num)

(count-leaves-helper (rson binTree)

(+ (count (lson binTree)) num))))

((leaf? (rson binTree))

(interior-node

(contents-of binTree)

(count-leaves-helper (lson binTree) num)

(leaf (+ (count (lson binTree)) num))))

(else

(interior-node

(contents-of binTree)

(count-leaves-helper (lson binTree) num)

(count-leaves-helper (rson binTree)

(+ (count (lson binTree)) num)))))))

(define tree

(interior-node 'lol 19 (interior-node 'lol 56 47)))

(define tree1

(interior-node 'lol 19 29))

(define tree2

(interior-node 'lol (interior-node 'lol 56 47) 19))

(define tree3

(interior-node 'lol (interior-node 'lol 19 29) (interior-node 'lol 56 47)))