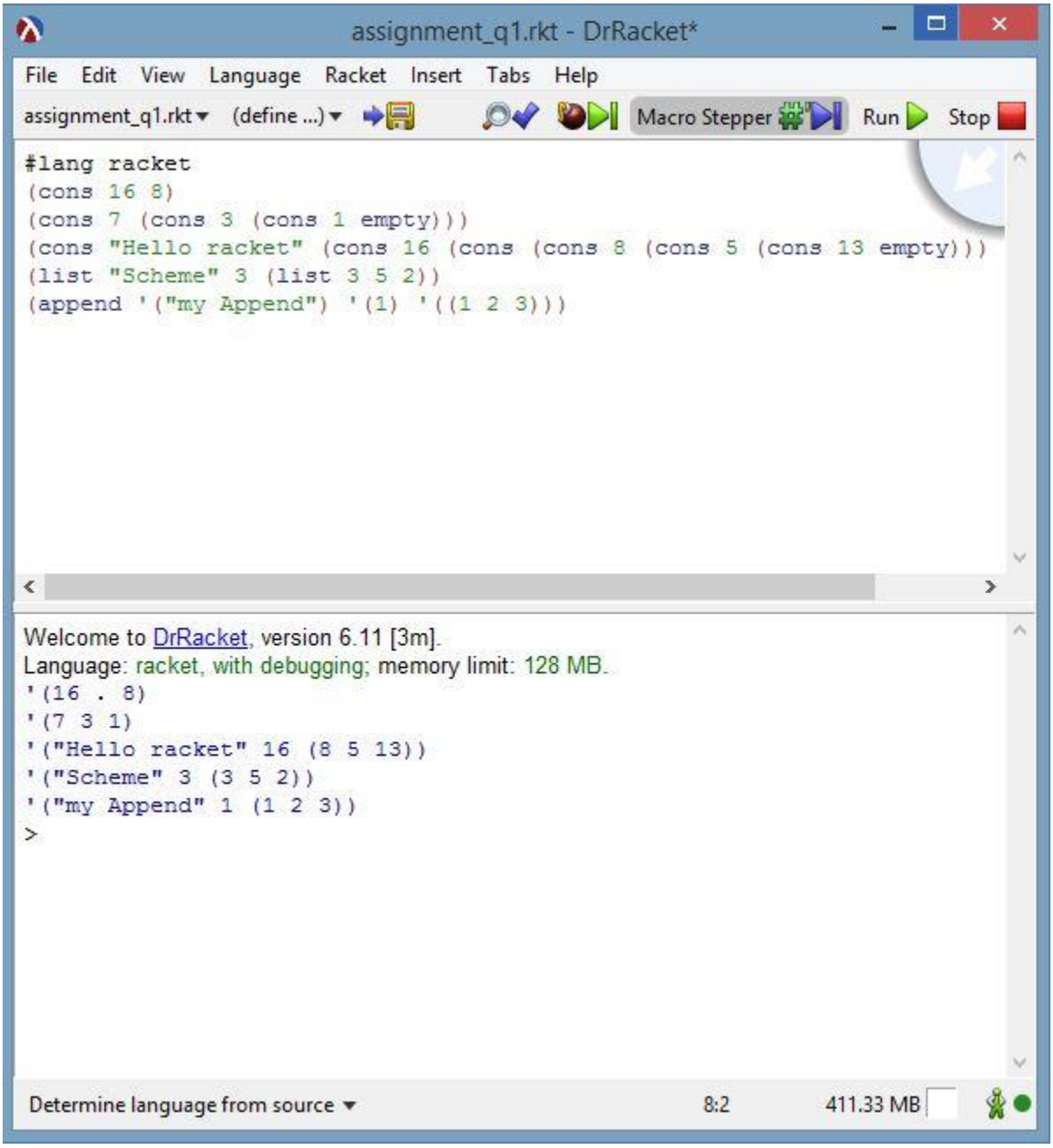
CT331 Assignment 2

Functional Programming with Scheme

Chika Onyia
15736825

Question 1
Screenshot



Question 1 Comments

Cons function allows you to combine two notations, example “x” “y” or simply S-expression into a single pair.

List on the other hand is implemented on top of cons can take any number of cons pairs and combine them into one list.

While the Append function like List can take multiple cons values, but or a List values simply copy the list into new block.

Question 2

assignment_q2.rkt - DrRacket

```
#lang racket
(provide ins_beg)
(provide ins_end)
(provide cout_top_level)
(provide count_instances)
(provide count_instances_tr)
(provide count_instances_deep)

(define (ins_beg el symbol)
  (display "Hello, I'm ins_beg!\n")
  (cons el symbol))

(define (ins_end el symbol)
  (display "Hello, I'm ins_end!\n")
  (append symbol (list el)))

(define (cout_top_level list)
  (display "Hello, I'm cout_top_level!\n")
  (define lst_el 0)
  (define (count symbol)
    (if (empty? symbol)
        (display "Finished counting elements! = ")
        (begin
          (set! lst_el (+ lst_el 1))
          (count (rest symbol)))))
  (count list)
  (display lst_el))

(define (count_instances el list)
  (display "Hello I'm count_instances!\n")
  (define lst_el 0)
  (define (count symbol)
    (if (empty? symbol)
        (display "Finished counting instances! = ")
        (begin
          (if (eq? el (first symbol))
              (set! lst_el (+ lst_el 1))
          (set! lst_el lst_el))
          (count (rest symbol)))))
  (count list)
  (display lst_el))


(define (count_instances_tr el list)
  (display "Hello I'm count_instances_tr!\n")
  (define (count_tr symbol totalSymbol)
  (if (empty? symbol)
      (begin
        (display "Finished counting instances! = ")
        (display totalSymbol))
      (begin
        (if (eq? el(first symbol))
            (count_tr (rest symbol) (+ totalSymbol 1))
            (count_tr (rest symbol) totalSymbol)))))
  (count_tr list 0))


  ;I tried to run this function using characters like the rest, but kept getting error unless I use numbers
  ;if you dont mind I will like to know how to refoctor this function to run with characters (just for the future)
(define (count_instances_deep el list)
  (display "Hello I'm count_instances_tr!\n")
  (define lst_el 0)
  (define (count symbol)
    (if (empty? symbol)
        (void)
        (begin
          (if (eq? el (first symbol))
              (begin
                (set! lst_el (+ lst_el 1))
                (count (rest symbol)))
          (begin
            (if (number? (first symbol))
                (count (rest symbol))
                 (begin
                   (count (first symbol))
                   (count (rest symbol))))))))

  (count list)
  (display "Finished counting instances! = ")
  (display lst_el))
```

Welcome to DrRacket, version 6.11 [3m].
Language: racket, with debugging; memory limit: 128 MB.
>

Question 3

assignment_q3.rkt - DrRacket

File Edit View Language Racket Insert Tabs Help

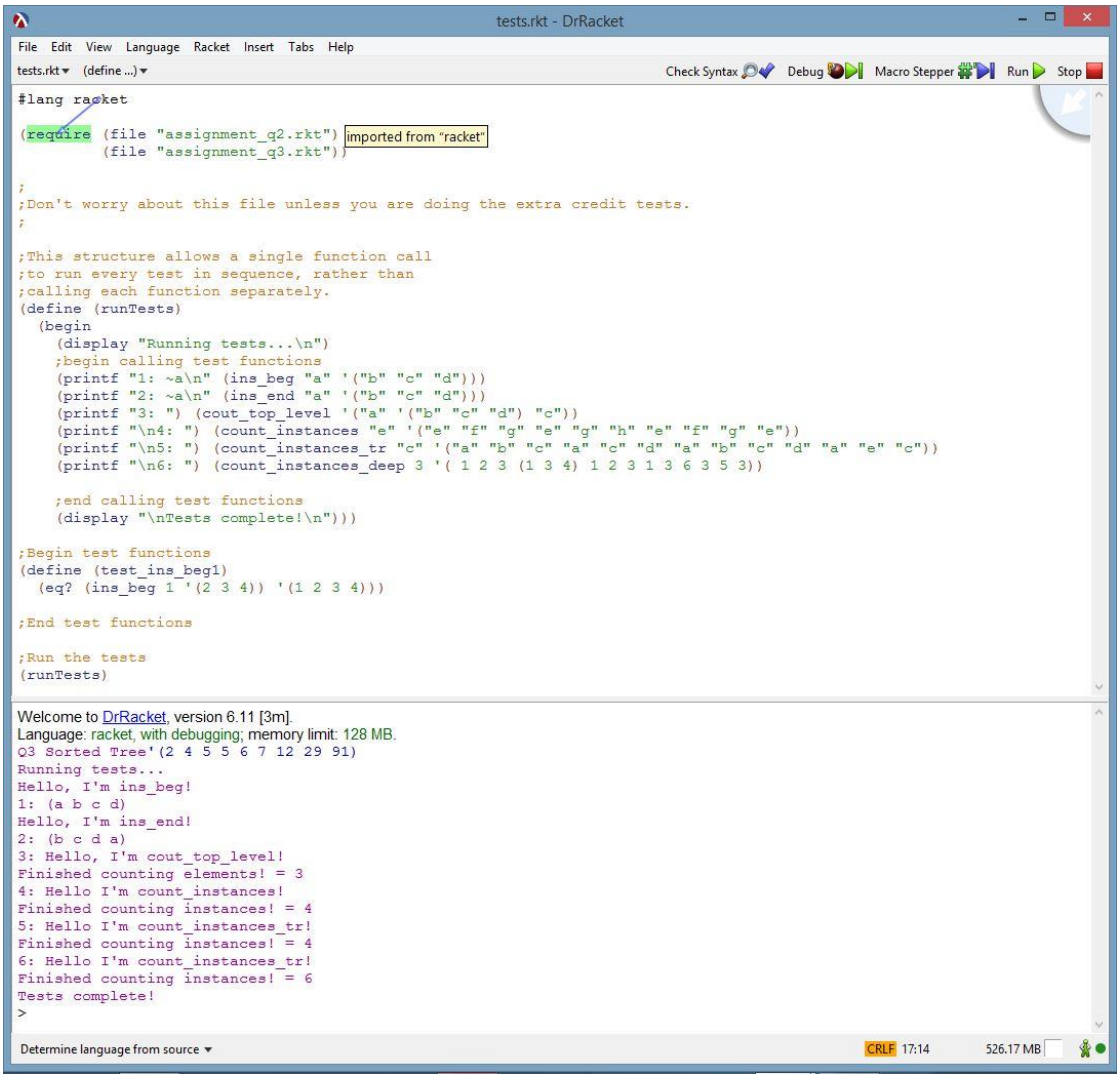
assignment_q3.rkt (define ...) Check Syntax Debug Macro Stepper Run Stop

```
#lang racket
(provide leftChild)
(provide rightChild)
(provide value)
(provide sorted)
(provide itemExists)
(provide treeSort)
(provide insertItem)
(provide insertList)
(provide inOrder)


;Defining the structure of the BST
  (define (leftChild BST)
        (car BST))
  (define (rightChild BST)
    (caddr BST))
  (define (value BST)
    (cadr BST))

  (define inOrder
    (match-lambda
      [`(() ,el ()) `(,el)]
      [`(,left ,el ()) (append (inOrder left) `(,el))]
      [`(() ,el ,right) (append `(,el) (inOrder right))]
      [`(,left ,el ,right) (append (inOrder left) `(,el) (inOrder right))]))

;Part (A)
  (define (sorted BST);
   (begin
     (cond [(not (empty?(leftChild BST))) (sorted (leftChild BST))])
     (printf "~a " (value BST));
     (cond [(not (empty?(rightChild BST))) (sorted (rightChild BST))])))

;Part (B)
   (define (itemExists item BST)
      (cond
        [(null? BST) #f]
        [(equal? item (value BST)) #t]
        [(< item (value BST)) (itemExists item (leftChild BST))]
        [(> item (value BST)) (itemExists item (rightChild BST))]))

;Part (C)
    (define insertItem
        (match-lambda**
         [(el '()) `(() ,el ())]
         [(el '(() () ())) `(() ,el ())]
         [(el `(,left ,item ,right)) #:when (<= el item) `(,(insertItem el left) ,item ,right)]
         [(el `(,left ,item ,right)) `(,left ,item ,(insertItem el right))]))

;Part (D)
    (define
      (insertList BST leaf)
      (if (empty? leaf) BST
          (insertList (insertItem BST (car leaf)) (cdr leaf))))

;Part (E)
      (define (treeSort lst)
        (define treeSortItration
          (match-lambda**
           [(el `()) (inOrder el)]
           [(el `(,a . ,b)) (treeSortItration (insertItem a el) b)]
           [(_ _) "incorrect arguments or broken tree"]))
        (treeSortItration '(() () ()) lst))

      ;Sample test list
      (display "Q3 Sample Tree")
      (treeSort '(5 2 7 91 5 12 29 4 6))
```

```
Welcome to DrRacket, version 6.11 [3m].
Language: racket, with debugging; memory limit: 128 MB.
Q3 Sample Tree'(2 4 5 5 6 7 12 29 91)
>
```

Determine language from source CRLF 67:25 516.20 MB

Test


tests.rkt - DrRacket
File Edit View Language Racket Insert Tabs Help
tests.rkt (define ...)
Check Syntax Debug Macro Stepper Run Stop
#lang racket
(require (file "assignment_q2.rkt") imported from "racket"
(file "assignment_q3.rkt"))
;
;Don't worry about this file unless you are doing the extra credit tests.
;
;This structure allows a single function call
;to run every test in sequence, rather than
;calling each function separately.
(define (runTests)
(begin
(display "Running tests...\n")
;begin calling test functions
(printf "1: ~a\n" (ins_beg "a" '("b" "c" "d")))
(printf "2: ~a\n" (ins_end "a" '("b" "c" "d")))
(printf "3: ") (cout_top_level '("a" '("b" "c" "d") "c"))
(printf "\n4: ") (count_instances "e" '("e" "f" "g" "e" "g" "h" "e" "f" "g" "e"))
(printf "\n5: ") (count_instances_tr "c" '("a" "b" "c" "a" "c" "d" "a" "b" "c" "d" "a" "e" "c"))
(printf "\n6: ") (count_instances_deep 3 '(1 2 3 (1 3 4) 1 2 3 1 3 6 3 5 3))
;end calling test functions
(display "\nTests complete!\n")))
;Begin test functions
(define (test_ins_beg1)
(eq? (ins_beg 1 '(2 3 4)) '(1 2 3 4)))
;End test functions
;Run the tests
(runTests)
Welcome to DrRacket, version 6.11 [3m].
Language: racket, with debugging; memory limit: 128 MB.
Q3 Sorted Tree'(2 4 5 5 6 7 12 29 91)
Running tests...
Hello, I'm ins_beg!
1: (a b c d)
Hello, I'm ins_end!
2: (b c d a)
3: Hello, I'm cout_top_level!
Finished counting elements! = 3
4: Hello I'm count_instances!
Finished counting instances! = 4
5: Hello I'm count_instances_tr!
Finished counting instances! = 4
6: Hello I'm count_instances_tr!
Finished counting instances! = 6
Tests complete!
>
Determine language from source
CRLF 17:14
526.17 MB