Assignment 3 part 1:

1.

As shown in class, ‘let’ is syntactic abbreviation for lambda procedure. We’ve seen the “rewriteLet” function that converts let expression to procedure. Therfore there is no special evaluation rule for ‘let’ expression and hence it is not a special form in L3.

2.

no, during the evaluation of ‘let’ expression we are immediately evaluating the body according to the bindings whereas in the evaluation of proc expression there is no obligation to evaluate tis body immediately and that’s why we must create closure when evaluating procedure, but it is not necessity in ’let’ evaluation

3.

a) contract violation: (+ 4 'a)

b) division by zero: (/ 5 0)

c) unbound identifier: (+ x 0)

d) undefined operator in appExp: (#t 1)

4.

The purpose of valueToLitExp is creating expressions from values (Lifting). it solves the case where in the substitution model as we evaluate an app expression, we substitute values with the corresponding varRef in the body of the procExp before the function is evaluated. the body of procExp is an expression so it should be composed only by other expressions. valueToLitExp will change the values with the corresponding expressions so we will have a valid compound expression as the body of procExp.

5.

In normal evaluation strategy the arguments are evaluated only when substituting expression, therefore we don’t need to convert values to expressions, hence there are no type issues

6.

Special form has a special evaluation rule, and sometimes not all their parts are even evaluated. In scheme every primitive operator is a procedure and all its arguments have been evaluated before applying the procedure.

7.

The reason to switch from the substitution model to the environment model is performance: in the substitution model every procedure call causes renaming of its body and evaluation of the arguments. This process charge heavy resources (memory allocation, garbage collecting, unnecessary long runtime) every time the procedure is called instead of once.

Example:

(define make-multiplier

    (lambda (n)

        (lambda (x) (\* n x))))

(let ((double (make-multiplier 2)))

    (let ((n 1))

    (double 5)))

In the substitution model:

8.

A picture containing text, blackboard

Description automatically generated

q2 – תרגול מול קוד

תשובה של אלחדד:

בשאלה 2 רשימת אפשרויות:דוג' ב3 יש ב4 אין לפי החלפה לפי סביבה וכו'

שאלה 3 סוגים?

נלך על המימוש של האינטרפרטר בזמן eval מחזיר failure

שאלה 8 מה זה לקסיקל בלוק וקונטרול לינקס

לקסיקל בלוק: המלבנים

קונטרול לינקס: החץ