מגישים: דן ירדן 316611854 נדב שקד 312494925

עבודה 2

חלק 1

שאלה 1

- Primitive atomic expression: 2
- Non-primitive atomic expression: x (x was defined before)
- Non-primitive compound expression: (+ 1 2)
- Primitive atomic value: the numerical value of 2
- Non-primitive atomic value: the symbol ‘green in L3
- Non-primitive compound value: the list (7, 7, 7)

שאלה 2

זהו מבנה מיוחד, הבנוי באופן הבא:
סוגריים שבמקום השמאלי נמצא אופרטור מיוחד, בנוסף כל אופרנד מיוחד מחושב לפי משמעות ייחודית ספציפית לו.
לדוגמא (define x (+ 1 2))

שאלה 3

משתנה חופשי בביטוי נתון הוא משתנה אשר ישנה התייחסות אליו (varRef) בביטוי, אך אין אזכור שלו בהכרזת המשתנים של הביטוי (varDecl).
בדוגמא הבאה y הוא משתנה חופשי: (lambda (x) x) y))

שאלה 4

זה למעשה היררכיה של tokens לאחר שמחרוזת הקוד נשלחת כקלט ל scanner מוחזר מערך של tokens ולאחר הפעלת ה reader על הפלט שהתקבל בפעולה הקודמת מוחזר מערך היררכי של tokens.
לדוגמא:
עבור ה tokens הבאים: ['(', ‘5’,’4’,’)’, ‘3’, ‘x’, ‘>’, ‘(’, ‘if’, ‘(’]
מוחזר ה s-exp הבא: ['if', ['>', ‘x’, ‘3’],’4’,’5‘]

שאלה 5

קיצור תחבירי הוא הגדרה מקוצרת של ביטוי כלשהו, כלומר הגדרת ביטוי בצורה קצרה יותר כאשר היה ניתן להגדיר אותו עם ביטויים אחרים הקיימים כבר בשפה.
לדוגמא בביטוי הבא let הוא קיצור תחבירי להפעלת הפרוצדורה:

```
(define f
  (lambda (x y)
    (
      (lambda (v1 v2)
        (+ (* x
              (square v1)))
        (* y
           v2)
```

(* v1 v2)))))

דוגמא נוספת:

((cond (((> x 3) 4) ((> y 8) 5) (else 6)) ⇒ (if (> x 3) 4 (if (< y 8) 5 6)

שאלה 6

ניתן להפוך כל תכנית ב L3 לתכנית ב L30 מכיוון שכל ביטוי שאיננו list ברור שניתן ליצור ב L30, וכאשר הביטוי הינו list אז ניתן ליצור אותו בעזרת שרשור של זוגות כפי שנלמד בכיתה. לדוגמא:

list(3 4) → (cons 3 (cons 4 ‘()))

שאלה 7

יתרון של primOp: מהיר לחישוב מכיוון שהוא חלק מה syntax של השפה.

יתרון של Closure: אין צורך לבצע שינוי באינטרפרטר על מנת להגדיר אופרטורים פרימיטיביים חדשים.

שאלה 8

אימפלמנטציה שונה של map שבה מתחילים את ביצוע הפונקציה על המערך בסדר הפוך (כלומר תחילה על האיבר האחרון במערך, לאחר מכן זה שלפניו ולבסוף האיבר הראשון) לא תשפיע על תוצאת הפעולה משום שהביצוע של הפעולה על כל תא אינו תלוי בביצועה על שאר התאים ולכן התוצאה הסופית תהיה זהה, וגם מכיוון שזהו תכנות פונקציונלי.

אימפלמנטציה שונה של reduce כפי שתיארנו מעלה תשפיע על התוצאה המוחזרת לדוגמא עבור המערך וביצוע פעולת ה reduce הבאים:

[3, 2].reduce((acc: number, cur: number) => power(cur, acc), 2)

original order: (3^2) = 9 → 2^9 = 512

new order: (2^2) = 4 → 3^4 = 81

- הערה: power(x, y) מחזיר את הערך x^y.

כנ״ל לדוגמא בכפל מטריצות, כלומר כל מה שאיננו אסוציאטיבי.

אימפלמנטציה שונה של filter כפי שתואר קודם לכן לא תשפיע על התוצאה המוחזרת משום שהביצוע של הפעולה על כל תא אינו תלוי בביצועה על שאר התאים ולכן התוצאה הסופית תהיה זהה, וגם מכיוון שזהו תכנות פונקציונלי.

אימפלמנטציה שונה של compose כפי שתיארנו מעלה תשפיע על התוצאה המוחזרת לדוגמא עבור מערך הפונקציות הבא:

compose([f1, f2])

כאשר f1 זוהי פונקציה מ T1 אל T2 ,ו f2 זוהי פונקציה מ T2 אל T3. (כך ש T1, T2, T3 אינם שווים זה לזה)
כאשר נבצע את ה compose מההתחלה לסוף נקבל את פונקציית ההרכבה הבאה: f = f1(f2)
ואם נבצע את ה compose בסדר הפוך נקבל פונקציה שאינה מוגדרת.

חלק 2

; Signature: last-element(lst)
; Type: [List(T) → T]
; Purpose: returns the last element of the given list.
; Pre-conditions: lst != ‘().
; **Tests**: (last-element (1 2 3 4)) → 4

; Signature: power(n1 n2)
; Type: [Number*Number → Number]
; Purpose: returns n1 to the power of n2.
; Pre-conditions: (n1 >= 0 & n2 >= 0)
; **Tests**: (power 2 4) → 16

; Signature: sum-lst-power(lst n)
; Type: [List(Number) * Number → Number]
; Purpose: given a list and a number n, returns the sum of all the numbers, each one raised to the power of n
; Pre-conditions: n and all the elements in the list lst are natural
; **Tests**: (sum-lst-power (list 1 4 2) 3) → 73

; Signature: num-from-digits(lst)
; Type: [List(Number) → Number]
; Purpose: returns a decimal value representation of the list lst.
; Pre-conditions: all the elements int the list lst are natural
; **Tests**: (power 2 4) → 16

; Signature: is-narcissistic(lst)
; Type: [List(
Number) → Boolean]
; Purpose: returns if a given number is narcissistic, meaning the sum of its own digits, each raised to the power of the number of the digits equals its value.
; Pre-conditions: all the elements in the list lst are natural
; **Tests**: (is-narcissistic (list 1 5 3)) → #t