דוח משוב לסטודנט

מזהה סטודנט	N102673440
תאריך בחינה	יום שני, 27 ביולי 2026
מזהה קורס	20905
שם קורס	שפות תכנות
מרצה	דני כלפון

ניקוד שאלות פתוחות	ציון מבחן מקורי	ציון מבחן סופי
97.00	97.00	97.00

סיכום

מספר שאלה	תיאור	ניקוד	ניקוד מירבי
1.1	עדכון תחביר השפה	8.00	10.00
1.2	בנאי נוסף לטיפוס proc	6.00	6.00
1.3	מימוש התנהגות compose-exp	11.00	12.00
1.4	עדכון apply-procedure	12.00	12.00
2.1	עדכון תחביר השפה	10.00	10.00
2.2	מימוש התנהגות env-proc-exp	20.00	20.00
3.1	פירוק למרכיבים והקצאת משתנים	7.00	7.00
3.2	חיבור משוואות	8.00	8.00
3.3	פתרון משוואות לפי צעדי האלגוריתם והגעה לתשובה סופית נכונה	15.00	15.00

מלא את הפרטים בכל המקומות הדרושים

מספר תעודת זהות (9 ספרות)	מספר הקורס	מועד
203379706	20905	91

הדבק כאן את מדבקת הנבחן

N102673440

ת.ז: 203379706 מספר סידורי: 4

לשימוש הבודק

247970

לשימוש הבודק

1
203379706

הדקדוק לא מוודא בזמן הפירסור שיש לפחות 2 ביטויים

שאלה 1:

-2 (1.1)

lang.scm

```
(define the-grammer ...
(expression ("compose" "[" (separated-list expression "<<") "]")
compose-exp))
```

data-structures.scm

(1.2)

```
(define-datatype proc proc?
 (procedure (var symbol?) (body expression?) (env environment?))
 (composition (p-lst (list-of proc?))))
```

interp.scm:

```
(define value-of ...
 ;; Few notes about my implementation of compose-exp case:
 ;; 1)
 ;; following the requirements in the exam I implemented error
 ;; handling for the case compose-exp get list of proc expressions of
 ;; length smaller than 2, at "(if (>= ..".
 ;; 2)
 ;; if any element in p-exps is not a proc-val
 ;; the existing 'expval->proc' extractor will trigger a
 ;; eopl:error via calling eopl-extractor-error
 ;; therefore I did not implemented here my own
 ;; custom eopl:error message since expval->proc do it for me already
 ;; 3)
 ;; we save the list of procs in reversed order since the last
 ;; proc expression is the first proc to invoke [reverse order]
 ;; it will be easer to recursively invoke procedures in a composition
 ;; call chain [easier then to use car/cdr recursively.
 ;; I was not sure if "reverse" function exist in Scheme/eopl or not
 ;; so I also wrote an implementation of the reverse function in case
 ;; we don't have any built-in reverse function
```

את זה היה צריך לוודא בשלב הפירסור ולא כאן (הורד על כך ניקוד פעם אחת)

צריך להוציא שגיאה בנוסח המתואר בשאלה במקרה ולא מדובר על פרוצדורה

```
 (compose-exp (p-exps)
 (if (>= (length p-exps) 2)
    (let ((pvals (map (λ (e) (expval->proc (value-of e env))) p-exps)))
    (proc-val (composition (reverse pvals))))
    (eopl:error 'value-of "composition must be composed from at
    least 2 procs, ~s have less than 2" p-exps)))
```

11 (1.3)

```
;; USE ONLY IF 'reverse' is not builtin scheme/eopl function
;; reverse: List -> List
(define reverse
 (λ (lst) (letrec ((reverse-rec (λ (src dst)
  (if (null? src) dst (reverse-rec (cdr src) (cons (car src) dst))))))
 (reverse-rec lst '()))))
```

המשך שאלה 1:

interp.scm:

```
(define apply-procedure
 (λ (proc1 val)
  (cases proc proc1
   (procedure (var body saved-env)
    (value-of body (extend-env var val saved-env)))
   (composition (lst) (apply-procedure-rec lst val))
)))
```

```
;; apply-procedure-rec : ListOf[Proc] * ExpVal -> ExpVal
(define apply-procedure-rec
 (λ (p-lst val)
  (if (null? p-lst) val
   (apply-procedure-rec (cdr p-lst) (apply-procedure (car p-lst) val))
)))
```

3
203379706

שאלה 2:

lang.scm:

```
(define the-grammer ...
 (expression ( "<" identifier ">" "." "EnvAdd" "=" "("
 (separated-list identifier ":" expression ",") ")" )
 env-proc-exp)
)
```

10
(2.1)

interp.scm:

```
(define value-of ...
 (env-proc-exp (id1 ids exps)
  (let* ((p-ref (apply-env env id1))
         (p-val (deref p-ref))
         (e-vals (map (λ(e)(value-of e env)) exps)))
         (cases expval p-val
          (proc-val (p)
           (cases proc p
            (procedure (var body saved-env)
             (let ((new-env (extend-env* ids e-vals saved-env)))
              (begin
               (setref! p-ref (proc-val (procedure var body new-env)))
               (num-val 27))))))
(else (eopl:error 'env-proc-exp "~s is not a procedure" id1))))))

;; extend-env* : ListOf[Symbol] * ListOf[ExpVal] * Env -> Env
(define extend-env*
 (λ (vars vals env)
  (if (null? vars) env
   (extend-env* (cdr vars) (cdr vals)
   (extend-env (car vars) (car vals) env)))))
```

20
(2.2)

שאלה 3:

```
proc (x:?)
  (proc (y:?) x 10)
```

שאלה 3 סעיף א:

$EXPRESSION$	VAR
$proc\ (x{:}?)\ (proc\ (y{:}?)\ x\ 10)$	T_0
x	T_x
$(proc\ (y\ {:}?)\ x\ 10)$	T_1
$proc\ (y{:}?)\ x$	T_2
y	T_y
x	T_x
10	T_3

7
(3.1)

לסיכום אלו המשתנים שהגדרתי:

$T_0 = proc\ (x{:}?)\ (proc\ (y{:}?)\ x\ 10)$
$T_1 = \ (proc\ (y{:}?)\ x\ 10)$
$T_2 = \ proc\ (y{:}?)\ x$
$T_3 = 10$
$T_x = x$
$T_y = y$

שאלה 3 סעיף ב:

$EXPRESSION$	VAR	EQ
$proc\ (x{:}?)\ (proc\ (y{:}?)\ x\ 10)$	T_0	$[T_0 = T_x \to T_1]$
x	T_x	
$(proc\ (y\ {:}?)\ x\ 10)$	T_1	$[T_1 = T_x]$
$proc\ (y{:}?)\ x$	T_2	$\begin{bmatrix} T_2 = T_y \to T_x \\ T_2 = T_3 \to T_x \end{bmatrix}$
y	T_y	
x	T_x	
10	T_3	$[T_3 = int]$

לסיכום אלו המשוואות שהתקבלו :

EQ
$T_3 = int$
$T_1 = T_x$
$T_0 = T_x \to T_1$
$T_2 = T_y \to T_x$
$T_2 = T_3 \to T_x$

8
(3.2)

שאלה 3 סעיף ג:

$$\left|\begin{array}{cc} EQS & SB \\ T_3 = int & \\ T_1 = T_x & \\ T_2 = T_y \to T_x & \\ T_2 = T_3 \to T_x & \\ T_0 = T_x \to T_1 & \end{array}\right| \to \left|\begin{array}{cc} EQS & SB \\ T_1 = T_x & T_3 \to int \\ T_2 = T_y \to T_x & \\ T_2 = int \to T_x & \\ T_0 = T_x \to T_1 & \end{array}\right|$$

$$\left|\begin{array}{cc} EQS & SB \\ & T_3 \to int \\ T_2 = T_y \to T_x & \\ T_2 = int \to T_x & T_1 \to T_x \\ T_0 = T_x \to T_x & \end{array}\right| \to \left|\begin{array}{cc} EQS & SB \\ & T_3 \to int \\ T_y \to T_x = int \to T_x & T_1 \to T_x \\ T_0 = T_x \to T_x & T_2 \to (T_y \to T_x) \end{array}\right|$$

$$\left|\begin{array}{cc} EQS & SB \\ & T_3 \to int \\ & T_1 \to T_x \\ T_0 = T_x \to T_x & T_2 \to (T_y \to T_x) \\ & T_y \to int \end{array}\right| \to T_0 = T_x \to T_x$$

לסיכום, הטיפוס שהוקש עבור התוכנית $T_0 = T_x \to T_x$ כלומר התוכנית מקבלת משתנה מטיפוס לא ידוע T_x ומחזירה ערך מאותו טיפוס לא ידוע T_x.

15
(3.3)

٦١٥

לשימוש הבודק

247970

לשימוש הבודק

לשימוש הבודק

247970

לשימוש הבודק

לשימוש הבודק

247970

לשימוש הבודק

לשימוש הבודק

לשימוש הבודק

247970

לשימוש הבודק

לשימוש הבודק

247970

לשימוש הבודק

לשימוש הבודק

247970

לשימוש הבודק

לשימוש הבודק

247970

לשימוש הבודק

גליון תשובות לשאלות רב-ברירתיות

הקף במעגל את התשובה שבחרת (לכל שאלה יש רק תשובה אחת נכונה).
אם תרצה לבטל תשובה שבחרת, סמן עליה X.
דוגמה לתשובה שבחרת: א ב ג (ד) ה ו ז ח ט
דוגמה לתשובה שבטלת: א ב ג ד ה (X) ז ח ט

שאלה	תשובה									
1	א	ב	ג	ד	ה	ו	ז	ח	ט	
2	א	ב	ג	ד	ה	ו	ז	ח	ט	
3	א	ב	ג	ד	ה	ו	ז	ח	ט	
4	א	ב	ג	ד	ה	ו	ז	ח	ט	
5	א	ב	ג	ד	ה	ו	ז	ח	ט	
6	א	ב	ג	ד	ה	ו	ז	ח	ט	
7	א	ב	ג	ד	ה	ו	ז	ח	ט	
8	א	ב	ג	ד	ה	ו	ז	ח	ט	
9	א	ב	ג	ד	ה	ו	ז	ח	ט	
10	א	ב	ג	ד	ה	ו	ז	ח	ט	
11	א	ב	ג	ד	ה	ו	ז	ח	ט	
12	א	ב	ג	ד	ה	ו	ז	ח	ט	
13	א	ב	ג	ד	ה	ו	ז	ח	ט	
14	א	ב	ג	ד	ה	ו	ז	ח	ט	
15	א	ב	ג	ד	ה	ו	ז	ח	ט	
16	א	ב	ג	ד	ה	ו	ז	ח	ט	
17	א	ב	ג	ד	ה	ו	ז	ח	ט	
18	א	ב	ג	ד	ה	ו	ז	ח	ט	
19	א	ב	ג	ד	ה	ו	ז	ח	ט	
20	א	ב	ג	ד	ה	ו	ז	ח	ט	

שאלה	תשובה									
21	א	ב	ג	ד	ה	ו	ז	ח	ט	
22	א	ב	ג	ד	ה	ו	ז	ח	ט	
23	א	ב	ג	ד	ה	ו	ז	ח	ט	
24	א	ב	ג	ד	ה	ו	ז	ח	ט	
25	א	ב	ג	ד	ה	ו	ז	ח	ט	
26	א	ב	ג	ד	ה	ו	ז	ח	ט	
27	א	ב	ג	ד	ה	ו	ז	ח	ט	
28	א	ב	ג	ד	ה	ו	ז	ח	ט	
29	א	ב	ג	ד	ה	ו	ז	ח	ט	
30	א	ב	ג	ד	ה	ו	ז	ח	ט	
31	א	ב	ג	ד	ה	ו	ז	ח	ט	
32	א	ב	ג	ד	ה	ו	ז	ח	ט	
33	א	ב	ג	ד	ה	ו	ז	ח	ט	
34	א	ב	ג	ד	ה	ו	ז	ח	ט	
35	א	ב	ג	ד	ה	ו	ז	ח	ט	
36	א	ב	ג	ד	ה	ו	ז	ח	ט	
37	א	ב	ג	ד	ה	ו	ז	ח	ט	
38	א	ב	ג	ד	ה	ו	ז	ח	ט	
39	א	ב	ג	ד	ה	ו	ז	ח	ט	
40	א	ב	ג	ד	ה	ו	ז	ח	ט	

לשימוש פנימי

מספר התשובות הנכונות: _____ ציון: _____

247970 שם הבודק: _____