דוח משוב לסטודנט

מזהה סטודנט	[redacted]
תאריך בחינה	יום שני, 22 ביוני 2026
מזהה קורס	20905
שם קורס	שפות תכנות
מרצה	דני כלפון

ניקוד שאלות פתוחות	ציון מבחן מקורי	ציון מבחן סופי
98.00	98.00	98.00

סיכום

מספר שאלה	תיאור	הערות	ניקוד	ניקוד מירבי
1.1	זיכרון מטמון – init-cache		5.00	5.00
1.2	זיכרון מטמון – search-cache		5.00	5.00
1.3	זיכרון מטמון – rearrange-cache		5.00	5.00
1.4	זיכרון מטמון – remove-at		5.00	5.00
1.5	זיכרון מטמון – add-cache		5.00	5.00
1.6	עדכון התנהגות הביטוי call-exp		14.00	15.00
2.1	עדכון תחביר השפה	עדכון דקדוק השפה	10.00	10.00
2.2	מימוש התנהגות update-proc-exp	מימוש התנהגות	15.00	15.00
2.3	טיפול נכון בשגיאות עם הודעה מתאימה	טיפול בשגיאות	5.00	5.00
3.1	פירוק נכון של הביטוי והקצאת משתני טיפוס	פירוק והגדרת משתנים	7.00	7.00
3.2	חיבור משוואות מתאימות לכל הביטויים	חיבור משוואות	7.00	8.00
3.3	פתרון המשוואות על פי צעדי האלגוריתם, והגעה לתשובה סופית נכונה	פתרון המשוואות	15.00	15.00

מלא את הפרטים בכל המקומות הדרושים

מספר תעודת זהות (9 ספרות)	מספר הקורס	מועד
2[redacted]	2 0 9 0 5	8 1

הדבק כאן את מדבקת הנבחן

N[redacted]

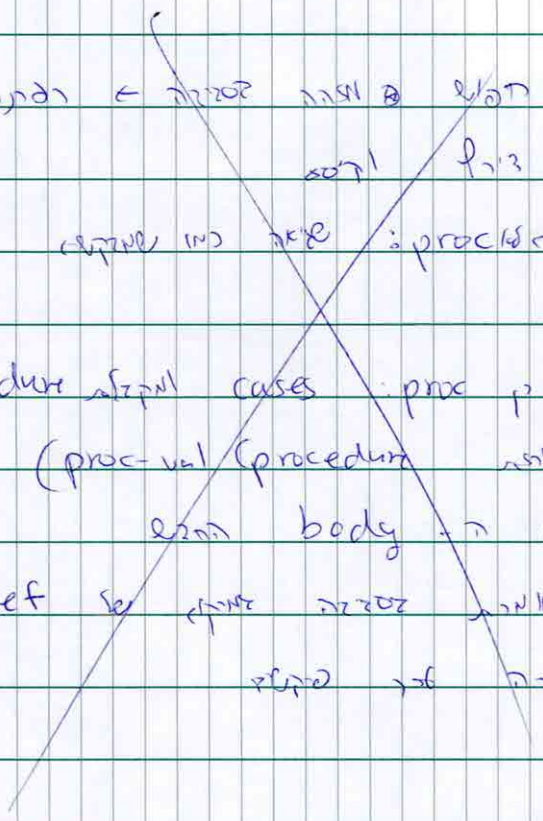
מספר סידורי: 4 [redacted]

לשימוש הבודק

210870

טיוטה

implicit



לשימוש הבודק

שאלה 2

10
(2.1)
עדכון דקדוק השפה

(1
(expression ("*" "(" identifier ")" "=" expression) update-proc-exp)

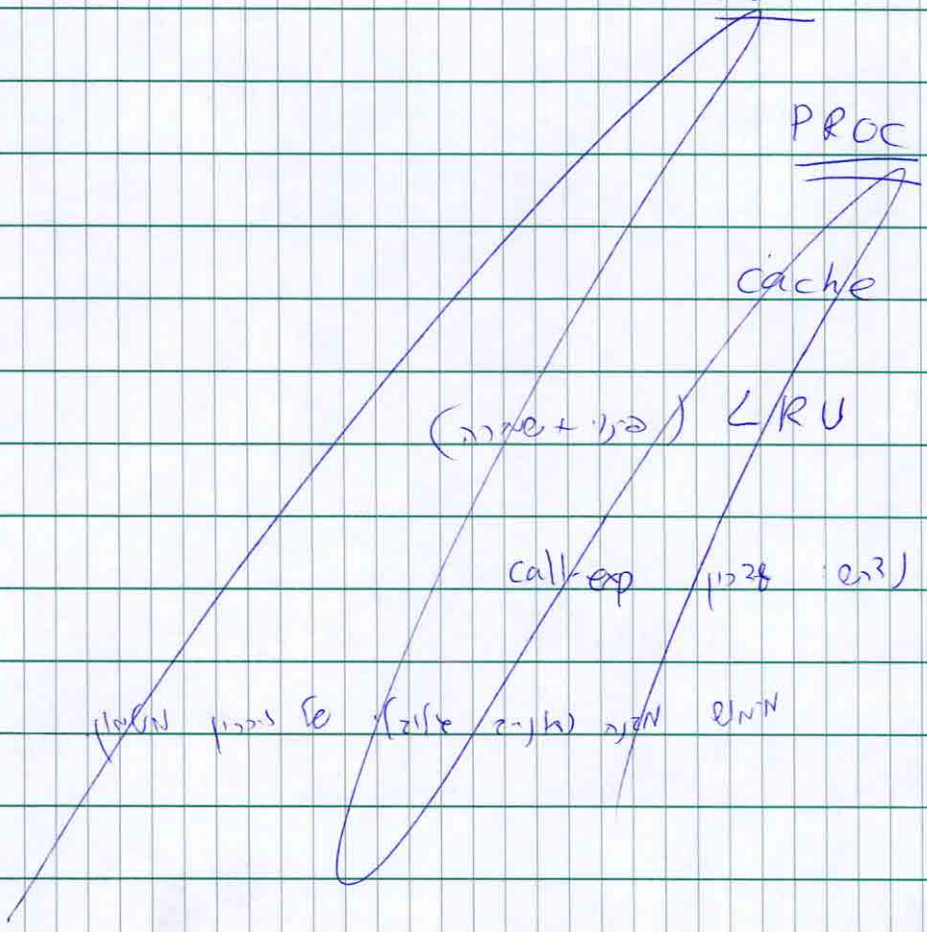
(2
:value-of עדכון

(update-proc-exp (id exp)
(let* ((val-ref (apply-env env id))
(val (deref val-ref)))
(cases expval val
(proc-val (proc1)
(cases proc proc1
(procedure (var body saved-env)
(let ((new-proc (proc-val
(procedure var exp saved-env)
))))
(begin
(setref! val-ref new-proc)
(num-val 27)))))))
(else (eopl:error 'update-proc "~s is
not a procedure" id)))))

15
(2.2)
מימוש התנהגות

5
(2.3)
טיפול בשגיאות

210870


C'C
PROC
cache
LRU
call-exp

לשימוש הבודק

שאלה 3

expression	type variables
1) (proc(x:?)x zero?(9))	T_0
2) (proc(x:?)x zero?(9))	T_1
3) proc(x:?)x	T_2
x	T_x
x	T_x
4) zero?(9)	T_3
9	int
5) proc(a:?)a	T_4
a	T_a
a	T_a
6) proc(b:?)b	T_5
b	T_b
b	T_b

7 (3.1)

פירוק והגדרת משתנים

זה אותו אחד

(ב) רשום משוואות שאלוגיות לפרוש המתאימות מסעיף שאלה קודמת:

exp	equ
1)	$T_0 = T_4$
	$T_4 = T_5$
	$T_1 = bool$
2)	$T_2 = T_3 \to T_1$
3)	$T_2 = T_x \to T_x$
4)	$T_3 = bool$ (השמה $q = int$)
5)	$T_4 = T_a \to T_a$
6)	$T_5 = T_b \to T_b$

7 (3.2)

חיבור משוואות

לא מדויק. אלה לא משוואות שמתאימות לביטוי if

5

210870

⑧ אחלה עדיין את על המשוואות הפשוטות

equ	sub
$T_0 = T_4$	$T_1 = bool$
$T_4 = T_5$	$T_3 = bool$
$T_2 = T_3 \to T_1$	
$T_2 = T_x \to T_x$	
$T_4 = T_a \to T_a$	
$T_5 = T_b \to T_b$	

$T_3 = bool$, $T_1 = bool$ sub אחלה עדיין $T_2 = T_3 \to T_1$ ~~אחלה~~ וכפל המשוואה

ואנו עדיין לפי sub

equ	sub
$T_0 = T_4$	$T_1 = bool$
$T_4 = T_5$	$T_3 = bool$
$T_2 = T_x \to T_x$	$T_2 = bool \to bool$
$T_4 = T_a \to T_a$	
$T_5 = T_b \to T_b$	

עוד וכפל המשוואה $T_2 = T_x \to T_x$ נשארנו עם המשוואה $T_2 = bool \to bool$

ועכשיו $T_x = bool$, $T_x = bool$ נשארנו עם המשוואה אחת אחרון ועכשיו

equ	sub
$T_0 = T_4$	$T_1 = bool$
$T_4 = T_5$	$T_3 = bool$
$T_4 = T_a \to T_a$	$T_2 = bool \to bool$
$T_5 = T_b \to T_b$	$T_x = bool$

equ	Sub
$T_0 = T_4$	$T_1 = bool$
$T_4 = T_5$	$T_3 = bool$
$T_5 = T_6 \to T_6$	$T_2 = bool \to bool$
	$T_x = bool$
	$T_4 = T_a \to T_a$

ג) $T_4 = T_a \to T_a$ נשאיר את $T_4 = T_5$ ונציב ב-Sub

equ	Sub
$T_0 = T_4$	$T_1 = bool$
$T_5 = T_6 \to T_6$	$T_3 = bool$
	$T_2 = bool \to bool$
	$T_x = bool$
	$T_4 = T_a \to T_a$
	$T_5 = T_a \to T_a$

ג) $T_5 = T_a \to T_a$ נשאיר את $T_5 = T_6 \to T_6$ ונציב $T_a = T_6$, $T_a = T_6$ ונקבל

equ	Sub
$T_0 = T_4$	$T_1 = bool$
	$T_3 = bool$
	$T_2 = bool \to bool$
	$T_x = bool$
	$T_4 = T_a \to T_a$
	$T_5 = T_a \to T_a$
	$T_a = T_b$

ג) $T_4 = T_a \to T_a$ נשאיר את $T_0 = T_4$ ונקבל

var	sub
	$T_1 = bool$
	$T_3 = bool$
	$T_2 = bool \to bool$
	$T_x = bool$
	$T_4 = T_a \to T_a$
	$T_5 = T_a \to T_a$
	$T_6 = T_b$
	$T_a = T_a \to T_a$

פעמיים את כל המשוואות חלו לפחות. השאיר T_a הוקלא
לעל הראשון ולין הסתם של הראשון הזה פלוטרוס ל- T_a
ולאז $T_a \to T_a$ רלוונטי בהוכחה שיוקלא שאיר מטפוס T_a
ויחס יהה שווה מטפוס T_a

15
(3.3)

פתרון המשוואות

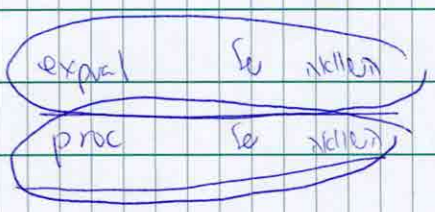
לשימוש הבודק

שאלה 1

proc

(define-datatype cache cache?
  (cache-val (p proc?)
             (arg expval?)
             (res expval?)))

search-cache:



(search-cache (lambda (proc arg)
  (letrec ((run-search (lambda (idx
    (cases cache (list-ref the-cache idx))

לשימוש הבודק

שאלה 1

מימוש בעזרת המקומן:

```
(define init-cache
  (lambda ()
    (set! the-cache '())))
```

5
(1.1)

```
(define search-cache
  (lambda (prc arg)
    (letrec ((run-search (lambda (idx)
                           (if (= idx (length the-cache))
                               #f
                               (cases cache (list-ref the-cache
                                                      idx)
                                 (cache-val (p res)
                                   (if (and (expval-equal? arg a)
                                            (proc-equal? prc p))
                                       #t
                                       (run-search (+ 1 idx)))))))))
      (run-search 0))))

(define-datatype cache cache?
  (cache-val (p proc?)
             (a expval?)
             (res expval?)))
```

(define (expensive

(define add-cache

(lambda (proc arg result)

(begin

(if (= cache-size (length the-cache))

(remove-at the-cache (- cache-size 1)))

ב"ה

add-cache: אם הקאש לא מלא

cache -ל set מוסיף

(cons (list proc arg result) (את כל האיברים ברשימה))

אחרת

(cons (list proc arg result) the-cache) את כל האיברים ברשימה

hearrange: משנה את הסדר בין האיברים

remove-at כדי לשנות את ה set!

add-cache -ל קוראת

5
(1.5)

```
(define add-cache
  (lambda (prc arg result)
    (if (= cache-size (length the-cache))
        (begin
          (set! the-cache (remove-at the-cache (-1
                                                cache-size)))
          (set! the-cache (cons (list prc arg result) the-cache)))
        (set! the-cache (cons (list prc arg result) the-cache)))))
```

5
(1.3)

```
(define rearrange-cache
  (lambda (idx)
    (if (or (< idx 0) (>= idx (length the-cache)))
        (eopterror 'rearrange-cache "invalid index ~s" idx)
        (let ((cache-item (list-ref the-cache idx)))
          (begin
            (set! the-cache (remove-at the-cache idx))
            (add-cache (car cache-item) (cadr cache-item)
                       (caddr cache-item)))))))
```

```
(define (remove-at lst n)
  (letrec ((remove-inner
            (lambda (l index)
              (if (zero? index)
                  (cdr l)
                  (cons (car l)
                        (remove-inner (cdr l) (- index 1)))))))
    (if (or (< n 0) (>= n (length lst)))
        (eopl:error 'remove-at "invalid index ~s" n)
        (remove-inner lst n))))
```

5 (1.4)

✓

```
(call-exp (rator rand)
  (let ((proc (expval->proc (value-of rator env)))
        (arg1 (value-of rand env))
        (idx (search-cache proc arg1)))
    (if idx
        (begin
          (rearrange-cache idx)
          (eopl:printf "procedure call result ~s found in cache :-)" (caddr (list-ref the-cache 0)))
          (caddr (list-ref the-cache idx)))
        (let ((res (apply-procedure proc arg1)))
          (begin
            (add-cache proc arg1 res)
            res))))))
```

14 (1.6)

caddr

✓

5
(1.2)

✓

```
(define search-cache
  (lambda (prc arg)
    (letrec ((run-search
               (lambda (lst idx)
                 (if (null? lst)
                     #f
                     ~~(((cadr (equal?~~
                     (let ((prc1 (car (car lst)))
                           (arg1 (cadr (car lst))))
                       (if (and (equal-proc? prc prc1)
                                (equal-expval? arg arg1))
                           idx
                           (run-search (cdr lst) (+1 idx))))))))
      (run-search the-cache 0))))
```

לא תשובה לשאלה ששאלת. במהלך המבחן, ניתן להשתמש ב-
equal? להשוואת ערכים ופרוצדורות. ואולם לא השוואות תקינות
הגדרתי ב- letrec בנוסף את ⊛ (equal-expval? equal-proc?)

```
⊛ (equal-expval? (lambda (ex1 ex2)
    (cases expval ex1
     (num-val (v1)
        (cases expval ex2
           (num-val (v2)
              (= v1 v2))
        (else #f)))
   (bool-val (v1)
```

```
(cases expval ex2
  (bool-val (v2)
    (and equal? v1 v2))
  (else #f)))
(proc-val (proc1)
  (cases proc proc1
    (procedure (var1 body1 env1)
      (cases expval ex2
        (proc-val (proc2)
          (cases proc proc2
            (procedure (var2 body2
                        env2)
              (and (equal? var1 var2)
                   (equal? body1 body2)
                   (equal? env1 env2))))))
        (else #f)))))))
```

כל זה פשוט ארוך ומיותר
שימוש ב- equal? ישירות
בין 2 הישויות שרוצים להשוות
היה עושה את העבודה

```
(equal-proc?
  (lambda proc1 proc2)
  (cases proc proc1
    (procedure (var1 body1 env1)
      (cases proc proc2
        (procedure (var2 body2 env2)
          (and (equal? var1 var2)
               (equal? body1 body2)
               (equal? env1 env2)))))))
```

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: _____

שם הבודק: _____ 210870