י"ג באב תשפ"ו

27

ביולי 2026

מס' שאלון - 528

סמסטר 2026ב

מס' מועד 91

20905 / 4

שאלון בחינת גמר

20905 - שפות תכנות

משך בחינה: 3 שעות

בשאלון זה **10** עמודים

מבנה הבחינה:

בבחינה 3 שאלות.
משך זמן הבחינה 3 שעות.
עליכם לענות על כל השאלות.
משקל השאלות מפורט בגוף השאלון.

שימו לב:
את תשובותיכם רשמו אך ורק במחברת הבחינה.
כתבו בכתב יד ברור ומסודר.

חומר עזר המותר בשימוש:
ספר הקורס "Essentials of Programming Languages".
מדריך הלמידה "שפות תכנות".

חומר עזר:
ספר הקורס: essentials of programming languages
מדריך הלמידה.
מותר לכתוב הערות בכתב יד בלבד, רק על גבי מדריך הלמידה.
אסור שימוש בעזרים ובחומרים מקוונים.

אינכם חייבים
להחזיר את השאלון לאוניברסיטה הפתוחה

בהצלחה !!!

י"ג באב תשפ"ו

מס' שאלון - 528

27

ביולי 2026

סמסטר 2026ב

מס' מועד 91

20905 / 4

שאלון בחינת גמר

20905 - שפות תכנות

משך בחינה: 3 שעות

בשאלון זה 10 עמודים

מבנה הבחינה:

בבחינה 3 שאלות.
משך זמן הבחינה 3 שעות.
עליכם לענות על כל השאלות.
משקל השאלות מפורט בגוף השאלון.

שימו לב:
את תשובותיכם רשמו אך ורק במחברת הבחינה.
כתבו בכתב יד ברור ומסודר.

חומר עזר המותר בשימוש:
ספר הקורס "Essentials of Programming Languages".
מדריך הלמידה "שפות תכנות".

חומר עזר:
ספר הקורס: essentials of programming languages
מדריך הלמידה.
מותר לכתוב הערות בכתב יד בלבד, רק על גבי מדריך הלמידה.
אסור שימוש בעזרים ובחומרים מקוונים.

אינכם חייבים
להחזיר את השאלון לאוניברסיטה הפתוחה

בהצלחה !!!

בבחינה 3 שאלות. עליכם לענות על כל השאלות.

שאלה 1 (40 נקודות)

שאלה זו עוסקת בשדרוג של שפת "וישגור" (שפת PROC) המתוארת בספר הלימוד בפרק 3 בעמודים 74-81.

נרצה להוסיף לשפת "וישגור" צורה נוספת להגדרת פרוצדורה ע"י הרכבה (composition) מפרוצדורות אחרות בדומה לקיים בשפת התכנות Apache Groovy.

הפעלה של פרוצדורה שמוגדרת באמצעות הרכבת פרוצדורות על ארגומנט מסוים, מחזירה תוצאה שנוצרה מהפעלה של שרשרת הפרוצדורות מהן הורכבה הפרוצדורה. הפרוצדורה הראשונה בשרשרת מופעלת על הארגומנט, וכל ערך שהוחזר מפרוצדורה בשרשרת משמש כארגומנט להפעלה של הפרוצדורה הבאה בשרשרת, ולבסוף מוחזרת תוצאה שהיא הערך שמוחזר מהפרוצדורה האחרונה שהופעלה בשרשרת הפרוצדורות.

אופן הגדרת פרוצדורה כפי שקיים בשפה ע"י ביטוי proc-exp ימשיך להתקיים, ולשפה יצטרף ביטוי חדש המגדיר אופן נוסף להגדרת פרוצדורה באמצעות תהליך הרכבה מפרוצדורות אחרות.

להלן הדקדוק המגדיר את הביטוי החדש להגדרת פרוצדורה ע"י הרכבה:

$$\textit{Expression} ::= \text{compose}\ [\ \{\ \textit{Expression}\ \}^{+(<<)}\ ]$$

```
compose-exp (exps )
```

הסבר הדקדוק ודרישות מימוש:

ביטוי compose-exp מוגדר מאוסף של **לפחות 2 ביטויים** או יותר **(נדרש לוודא זאת בשלב הפרסור של התוכנית)**

הביטויים מופרדים ביניהם ע"י "<<". ערכם של הביטויים אמור להיות מסוג פרוצדורות. הפרוצדורות יכולות להיות מוגדרות באופן הקיים בשפה או בעצמם בצורת הרכבה. הפרוצדורה הימנית ביותר בשרשרת ההרכבה היא הפרוצדורה הראשונה בשרשרת, והפרוצדורה השמאלית ביותר היא הפרוצדורה האחרונה בשרשרת ההרכבה.

להלן מספר דוגמאות להמחשת העבודה עם פרוצדורות שהוגדרו ע"י הרכבה

המשך השאלה בעמוד הבא:

דוגמה 1:

```
> (run "let p1 = proc (x) -( x, 10)
        in
          let p2 = proc (x) *( x, 3)
          in
            let p3 = proc (x) +( x, 2)
            in
              let p4 = compose [p1<<p2<<compose [p3<<p1]<< proc (x) *( x, 5)<< p3  ]
              in
                (p4 3)")
(num-val 41)
>
```

הסבר דוגמה 1:

דוגמה זו מניחה שלהגדרות השפה התווספו גם פעולות חיבור וכפל בנוסף לפעולת חיסור הקיימת בשפה (איננכם נדרשים לממש זאת, זה נעשה רק לצורך הדוגמה).

בתוכנית המודגמת הוגדרו 3 פרוצדורות p1,p2,p3 באופן הרגיל הקיים בשפה (ע"י ביטוי מסוג proc-exp).

פרוצדורה p4 הוגדרה ע"י האופן החדש להגדרת פרוצדורה (ע"י הרכבה מפרוצדורות אחרות), ניתן לראות שפרוצדורה p4 מורכבת משרשרת הפרוצדורות הבאה: פרוצדורה p3 היא הראשונה בשרשרת, לאחריה פרוצדורה אנונימית proc (x) *(x,5), לאחריה פרוצדורה מורכבת בעצמה מהשרשרת p1 ו- p3. ולאחר מכן, הפרוצדורה p2 ולבסוף p1.

פרוצדורה p4 מופעלת על הארגומנט 3 ומחזירה תוצאה של 41, שהתקבלה מכך ש-3 נשלח כארגומנט תחילה לפרוצדורה p3 שהחזירה עבורו 5, התוצאה 5 הועברה כארגומנט לפרוצדורה האנונימית שהחזירה 25, התוצאה 25 הועברה כארגומנט אל p1 (הוא הבא בתור בשרשרת הפרוצדורות) 15, התוצאה 15 הועבר כארגומנט אל p3 שהחזירה 17, התוצאה 17 מועברת כארגומנט אל p2 שמחזירה 51, ולבסוף 51 מועבר כארגומנט אל הפרוצדורה p1 האחרונה בשרשרת המחזירה 41.

דוגמה נוספת בעמוד הבא

דוגמה 2:

```
> (run "let p1 = proc (x) -( x, 10)
        in
          let p2 = proc (x) *( x, 3)
          in
            let p3 = compose [p1<<p2<<8]
             in
               (p3 7)")
interp.scm:93:57: compose-exp:
 cannot compose with  non-procedure #(struct:const-exp 8)
>
```

הסבר דוגמה 2:

בדוגמה זו ניתן לראות שימוש שגוי בהגדרת פרוצדורה ע"י הרכבה. הפרוצדורה p3 מוגדרת ע"י הרכבת פרוצדורות, כאשר אחד הביטויים בשרשרת ההרכבה אינו פרוצדורה (הביטוי 8). ועל כן, תוצאת הרצת התוכנית הניבה שגיאה מתאימה.

מה עליכם לממש בשאלה זו?

בשאלה זו נדרש לממש את השינויים הבאים:

1. עדכון תחביר השפה עם הדקדוק של הביטוי החדש compose-exp כולל וידוא האילוצים שהוסברו קודם במסגרת הצגת הדקדוק החדש
2. בנאי נוסף לטיפוס הנתונים המופשט proc לבניית ייצוג לפרוצדורה ע"י הרכבה (נקרא לו composition)
3. עדכון ההתנהגות של הביטוי החדש compose-exp לבניית פרוצדורה
4. עדכון הפרוצדורה apply-procedure כך שתדע להפעיל גם פרוצדורה שהוגדרה ע"י הרכבה.

לנוחותכם, להלן קטעי הקוד מתוך מפרש השפה בהם נדרשים מימושים, עליכם להשלים במחברת הבחינה את המימושים הדרושים.

המשך השאלה בעמוד הבא:

1. **עדכון תחביר השפה:**

להלן קטע מתוך הגדרות השפה המופיעים בקובץ lang.scm, השלימו במחברת הבחינה את עדכון תחביר השפה עם הדקדוק החדש של ביטוי compose-exp כולל האילוצים הנדרשים שהוסברו לגביו.

```
(expression
 ("proc" "(" identifier ")" expression)
 proc-exp)

 (expression
   ____________________________________
   ____________________________________

 compose-exp)

(expression
 ("(" expression expression ")")
 call-exp)
```

2. **עדכון טיפוס הנתונים המופשט proc לייצוג פרוצדורה עם בנאי חדש**

להלן קטע הקוד של טיפוס הנתונים proc אותו נדרש לעדכן. השלימו במחברת הבחינה את העדכון הדרוש בו.

```
(define-datatype proc proc?
   (procedure
     (var symbol?)
     (body expression?)
     (env environment?))
   (composition
     ____________________
     ____________________
    )

 )
```

3. עדכון ההתנהגות של הביטוי החדש compose-exp לבניית פרוצדורה

להלן קטע קוד מתוך הפרוצדורה value-of, השלימו במחברת הבחינה את עדכון התנהגות הדקדוק החדש של ביטוי compose-exp כפי שהוסבר והודגם קודם.

```
(proc-exp (var body)
  (proc-val (procedure var body env)))

(compose-exp (_____________)

   _____________________________________
   _____________________________________
   _____________________________________

)

(call-exp (rator rand)
  (let ((proc1 (expval->proc (value-of rator env)))
        (arg (value-of rand env)))
   (apply-procedure proc1 arg)))
```

4. עדכון הפרוצדורה apply-procedure כך שתדע להפעיל גם פרוצדורה שהוגדרה ע"י הרכבה

להלן קטע הקוד של הפרוצדורה apply-procedure. השלימו במחברת הבחינה את עדכון הפרוצדורה כך שתדע להפעיל גם פרוצדורה מסוג composition.

```
(define apply-procedure
  (lambda (proc1 val)
    (cases proc proc1
      (procedure (var body saved-env)
        (value-of body (extend-env var val saved-env)))
      (composition (_____________)
            _____________________________
            _____________________________
            _____________________________
       )
  )))
```

שאלה 2 (30 נקודות)

בספר הלימוד בעמודים 113-120 מתוארת שפת "וירמוז"(IMPLICIT-REFS).

ברצוננו להרחיב שפה זו עם יכולת לעדכן לפרוצדורה שכבר הוגדרה את הסביבה איתה הוגדרה הפרוצדורה, באופן שנוכל להוסיף לסביבה של הפרוצדורה משתנים נוספים עם ערכים עבורם, בכל שלב שנחפוץ בכך בתוכנית, כך שבהפעלות הבאות של הפרוצדורה היא תכיר משתנים אלה במידה והיא פונה אליהם.

לשם כך, נוסיף לשפה ביטוי חדש בשם update-proc-env.

להלן הדקדוק המגדיר את הביטוי החדש שברצוננו להוסיף:

Expression ::= <***identifier***>.EnvAdd = ({***identifier*** : ***expression***}*(,))

env-proc-exp (id1 ids exps)

ביטוי **env-proc-exp** פועל על פרוצדורה שכבר הוגדרה (שמה נתון ע"י id1) ומעדכן את הסביבה של הפרוצדורה ע"י הוספת משתנים ששמם נתון ע"י ids וערכם נתון ע"י תוצאת הביטויים המוגדרים ע"י exps. במקרה ולא מוגדרת פרוצדורה כזו יש להדפיס הודעת שגיאה מתאימה.

ביטוי **update-proc-exp** אינו מחזיר תשובה (ולכן יש להחזיר ערך פיקטיבי כלשהו)

להלן מספר דוגמאות להמחשת השימוש בביטוי החדש.

דוגמה 1:

```
> (run "
    let a=100
    in
      let p= proc (w) -(w, -(a,b))
      in
        (p 400)")
environments.scm:30:10: apply-env: No binding for b
```

הסבר דוגמה 1:

דוגמה זו לא משתמשת בביטוי החדש. הדוגמה ממחישה הגדרת פרוצדורה p המתייחסת בגופה למשתנה b שלא קיים בזמן הגדרתה (כנראה המתכנת שכח להגדיר) ולכן כאשר מפעילים את p מקבלים שגיאה.

דוגמה 2:

```
> (run "
    let a=100
    in
      let p= proc (w) -(w, -(a,b))
      in
       begin
         <p>.EnvAdd = (b:250, c:700);
        (p 400)
       end")
(num-val 550)
>
```

הסבר דוגמה 2:

דוגמה 2, ממחישה שימוש בביטוי החדש. הדוגמה זהה לדוגמה הקודמת, אלא שכאן לפני ההפעלה, עודכנה הגדרת סביבת הפרוצדורה בתוספת של 2 משתנים נוספים, b ו-c , וכן כעת כאשר מפעילים את הפרוצדורה p לא מתקבלת שגיאה, ויש לנו תוצאת הפעלה.

דוגמה 3:

```
> (run "
    let a=100
    in
      <a>.EnvAdd = (b:250, c:700)")
interp.scm:120:22: env-proc-exp: a is not a procedure
>
```

הסבר דוגמה 3:

בדוגמה זו מנסים לעדכן סביבה של ישות שאינה פרוצדורה ולכן התקבלה שגיאה.

המשך השאלה בעמוד הבא

מה עליכם לממש בשאלה זו?

1. **(10 נקודות) עדכון תחביר השפה עם הדקדוק של הביטוי החדש env-proc-exp**

השלימו במחברת הבחינה את קטע הקוד הבא לעדכון דקדוק הביטוי החדש:

```
(expression
   (_____________________________________)

 env-proc-exp)
```

2. **(20 נקודות) מימוש התנהגות של הביטוי החדש env-proc-exp**

השלימו במחברת הבחינה את קטע הקוד הבא שאמור להתווסף לפרוצדורה value-of לטיפול בהתנהגות של הביטוי החדש:

```
(env-proc-exp (_____________)

   ____________________________________
   ____________________________________
   ____________________________________

 )
```

שאלה 3 (30 נקודות)

להלן נתונה תוכנית בשפת "ויקיש" (INFERRED):

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

ברצוננו לקבוע מהו טיפוס התוכנית (אם קיים). לשם כך, עליכם להשתמש באלגוריתם שנלמד בפרק 7 להקשת הטיפוס.

א. (7 נקודות)

הקצו לביטוי הנתון בשאלה ולתתי הביטויים שלו משתני-טיפוס (Type Variables) מתאימים.

ב. (8 נקודות)

חברו משוואות מתאימות לביטוי הנתון ולתתי הביטויים שלו לפי משתני הטיפוס שהקצתם בסעיף א'.

ג. (15 נקודות)

פתרו את המשוואות שהרכבתם בסעיף ב' צעד אחר צעד, תוך שימוש באלגוריתם שנלמד בפרק 7 בדומה למתואר בספר בעמודים 252-258. בסיום פתרון המשוואות רשמו מהו הטיפוס שהוקש עבור התוכנית במידה וקיים.

ב ה צ ל ח ה !