האוניברסיטה הפתוחה

י"ג באב תשפ"א

22

ביולי 2021

מס' שאלון - 532

סמסטר 2021ב

מס' מועד 78

4 / 20905

שאלון בחינת גמר

20905 - שפות תכנות

משך בחינה: 3 שעות

בשאלון זה **9** עמודים

מבנה הבחינה:

בבחינה 3 שאלות.

עליכם לענות על כל השאלות.

משקל השאלות מפורט בגוף השאלון.

שימו לב:
* את תשובותיכם יש לרשום על דפים בכתב יד ברור לסרוק וליצור קובץ PDF יחידולהעלותו בסיום הבחינה על פי ההנחיות שנשלחו אליכם.
* כתבו בכתב יד ברור ומסודר.

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

בהצלחה !!!

שאלה 1 (35 נקודות)

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

בשאלה זו נרצה לשנות את האופן של הגדרה והפעלה של פרוצדורות כך שלפרוצדורות יוכלו להיות פרמטרים עם ערכי ברירת מחדל, ובהפעלת פרוצדורה נוכל לבחור עבור פרמטר מסוים האם להשתמש בערך ברירת המחדל שלו או בארגומנט שישלח עבורו.

להלן הדקדוקים המגדירים מחדש את אופן הגדרת והפעלת פרוצדורה:

Expression **::=** proc ({ ***identifier*** = ***Expression*** }*(,)) ***Expression***

proc-exp (ids defs body)

Expression **::=** (***Expression*** { ***identifier*** = ***Expression*** }*(,))

call-exp (rator parms exps)

<u>הסבר על המרכיבים של הביטויים החדשים:</u>

- ביטוי proc-exp מורכב מ-ids שהיא רשימה של שמות הפרמטרים של הפרוצדורה, ומ-defs שהיא רשימה של ערכי ברירת המחדל עבור הפרמטרים, ומ-body שהוא גוף הפרוצדורה.
  ביטוי proc-exp מחזיר פרוצדורה.
- ביטוי call-exp מורכב מ-rator שהוא ביטוי המתאר את הפרוצדורה שיש להפעיל, מ-parms שהיא רשימה המכילה רק את שמות הפרמטרים שעבורן **<u>לא</u>** יעשה שימוש בערך ברירת המחדל, ומ- exps שהיא רשימה של ביטויים שאת ערכם יש לשלוח לפרמטרים שצוינו ב-parms. שמות הפרמטרים ב-parms לא חייבים להופיע באותו סדר שבו הופיעו בהגדרת הפרוצדורה. שאר הפרמטרים שלא צוינו ב-parms יקבלו את ערך ברירת המחדל שהוגדר עבורם בזמן הגדרת הפרוצדורה.

(המשך השאלה בעמוד הבא)

דוגמאות להמחשת אופן השימוש במרכיבים החדשים:

דוגמה 1:

```
> (run "
        let p = proc (a=10,b=20,c=30)
                -(c, -(a,b))
        in
          (p)")
(num-val 40)
```

הסבר דוגמה 1:

בדוגמה 1, הוגדרה פרוצדורה p עם 3 פרמטרים a, b, c בעלי ערכי ברירת מחדל 10,20,30 בהתאמה. הפרוצדורה מחזירה את ערכו של הביטוי c-a+b . בדוגמה זו, הפרוצדורה מופעלת ללא ארגומנטים, ולכן כל הפרמטרים יקבלו את ערכי ברירת המחדל. ולכן הוחזר ערך של 40 (על פי: 20+10-30)

דוגמה 2:

```
> (run "
        let p = proc (a=10,b=20,c=30)
                -(c, -(a,b))
        in
          (p b=50)")
(num-val 70)
```

הסבר דוגמה 2:

בדוגמה 2, מדובר על אותה פרוצדורה כמו בדוגמה 1, אלא שהפעם היא מופעלת כאשר הפרמטר b מקבל ערך של 50 ולא את ערך ברירת המחדל שלו, יתר הפרמטרים, a, c מקבלים את ערכי ברירת המחדל שלהם. ולכן הוחזר ערך של 70 (על פי: 50+10-30).

דוגמה 3:

```
> (run "
        let p = proc (a=10,b=20,c=30)
                -(c, -(a,b))
        in
          (p x=50)")
interp.scm:130:17: apply-procedure: wrong arguments to procedure
```

הסבר דוגמה 3:

בדוגמה 3, מדובר על אותה פרוצדורה כמו בדוגמה 1, אלא שהפעם הפרוצדורה מופעלת עם פרמטר בשם x שאינו מתאים לאף אחד מהפרמטרים של הפרוצדורה, ולכן התקבלה שגיאה.

דוגמה 4:

```
> (run "
        let p = proc (a=10,b=20,c=30)
                  -(c, -(a,b))
        in
          (p c=100, a=35)")
(num-val 85)
```

הסבר דוגמה 4:

בדוגמה 4, מדובר על אותה פרוצדורה כמו בדוגמה 1. הפעם הפרוצדורה מופעלת עם פרמטרים c ו-a עם ערכים 100 ו-35 בהתאמה, ערכו של b יהיה ערך ברירת המחדל שלו. שימו לב לסדר השרירותי של הפרמטרים הניתנים בהפעלת הפרוצדורה. במקרה זה התוצאה שהוחזרה היא 85 (על פי: 100-35+20)

יש לשים לב ולבדוק מקרים של שימוש לא תקין בתחביר (כגון: חוסר תאימות בשמות הפרמטרים המופיעים בהגדרת הפרוצדורה אל מול אלה המופיעים בהפעלתה, כמות פרמטרים נשלחים הגדולה יותר מכמות הפרמטרים להם מצפה הפרוצדורה, וכדומה)

במקרים של שימוש לא תקין, יש להדפיס הודעות שגיאה מתאימות.

ממשו והטמיעו את השינויים הדרושים בתוך שפת **"וישגור"** (**שפת PROC**).
הקפידו להסביר **היכן בדיוק** נדרשים שינויים ותוספות בקבצי המפרש, **ומהם** השינויים והתוספות.

הנחיות כלליות לפתרון:

1. הקפידו על כתב ברור, ועל קוד מבני ומסודר.
2. הקפידו על פתיחת וסגירת סוגריים במקומות הנכונים.
3. הפתרון אמור לשמור על הגדרות השפה המקוריות (פרט למקרים בהם נדרש שינוי כזה במפורש בשאלה).
4. על מנת לפשט קוד ארוך, כדאי ורצוי להיעזר בכתיבת פרוצדורות עזר (מחוץ ל-value-of או כאלה המוטמעות בתוכו).
5. בפתרונכם, הקפידו על אבחנה ושימוש נכון בין ביטוי (expression) לבין תוצאת חישוב ביטוי (expval) לבין identifier.
6. ניקוד יורד על אי ניתוח ומימוש נכון של הדקדוק הנתון בשאלה, כלומר יש להפקיד על זיהוי וטיפול נכון ב- terminals, non-terminals ופעולות של סגור ($^*\{\ \}$ או $^+\{\ \}$)

שאלה 2 (40 נקודות)

בספר הלימוד בעמודים 104-113 מתוארת **<u>שפת "ויפנה" (EXPLICIT-REFS)</u>**.

ברצוננו להרחיב שפה זו עם יכולות חדשות שיאפשרו הגדרה של מערך ומתן גישה לתאים שלו. נרצה בפרט לאפשר הגדרת מערכים מטיפוסים שונים (מערך של מספרים, מערך של בוליאניים, מערך של פרוצדורות) תאי המערכים שיוגדרו יהיו ניתנים לשינוי (Mutable)

להלן הדקדוק המגדיר את הביטויים החדשים שיש להטמיע בשפה:

Expression ::= ***Type*** [***Expression***] { { ***Expression*** }$^{+(,)}$ }

Type ::= # | ? | @

arr-exp (typ size initexps)

Expression ::= [***Expression , Expression***]

index-exp (arrexp index)

שימו לב, Type המוגדר בשאלה זו, אינו קשור כלל ל- Type שפגשנו בפרק 7.

<u>הסבר על מרכיבי הביטויים ואופן פעולתם:</u>

ביטוי **arr-exp** מורכב מהמרכיבים הבאים:

- typ – מציין את טיפוס המערך שיוגדר. כאשר, # מציין מערך של מספרים, ? מציין מערך של בוליאניים, @ מציין מערך של פרוצדורות.
- size – ביטוי שתוצאתו מספר המציין את גודל המערך (מספר תאי המערך)
- initexps – רשימה של ביטויים שערכיהם מהווים את ערכי אתחול המערך. (נדרש לספק ביטויים בהתאם לסוג המערך המוגדר)

ביטוי arr-exp מגדיר ומחזיר מערך בגודל המצוין ע"י size, תאי המערך יוכלו לקבל אליהם ערכים מסוג typ, והמערך יאותחל בערכים המוגדרים ע"י הביטויים initexps.

שימו לב, שיש כאן טיפוס נתונים חדש לשפה.

ביטוי **index-exp** מורכב מהמרכיבים הבאים:

- arrexp - ביטוי שתוצאתו היא מערך.
- index - ביטוי שתוצאתו היא מספר. המספר מציין את האינדקס של התא במערך אליו רוצים לפנות. האינדקסים מתחילים מ-0. הביטוי מחזיר reference (מצביע) לתא המבוקש במערך.

להלן דוגמאות להמחשת השימוש:

דוגמה 1:

```
> (run "let a= #[4]{10,20,30,40}
    in begin
        setref([a,2], 80);
        deref([a,2])
    end")
(num-val 80)
```

בדוגמה 1, מוגדר מערך של מספרים, בגודל 4 תאים, המאותחל לערכים 10,20,30,40. בהמשך יש פנייה לתא באינדקס 2 (התא השלישי) ועדכונו לערך של 80 במקום 30. לאחר מכן מוחזר תוכנו של התא באינדקס 2, וניתן לראות שהתוצאה היא 80.

דוגמה 2:

```
> (run "let a= ?[4]{10,20,30,40}
    in begin
        setref([a,2], 80);
        deref([a,2])
    end")
interp.scm:129:26: value-of: Array initialization mismatch array type
```

בדוגמה 2, מוגדר מערך של 4 בוליאניים, אך המערך מאותחל בערכים מספריים, ולכן זו נחשבת שגיאה, ומודפסת הודעת שגיאה בהתאם.

דוגמה 3:

```
> (run "let a= #[4]{10,20,30}
    in begin
        setref([a,2], 80);
        deref([a,2])
    end")
interp.scm:127:22: value-of: Array initialization mismatch array size
```

בדוגמה 3, מוגדר מערך של 4 מספרים, אך ניתן אתחול רק ל-3 מהתאים ועל כן זו שגיאה. מספר הערכים בחלק האיתחול אמור להיות תואם לגודל המערך. ולכן התקבל הודעת שגיאה מתאימה.

(המשך השאלה בעמוד הבא)

דוגמה 4:

```
> (run "let a= #[4]{10,20,30,40}
    in begin
        setref([a,7], 80);
        deref([a,2])
    end")
interp.scm:138:40: value-of: Bad array indexing
```

בדוגמה 4, מוגדר מערך בגודל 4 תאים של מספרים המאותחל בערכים 10,20,30,40 בהמשך יש ניסיון לפנות לתא באינדקס 7, אינדקס שאינו קיים עבור מערך זה. ולכן התקבלה הודעת שגיאה מתאימה

דוגמה 5:

```
> (run "let p1= proc (x) -(x,1)
        in let p2= proc (x) -(x,2)
           in let p3= proc (x) -(x,3)
              in let a= @[3]{p1,p2,p3}
    in begin
        setref([a,2], p1);
        (deref([a,2]) 9)
    end")
(num-val 8)
```

בדוגמה 5, הוגדר מערך (בשורה 4) בגודל 3 של פרוצדורות. המערך אותחל להכיל את הפרוצדורות p1,p2,p3 בהתאמה שהוגדרו קודם לכן. בהמשך (בשורה 6) משנים את תוכנו של התא באינדקס 2, ובמקום שיכיל את הפרוצדורה p3 שהייתה בו, הוא כעת מכיל את הפרוצדורה p1. לאחר מכן, מופעלת הפרוצדורה בתא 2, (הפרוצדורה p1), המקבלת מספר ומחזירה את הקודם שלו, לכן אם נשלח 9 נקבל 8.

שימו לב, יש להקפיד על עבודה עם חוקי וכללי השפה הנתונה בשאלה (שפת "ויפנה" – Explicit-Refs)

(המשך השאלה בעמוד הבא)

ממשו והטמיעו את הביטויים החדשים בתוך שפת **"ויפנה"** (**שפת EXPLICIT-REFS**).
הקפידו להסביר **היכן בדיוק** נדרשים שינויים ותוספות בקבצי המפרש, **ומהם** השינויים והתוספות.

יש לשים לב ולבדוק מקרים של שימוש לא תקין בתחביר.
במקרים של שימוש לא תקין, יש להדפיס הודעות שגיאה מתאימות כפי שהודגם.

הנחיות כלליות לפתרון:

1. הקפידו על כתב ברור, ועל קוד מבני ומסודר.
2. הקפידו על פתיחת וסגירת סוגריים במקומות הנכונים.
3. הפתרון אמור לשמור על הגדרות השפה המקוריות. (כגון אופן ניהול הסביבה וסוגי הערכים בה, אופן ניהול ההחסן והערכים שניתן לשמור בו, וכדומה)
4. על מנת לפשט קוד ארוך, כדאי ורצוי להיעזר בכתיבת פרוצדורות עזר (מחוץ ל-value-of או כאלה המוטמעות בתוכו)
5. בפתרונכם, הקפידו על אבחנה ושימוש נכון בין ביטוי (expression) לבין תוצאת חישוב ביטוי (expval) לבין identifier
6. ניקוד יורד על אי ניתוח ומימוש נכון של הדקדוק הנתון בשאלה, כלומר יש להפקיד על זיהוי וטיפול נכון ב- terminals, non-terminals ופעולות של סגור ($^*\{\ \}$ או $^+\{\ \}$)

המשך הבחינה בעמוד הבא

שאלה 3 (25 נקודות)

להלן נתונה תוכנית בשפת "ויקיש" (INFERRED):

```
let p1 = proc (x:?) (x 9)
in
  let p2 = proc (y:?) zero?(y)
  in
     (if (p1 p2) then p2 else proc (z:?) zero?(-(z,5))   7)
```

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

א. הקצו לביטוי הנתון בשאלה ולתתי הביטויים שלו משתני-טיפוס (Type Variables) מתאימים, וחברו משוואות מתאימות לביטוי הנתון ולתתי הביטויים שלו.

ב. פתרו את המשוואות שהרכבתם בסעיף א' תוך שימוש ב- substitution ו- unification בדומה למתואר בספר בעמודים 252-258. בסיום פתרון המשוואות רשמו מהו הטיפוס שאותו הקשתם עבור התוכנית הנתונה.

ב ה צ ל ח ה !