ה א ו נ י ב ר ס י ט ה ה פ ת ו ח ה

20905

שפות תכנות

חוברת הקורס- אביב 2023ב

כתב: דני כלפון

מרץ 2023 - סמסטר אביב – תשפ"ג

פנימי – לא להפצה.

תוכן העניינים

אל הסטודנט,

אנו מקדמים את פניך בברכה עם הצטרפותך אל הלומדים בקורס "שפות תכנות". בחוברת זו תמצא לוח הזמנים של הקורס, מטלות ותנאים לקבלת נקודות זכות בקורס.

לקורס קיים אתר באינטרנט בו תמצאו חומרי למידה נוספים, אותם מפרסם/מת מרכז/ת ההוראה. בנוסף, האתר מהווה עבורכם ערוץ תקשורת עם צוות ההוראה ועם סטודנטים אחרים בקורס. פרטים על למידה מתוקשבת ואתר הקורס, תמצאו באתר שה"ם בכתובת:

http://telem.openu.ac.il

מידע על שירותי ספרייה ומקורות מידע שהאוניברסיטה מעמידה לרשותכם, תמצאו באתר הספריה באינטרנט www.openu.ac.il/Library.

שעות הייעוץ הטלפוני שלי יפורסמו סמוך לפתיחת הסמסטר באתר הקורס. אפשר ורצוי לפנות אלי בדואר אלקטרוני: dannyca@openu.ac.il, תוך ציון שם מלא, מספר תעודת זהות ומספר טלפון. פגישות חובה לתאם מראש. לצורך בירורים בנושאים אדמיניסטרטיביים יש לפנות בכתב או טלפונית למחלקות האוניברסיטה הפתוחה.

הקורס הוא תכנותי באופיו, וכולל מטלות תכנותיות החשובות להבנת החומר ותרגולו. מומלץ שתקדישו זמן ראוי ללמידת חומר הקורס, שכן זה קורס השונה באופיו מקורסים תכנותיים אחרים המוכרים לכם באו"פ. הקורס כולל פיתוח מפרשים לשפות תכנות פשוטות המדגימות עקרונות הקיימים בשפות תכנות מודרניות. השתתפות במפגשים, הקדשת זמן ראוי ללמידת החומר והגשת המטלות הם הדרך הנכונה לסיום הקורס בהצלחה.

בברכת לימוד מהנה

כלפון דני

מרכז ההוראה בקורס

1. לוח זמנים ופעילויות (20905 / ב2023)

שבוע לימוד	תאריכי שבוע הלימוד	יחידת הלימוד המומלצת	מפגשי ההנחיה*	תאריך אחרון למשלוח ממ"ן (למנחה)
1	10.03.2023-5.03.2023	1	מפגש ראשון	
2	17.03.2023-12.03.2023	2		
3	24.03.2023-19.03.2023	2	מפגש שני	ממ"ן 11 24.3.2023
4	31.03.2023-26.03.2023	2		
5	07.04.2023-02.04.2023 (ד-ו פסח)	3	מפגש שלישי	
6	14.04.2023-09.04.2023 (א-ד פסח)	3		ממ"ן 12 09.04.2023
7	21.04.2023-16.04.2023 (ג יום הזכרון לשואה)	3	מפגש רביעי	
8	28.04.2023-23.04.2023 (ג יום הזיכרון, ד יום העצמאות	3		
9	05.05.2023-30.04.2023	3	מפגש חמישי	ממ"ן 13 30.04.2023

** התאריכים המדויקים של המפגשים הקבוצתיים מופיעים ב"לוח מפגשים ומנחים".*

לוח זמנים ופעילויות - המשך

שבוע הלימוד	תאריכי שבוע הלימוד	יחידת הלימוד המומלצת	מפגשי ההנחיה*	תאריך אחרון למשלוח הממ"ן (למנחה)
10	12.05.2023-07.05.2023 (ג ל"ג בעומר)	4		
11	19.05.2023-14.05.2023	4	מפגש שישי	ממ"ן 14 14.5.2023
12	26.05.2023-21.05.2023 (ו שבועות)	4		
13	02.06.2023-28.05.2023	7	מפגש שביעי	ממ"ן 15 28.05.2023
14	09.06.2023-04.06.2023	7		
15	16.06.2023-11.06.2023	7	מפגש שמיני	ממ"ן 16 11.06.2023

מועדי בחינות הגמר יפורסמו בנפרד

** התאריכים המדויקים של המפגשים הקבוצתיים מופיעים ב"לוח מפגשים ומנחים".*

2. תיאור המטלות

לתשומת לבכם!

כדי לעודדכם להגיש לבדיקה מספר רב של מטלות הנהגנו את ההקלה שלהלן:

אם הגשתם מטלות מעל למשקל המינימלי הנדרש בקורס, **המטלות** בציון הנמוך ביותר, שציוניהן נמוכים מציון הבחינה (**עד שתי מטלות**), לא יילקחו בחשבון בעת שקלול הציון הסופי.

זאת בתנאי שמטלות אלה **אינן חלק מדרישות החובה בקורס** ושהמשקל הצבור של המטלות האחרות שהוגשו, מגיע למינימום הנדרש.

זכרו! ציון סופי מחושב רק לסטודנטים שעברו את בחינת הגמר בציון 60 ומעלה והגישו מטלות כנדרש באותו קורס.

שימו לב!

בקורס זה **חובה** להגיש את המטלות בזמן, בהתאם לתאריך ההגשה המצוין עליהן. במקרים חריגים, כאשר יש סיבה מוצדקת להגשת המטלה באיחור, יש לפנות **בכתב** בדואר אלקטרוני אל מרכז ההוראה בקורס. **את הבקשה יש להגיש מראש!** יש לצרף לבקשה אישורים רשמיים, להצדקת סיבת הבקשה.

מטלות שיוגשו באיחור ללא אישור יבדקו והציון שיוזן עבורן יהיה 0, ללא תלות בציון של הבדיקה.

שימו לב, טיפול בבקשות שנשלחות לאחר מועד ב׳ של הסמסטר אינו בסמכות מרכז ההוראה, ויש להפנותן את האחראית על פניות סטודנטים של החטיבה למדעי המחשב.

להלן פירוט הניקוד לכל מטלה:

ממ״ן	ניקוד
11	5
12	5
13	5
14	5
15	5
16	5

3. התנאים לקבלת נקודות זכות

כדי לקבל נקודות זכות בקורס זה עליך לעמוד בדרישות הבאות:

א) **צבירת משקל של לפחות 20 נקודות במטלות.**

ב) ציון של לפחות 60 נקודות בבחינת הגמר.

ג) ציון סופי בקורס של 60 נקודות לפחות.

לתשומת לבכם:

מדיניות קורס זה היא לאשר הזנת ציון אפס במטלות שלא הוגשו כנדרש בקורס.

סטודנטים אשר לא הגישו את מכסת המטלות המינימאלית לעמידה בדרישות הקורס ולקבלת זכאות להיבחן, ומבקשים שמטלות חסרות יוזנו בציון אפס, יפנו למוקד הפניות והמידע בטלפון **09-7782222** או **יעדכנו בעצמם** באתר שאילתא http://www.openu.ac.il/sheilta **קורסים ⇦ ציוני מטלות ובחינות ⇦ הזנת ציון 0 למטלות רשות שלא הוגשו.**

יש לקחת בחשבון כי מטלות אשר יוזן להן ציון אפס ישוקללו בחישוב הציון הסופי ובכך יורידו ציון זה ולא ניתן יהיה להמירן במטלות חלופיות במועד מאוחר יותר. על כן קיימת אפשרות שסטודנט אשר יעבור את הבחינה בהצלחה ייכשל בקורס (כשהממוצע המשוקלל של המטלות והבחינה יהיה נמוך מ- 60).

כלל זה איננו חל על מטלות חובה או על מטלות שנקבע עבורן ציון מינימום.

מטלת מנחה (ממ"ן) 11

הקורס: שפות תכנות (20905)

חומר הלימוד למטלה: פרק 1 בספר הלימוד, פרק 3 במדריך הלמידה, חלקו השני של מדריך הלמידה העוסק בשפת scheme ובסביבת העבודה racket וכן מדריכים באתר הקורס.

מספר השאלות: 4 **משקל המטלה:** 5 נקודות

סמסטר: 2023ב **מועד הגשה:** **24.3.2023**

- שליחת המטלות תתבצע רק באמצעות מערכת המטלות המקוונת באתר הבית של הקורס.

שימו לב, בכל מקום במטלה בו נכתבה המילה סמל, הכוונה היא ל-scheme symbol.

שאלה 1 (30 נקודות)

א) הפרוצדורה append בשפת Scheme מקבלת 2 רשימות ומאחדת אותם לרשימה אחת ע"י שרשור של הרשימה השניה לסוף של הראשונה. ממשו את append בעצמכם (כמובן אסור להשתמש ב- append של השפה). קראו לפרוצדורה בשם my_append להלן דוגמא:

```
>(my_append '(a b c) '(x y z))
(a b c x y z)
```

ב) ממשו כעת את append ע"י שימוש ב- foldr וללא רקורסיה ישירה (כלומר, במימוש לא תופיע קריאה ישירה לפרוצדורה אותה אתם נדרשים לממש). קראו לפרוצדורה בשם my_append_fr

שאלה 2 (20 נקודות)

כתבו פרוצדורה בשם filter המקבלת כפרמטרים: פרדיקט ורשימה. הפרוצדורה מחזירה רשימה חדשה המכילה רק את האיברים מהרשימה המקורית שעבורם ערכו של הפרדיקט היה אמת. הפרדיקט הוא פרוצדורה המקבלת ארגומנט יחיד ומחזירה #t או #f למשל, הפרדיקט number? מחזיר #t אם הארגומנט שלו הוא מספר, אחרת מחזיר #f.
להלן דוגמא:

```
>(filter even? '(1 2 3 4 5 6))
(2 4 6)
```

ממשו את filter באמצעות שימוש ב- foldr, וללא רקורסיה ישירה (כלומר, במימוש לא תופיע קריאה ישירה לפרוצדורה אותה אתם נדרשים לממש)

שאלה 3 (20 נקודות)

כתבו פרוצדורה בשם set-dif המקבלת 2 רשימות שטוחות (ללא רשימות מקוננות וללא חזרות של איברים באותה רשימה), כל רשימה מייצגת קבוצה של איברים. הפרוצדורה תחזיר רשימה המייצגת את **ההפרש הסימטרי** בין הקבוצות. אין להשתמש בפעולות מובנות דומות על קבוצות הקיימות בשפה, נדרש לממש בעצמכם ע״י שימוש בפעולות בסיסיות על רשימות

לדוגמא:

```
>(set-dif  '(a b c d)  '(x b d w))
(a c  x w)
```

שאלה 4 (30 נקודות)

הפרוצדורה foo להלן מקבלת כפרמטר רשימה ls, היכולה להכיל גם תתי-רשימות עם רמת קינון שרירותית (כלומר איברי הרשימה יכולים להיות בעצמם רשימות שאף איבריהם הם רשימות וכו׳) וכן, משתנה s המתפקד כמונה עם ערך התחלתי של 0. הפרוצדורה מחזירה זוג שה-car שלו הוא רשימה המורכבת מהאיברים ב- ls תוך מחיקת האיברים שערכם היה זוגי, ואילו ה-cdr של הזוג הוא מספר המציין את כמות המספרים הזוגיים שהיו ב-ls.

להלן דוגמא:

```
>(foo `(2 3 (7 4 5 6) 8 (9) 2) 0)
((3 (7 5) (9)) . 5)
```

להלן הפרוצדורה foo, השלימו בה את החסר:

```
(define foo
  (lambda (ls s)
    (cond
        [(null?  ls)  `(() .  ,s)]
        [(pair?  (car ls))
          (let ((p  (foo (car ls) s)))
            (let ((p1 (foo (cdr ls) (cdr p))))
              `(,(cons _____  _____) .  ,(cdr  p1))))]
        [(or (null?  (car ls))  (odd? (car ls)))
         (let  ((p  (foo (cdr ls) s)))
           `(,(cons  _______  _______) .  ,(cdr p)))]
        [else  (let ((p (foo  (cdr ls) s)))
                 `(,(car p) . ,(add1 (cdr p))))]))))
```

מטלת מנחה (ממ"ן) 12

הקורס: שפות תכנות (20905)

חומר הלימוד למטלה: פרק 2 בספר הלימוד, פרק 4 במדריך הלמידה.

מספר השאלות: 3 **משקל המטלה:** 5 נקודות

סמסטר: 2023ב **מועד הגשה:** **9.4.2023**

- שליחת המטלות תתבצע רק באמצעות מערכת המטלות המקוונת באתר הבית של הקורס.

לצורך פתרון מטלה זו עליכם ללמוד היטב את פרק 2 בספר הלימוד בליווי של פרק 4 במדריך הלמידה.

שאלה 1 (50 נקודות)

פולינום היא פונקציה מהצורה: $p(x)= a_0+a_1x+a_2x^2+\ldots+a_kx^k$. דרגת הפולינום היא החזקה הגבוהה ביותר בעלת מקדם שונה מאפס.

poly הוא טיפוס נתונים מופשט (ADT) המייצג פולינום ותומך בפעולות הבאות:

- **zero** – מחזירה פולינום ריק (ללא איברים), כלומר את $p(x)=0$.
- **make-poly** – מקבלת כפרמטרים מקדם (coefficient) וחזקה (exponent) ומחזירה פולינום מתאים.

  למשל, (**make-poly** 3 4) מחזירה את הפולינום $p(x)=3x^4$
- **add-poly** – מקבלת זוג פולינומים ומחזירה פולינום המייצג את סכומם.

  למשל,

  (**add-poly** (**make-poly** 3 4) (**make-poly** 2 1))

  מחזירה את הפולינום $p(x)= 2x^1 + 3x^4$.
- **degree**– מקבלת פולינום ומחזירה את דרגתו.

  למשל, (**degree** (**add-poly** (**make-poly** 3 4) (**make-poly** 2 1))) מחזירה את הערך 4.
- **coeff** – מקבלת פולינום וחזקה ומחזירה את המקדם של האיבר בעל החזקה הנתונה. אם אין איבר כזה מוחזר 0.

  למשל, (**coeff** (**make-poly** 3 4) 4) מחזירה 3.

  אבל (**coeff** (**make-poly** 3 4) 9) מחזירה 0.
- **is-zero?** – מקבלת פולינום ובודקת האם הוא הפולינום הריק. הפעולה מחזירה ערך בוליאני מתאים.

  למשל, (**is-zero?** (**zero**)) מחזירה true. ואילו, (**is-zero?** (**make-poly** 3 4)) מחזירה false.

- **print-poly** – מקבלת פולינום וממירה ומחזירה אותו בצורת רשימה המורכבת מזוגות סדורים, כאשר כל זוג מייצג מקדם וחזקה של גורם בפולינום.
- **Calc-poly** – מקבלת פולינום וערך מספרי, ומחשבת את ערכו של הפולינום עבור הערך הנתון.

א) כתבו אפיון לחתימות של הפעולות. סווגו את הפעולות השונות לסוגים (בנאים, מחלצים, מנבאים).

ב) כתבו אפיון אלגברי לכל אחת מהפעולות המגדיר את הסמנטיקה (המשמעות) שלהם.

ג) ממשו את **poly** ואת המנשק לעבודה איתו (הפעולות שהוגדרו למעלה). ע״י שימוש בכלי define-datatype

שאלה 2 (30 נקודות)

שאלה זו עוסקת בתחביר מוחשי (concrete syntax) ובתחביר מופשט (abstract syntax).

ביטוי חשבוני בכתיב prefix (כתיב פולני) הוא ביטוי שבו תחילה רושמים את הפעולה החשבונית שרוצים לבצע ולאחריה את האופרנדים הדרושים. בשאלה זו נעסוק רק בפעולת חיסור.

להלן נתון דקדוק המגדיר prefix-list שהיא רשימה שכוללת בתוכה ביטוי בכתיב prefix.

Prefix-list ::= (***Prefix-exp***)

Prefix-exp ::= ***Int***

::= - ***Prefix-exp*** ***Prefix-exp***

למשל,הביטוי בכתיב המוחשי (7 12 - 4 - 2 3 - -) הוא *Prefix-list* חוקי.

להלן נתון ייצוג של *Prefix*-exp באמצעות define-datatype:

```
(define-datatype  prefix-exp prefix-exp?
  (const-exp
    (num integer?))
  (diff-exp
    (operand1 prefix-exp?)
    (operand2 prefix-exp?)))
```

הביטוי הבא הוא דוגמא לביטוי הכתוב בכתיב מוחשי:

(7 12 - 4 - 2 3 - -)

להלן עץ תחביר מופשט (AST) עבור ביטוי זה :

```
(diff-exp
  (diff-exp
    (const-exp 3)
    (const-exp 2))
  (diff-exp
    (const-exp 4)
    (diff-exp
      (const-exp 12)
      (const-exp 7))))
```

כתבו פרוצדורה בשם parse-prefix המקבלת רשימה המייצגת *Prefix-list* חוקי (בתחביר מוחשי)

וממירה אותו לתחביר מופשט. (מדפיסה את ה-AST)

רמז : כתבו פרוצדורה שמקבלת רשימה ומחזירה *Prefix-exp* וגם את יתרת הרשימה שעדיין לא טופלה.

לפני פתרון השאלה רצוי לחזור על הדוגמא להמרת ביטוי מתחביר מוחשי לתחביר אבסטרקטי המופיעה בעמוד 53 בספר הלימוד.

שאלה 3 (20 נקודות)

בעמוד 40 בספר הלימוד נתון מימוש לטיפוס הנתונים **סביבה.** מימוש זה עושה שימוש **<u>בייצוג פרוצדורלי לייצוג הסביבה.</u>**

על סמך **<u>מימוש זה</u>**, כתבו פרוצדורה בשם count-binding המקבלת סביבה e ומשתנה v ובודקת ומחזירה את מספר הכריכות שיש ל- v תחת הסביבה e.

שימו לב, בשאלה זו הייצוג של סביבה הוא פרוצדורלי.

מטלת מנחה (ממ"ן) 13

הקורס: שפות תכנות (20905)

חומר הלימוד למטלה: פרק 3 בספר הלימוד, פרק 5 במדריך הלמידה, נספח B בספר הלימוד.

מספר השאלות: 3 **משקל המטלה:** 5 נקודות

סמסטר: 2023ב **מועד הגשה:** **30.4.2023**

- שליחת המטלות תתבצע רק באמצעות מערכת המטלות המקוונת באתר הבית של הקורס.

לצורך פתרון מטלה זו עליכם ללמוד היטב את פרק 3 בספר הלימוד בליווי של פרק 5 במדריך הלמידה, וכן בנספח B בספר הלימוד.

שאלה 1 (20 נקודות)

א) פתרו את תרגיל 3.7 בספר הלימוד בעמוד 72 (הוספת פעולות חשבון נוספות בנוסף לפעולת חיסור שהיא חלק מהשפה) הוסיפו את הפעולות: כפל, חיבור, וחילוק בשלמים, השתמשו בסימנים * עבור כפל, + עבור חיבור, / עבור חילוק.

ב) פתרו את תרגיל 3.8 בספר הלימוד בעמוד 73.

שאלה 2 (30 נקודות)

א) פתרו את תרגיל 3.9 מספר הלימוד בעמוד 73

ב) פתרו את תרגיל 3.10 מספר הלימוד בעמוד 73.

שאלה 3 (50 נקודות)

הרחיבו את שפת LET (שפת "ויהי") בביטויים חדשים המאפשרים הגדרת מערך, וגישה לאיבר באינדקס מסוים במערך.

להלן הדקדוקים שנדרש לממש:

Expression ::= **array** { { *Expression* }$^{+(,)}$ }

array-exp (exps)

Expression ::= < *Expression* >[*Expression*]

index-exp (arr indx)

הביטוי array-exp מחזיר מערך המורכב מערכם של הביטויים המתוארים ע״י exps.
הביטויים המתוארים ע״י exps יכולים להיות ממגוון הביטויים בשפה.

הביטוי index-exp מחזיר את ערכו של האיבר במערך הנתון ע״י הביטוי arr הנמצא באינדקס הנתון ע״י ערכו של הביטוי indx. יש להתייחס לאינדקסים החל מ-1

להלן דוגמא להמחשת השימוש בביטוי החדש:

```
let A = array {10, -(5,7) , zero?(8), array {1,2,3}, 12  }
in
  -(<A>[1] ,  <<A>[4]>[2])
```

הסבר הדוגמה:
בתחילה מוגדר מערך A המכיל איברים שונים, בפרט האיבר באינדקס 4 הוא בעצמו מערך.

תוכנית זו מחזירה את תוצאת פעולת החיסור בין האיבר הראשון במערך A, לבין האיבר שני במערך שנמצא באינדקס 4 במערך A. על כן, תוצאתה של תוכנית זו היא: (num-val 8)

מטלת מנחה (ממ"ן) 14

הקורס: שפות תכנות (20905)

חומר הלימוד למטלה: פרק 3 בספר הלימוד, פרק 5 במדריך הלמידה, נספח B בספר הלימוד.

מספר השאלות: 3 **משקל המטלה:** 5 נקודות

סמסטר: 2023ב **מועד הגשה:** **14.5.2023**

- שליחת המטלות תתבצע רק באמצעות מערכת המטלות המקוונת באתר הבית של הקורס.

לצורך פתרון מטלה זו עליכם ללמוד היטב את פרק 3 בספר הלימוד בליווי של פרק 5 במדריך הלמידה, וכן בנספח B בספר הלימוד.

<u>שאלה 1 (30 נקודות)</u>

בשפת "**וישגור**" פרוצדורות מחושבות על פי כריכה לקסיקלית. דרך אחרת לחשב פרוצדורות היא ע"י כריכה דינמית (Dynamic binding). בכריכה דינמית, גוף הפרוצדורה מחושב באמצעות הרחבת הסביבה ממנה מתבצעת הקריאה לפרוצדורה. בניגוד לכריכה לקסיקלית שבה גוף הפרוצדורה מחושב על פי הסביבה בה מוגדרת הפרוצדורה.

נסתכל למשל על התוכנית הבאה בשפת "**וישגור**":

```
let a=3
in let p = proc (x)  -(x,a)
      a=5
   in –(a, (p 2))
```

בכריכה דינמית, המשתנה a המופיע בגוף הפרוצדורה כרוך לערך 5 ולא לערך 3.

שימו לב, בדוגמא זו שולבה גם יכולת להגדיר מספר ביטויים בתוך ביטוי let.

א) שנו את שפת "**וישגור**" כך שפרוצדורות יחושבו רק בכריכה דינמית. לשם כך השתמשו במימוש שבו פרוצדורות מיוצגות ע"י מבנה נתונים.

ב) כזכור שפת "**וישגור**" מאפשרת כתיבת פרוצדורות שאינן רקורסיביות. בסעיף א' שפת "**וישגור**" שונתה, ופרוצדורות מחושבות בכריכה דינמית, יכולת זו מאפשרת כעת לכתוב פרוצדורות רקורסיביות. הסבירו מדוע?

ג) כיתבו באמצעות הפתרון של סעיף א', פרוצדורה רקורסיבית המקבלת פרמטר n ומחזירה את !n (עצרת של n).

שאלה 2 (35 נקודות)

א) בשפת "וישגור" (שפת PROC) הוגדר שלפרוצדורה יש רק ארגומנט יחיד. אך ניתן לעקוף מגבלה זו ולכתוב פרוצדורה עם מספר ארגומנטים ע"י שימוש בפרוצדורות שמחזירות בעצמן פרוצדורות.

נסתכל למשל על התוכנית הבאה בשפת "וישגור" :

```
let f = proc (x) proc (y) …
in ((f 3) 4)
```

טכניקה זו נקראת Currying וכבר נתקלנו בה במסגרת השיעור הראשון בקורס.
כתבו בשפת "וישגור" באמצעות שימוש בטכניקה זו, פרוצדורה בעלת 2 ארגומנטים x ו- y המחזירה תשובה בוליאנית (כמובן בצורת exp-val) של true כאשר מתקיים x=2y. אחרת יוחזר false. למשל, עבור המספרים x=6 ו- y=3 יוחזר true.

יש לפתור רק על ידי שימוש בדקדוק הקיים של שפת PROC (לא ניתן להרחיב את השפה במרכיבים חדשים)

ב) להלן תוכנית בשפת "וישגור".

```
let makemult = proc (maker)
                proc (x)
                  if zero? (x)
                  then 0
                  else  -(((maker maker)  -(x,1)) , -4)
in  let times4 = proc (x) ((makemult makemult) x)
    in  (times4  3)
```

מהו הערך של תוכנית זו?

ג) כתבו **בשפת "וישגור"** פרוצדורה בשם f המקבלת מספר שלם חיובי x ומחזירה תשובה בוליאנית של true (כמובן בצורת expval) האם x הוא מהצורה 2^n ?
לשם כך השתמשו בטכניקה מסעיף ב', וכן ב- Currying.
את הבדיקה יש לבצע עלפי העיקרון הרקורסיבי הבא :
מספר X הוא חזקה של 2 אם ערכו 1 או
שניתן להציג את X בצורה של 2*Y כאשר Y הוא חזקה של 2

שאלה 3 (35 נקודות)

שאלה זו עוסקת בשדרוג של **שפת "וישגור" (שפת PROC)** 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:

$$\textit{Expression} ::= \text{proc}\ (\ \{\ \textit{identifier} = \textit{Expression}\ \}^{*(,)}\ )\ \textit{Expression}$$

proc-exp (ids defs body)

$$\textit{Expression} ::= (\textit{Expression}\ \{\ \textit{identifier} = \textit{Expression}\ \}^{*(,)}\ )$$

call-exp (rator parms exps)

הסבר על המרכיבים של הביטויים החדשים:

- 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 **לא** 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 10,20,30בהתאמה. 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: 30-10+20)

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: 30-10+50)

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)

ממשו והטמיעו את השינויים הדרושים בתוך שפת **"וישגור"** (**שפת PROC**).

הקפידו להסביר **היכן בדיוק** נדרשים שינויים ותוספות בקבצי המפרש, **ומהם** השינויים והתוספות

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.

מטלת מנחה (ממ"ן) 15

הקורס: שפות תכנות (20905)

חומר הלימוד למטלה: פרק 4 בספר הלימוד, פרק 6 במדריך הלמידה.

מספר השאלות: 2 **משקל המטלה:** 5 נקודות

סמסטר: 2023ב **מועד הגשה:** **28.5.2023**

- שליחת המטלות תתבצע רק באמצעות מערכת המטלות המקוונת באתר הבית של הקורס.

לצורך פתרון מטלה זו עליכם ללמוד היטב את פרק 4 בספר הלימוד בליווי של פרק 6 במדריך הלמידה.

שאלה 1 (50 נקודות)

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

ברצוננו להרחיב שפה זו עם ביטוי חדש בשם foreach-exp.

להלן הדקדוק המגדיר את הביטוי החדש שברצוננו להוסיף:

Expression ::= **foreach** *identifier* **in** **(** {*identifier*}$^{+(,)}$ **)** *Expression*

foreach-exp (id1 ids body)

לביטוי **foreach-exp** המרכיבים הבאים:

- **id1** – שם של משתנה חדש עבור חישוב הביטוי.
- **ids** – רשימת שמות משתנים (שאמורים להיות מוגדרים כבר)
- **body** – ביטוי המשמש כגוף ה- foreach

ביטוי foreach, אמור לבצע פעולה זהה על מספר משתנים, בכל פעם id1 יתפקד כאחד המשתנים המוגדרים ע"י ids (החל מהראשון וכלה באחרון) ועבורו יש לבצע את body.

ביטוי foreach אינו מחזיר תשובה, לכן יש להחזיר ערך סתמי של num-val 23

להלן דוגמא :

```
let a=3
in let b=4
   in let c= 5
      in
        begin
           foreach x in (a,b,c) set x= -(x,1)
           -(a, -(0, -(b, -(0,c))))
        end
```

בדוגמה זו, מוגדרים 3 משתנים a,b,c עם ערכים 3,4,5 בהתאמה. ביצוע ביטוי foreach גורם להקטנת ערכו של כל אחד מהם ב-1, ל- 2,3,4 בהתאמה. ערכה של כל התוכנית בדוגמה, הוא כערכו של הביטוי האחרון בבלוק ה-begin המחזיר את סכומם של המשתנים a,b,c ולכן התשובה המוחזרת היא (num-val 9)

<u>ממשו והטמיעו</u> את הביטויים החדשים בתוך שפת **<u>"וירמוז"</u>** (**<u>שפת IMPLICIT-REFS</u>**).

הקפידו להסביר **<u>היכן בדיוק</u>** נדרשים שינויים ותוספות בקבצי המפרש, **<u>ומהם</u>** השינויים והתוספות.

שאלה 2 (50 נקודות)

בספר הלימוד בעמודים 104-113 מתוארת **שפת "יפנה" (EXPLICIT-REFS)**.

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:

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

$$\textit{Type} ::= \#\ |\ ?\ |\ @$$

arr-exp (typ size initexps)

$$\textit{Expression} ::= [\ \textit{Expression}\ ,\ \textit{Expression}]$$

index-exp (arrexp index)

<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.

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)

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.

מטלת מנחה (ממ"ן) 16

הקורס: שפות תכנות (20905)

חומר הלימוד למטלה: פרק 7 בספר הלימוד, פרק 7 במדריך הלמידה.

מספר השאלות: 1 **משקל המטלה:** 5 נקודות

סמסטר: 2023ב **מועד הגשה:** **11.6.2023**

- שליחת המטלות תתבצע רק באמצעות מערכת המטלות המקוונת באתר הבית של הקורס.

לצורך פתרון מטלה זו עליכם ללמוד היטב את פרק 7בספר הלימוד בליווי הפרק המתאים במדריך הלמידה.

שאלה 1 (100 נקודות)

להלן נתונה תוכנית בשפת "ויקיש" (INFERRED):

```
proc (x:?)
  if (x proc (y:?) zero?(-(y,3)))
  then 80
  else 90
```

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

א. (40 נקודות)

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

ב. (60 נקודות)

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.