♣ ה א ו נ י ב ר ס י ט ה ה פ ת ו ח ה

20905

שפות תכנות

חוברת הקורס- קיץ 2022ג

כתב: דני כלפון

יולי 2022 - סמסטר קיץ – תשפ"ב

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 /ג2022)

שבוע הלימוד	תאריכי שבוע הלימוד	יחידת הלימוד המומלצת	מפגשי ההנחיה*	תאריך אחרון למשלוח הממ"ן (למנחה)
1	8.7.2022-3.7.2022	1	מפגש ראשון	
2	15.7.2022-10.7.2022	2	מפגש שני	
3	22.7.2022-17.7.2022	2	מפגש שלישי	ממ"ן 11 17.7.2022
4	29.7.2022-24.7.2022	3	מפגש רביעי	ממ"ן 12 24.7.2022
5	5.8.2022-31.7.2022	3	מפגש חמישי	
6	12.8.2022-7.8.2022 (א צום ט' באב)	3	מפגש שישי	ממ"ן 13 5.8.2022
7	19.8.2022-14.8.2022	3,4	מפגש שביעי	ממ"ן 14 12.8.2022
8	26.8.2022-21.8.2022	4	מפגש שמיני	ממ"ן 15 26.8.2022
9	2.9.2022-28.8.2022	7		ממ"ן 16 2.9.2022

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: 2022ג **מועד הגשה:** **17.7.2022**

- 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 (20 נקודות)

כתבו פרוצדורה בשם my_flatten המקבלת רשימה כפרמטר. הפרוצדורה מחזירה את תוצאת שיטוח הרשימה.

להלן דוגמא:

```
>(my_flatten  '( (1 2)  ((3))  (4 5 6)))
'(1 2 3 4 5 6)
```

שאלה 2 (30 נקודות)

א) כתבו פרוצדורה בשם my_count המקבלת כפרמטרים פרדיקט ורשימה.
הפרוצדורה מחזירה כמה איברים ברשימה מקיימים את הפרדיקט.
להלן דוגמה:

```
>(my_count  positive?  '(1 3 -7 9 -2))
3
```

את הפרוצדורה ממשו באמצעות רקורסיה.

ב) כתבו פרוצדורה בשם my_count_foldr המחזירה תוצאה זהה לזו של my_count ,
הפעם ממשו אותה באמצעות שימוש ב- foldr וללא שימוש ברקורסיה (כלומר,
במימוש לא תופיע קריאה ישירה לפרוצדורה אותה אתם נדרשים לממש).

שאלה 3 (20 נקודות)

כתבו פרוצדורה בשם my_partition המקבלת פרדיקט ורשימה כפרמטרים. הפרוצדורה מחזירה רשימה המורכבת מ-2 תתי רשימות, האחת תכיל איברים שקיימו את הפרדיקט, והשניה איברים שלא קיימו את הפרדיקט.

לדוגמה:

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

שאלה 4 (30 נקודות)

כתבו פרוצדורה בשם my_permutations המקבלת כפרמטר רשימה. הפרוצדורה תחזיר את קבוצת כל התמורות (פרמוטציות) של איברי הרשימה. התשובה המוחזרת תהיה רשימה של תתי רשימות, כאשר כל תת-רשימה מייצגת את אחת התמורות.

לדוגמה:

```
> (my_permutations '(1 2 3))
'( (1 2 3) (2 1 3) (1 3 2) (3 1 2) (2 3 1) (3 2 1))
```

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: 2022ג **מועד הגשה:** **24.7.2022**

- 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)

בשאלה זו עליכם לממש טיפוס נתונים מופשט המייצג מספר בינרי ופעולות שונות על טיפוס נתונים זה.

BinaryNum הוא טיפוס נתונים מופשט (ADT) המייצג מספר בינרי ותומך בפעולות הבאות:

- create- מחזירה מספר בינרי עם סיבית אחת שערכה 0.
- add-bit - מקבלת כפרמטרים מספר בינרי וספרה 0 או 1, ומוסיפה ספרה זו למספר הבינרי כביט חדש מימין. פעולה זו מחזירה את המספר הבינרי החדש.
- change-bit - מקבלת מספר בינרי, אינדקס (החל מ- 1 מימין) ומספר 0 או 1. פעולה זו משנה את הביט במספר הבינרי באינדקס הנתון לערך שהתקבל כפרמטר, ומחזירה את המספר הבינרי החדש.
- to-base10 - מקבלת מספר בינרי ומחזירה את ערכו לפי בסיס 10.
- rotate-right - מקבלת מספר בינרי ומסובבת את סיביותיו ימינה בסיבית אחת, ומחזירה את המספר הבינרי החדש.
- rotate-left - מקבלת מספר בינרי ומסובבת את סיביותיו שמאלה בסיבית אחת, ומחזירה את המספר הבינרי החדש.

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) שלהן.

הקפידו על הגדרה מלאה לכל פעולה.

ג) ממשו את טיפוס הנתונים המופשט BinaryNum עלפי הפעולות שהוגדרו באפיון של טיפוס הנתונים. ניתן לכתוב בנוסף פרוצדורות עזר במקרה הצורך.
אופן המימוש נתון לבחירתכם ויכול להיות ע"י רשימות או פרוצדורלי או ע"י שימוש ב-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:

(- - 3 2 - 4 - 12 7)

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 ?empty-env המקבלת סביבה e ומחזירה תשובה בוליאנית מתאימה, האם הסביבה ריקה או מורחבת?

שימו לב, בשאלה זו הייצוג של סביבה הוא פרוצדורלי.

שאלה זו היא שאלה 2.13 מספר הלימוד

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: 2022ג **מועד הגשה:** **5.8.2022**

- 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)

פתרו את תרגיל 3.9 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: 2022ג **מועד הגשה:** **12.8.2022**

- 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 רקורסיבי/ות. הסבירו מדוע?

ג) כיתבו באמצעות הפתרון של סעיף א', פרוצדורה רקורסיבית המקבלת פרמטר n ומחזירה את האיבר ה-n-י בסדרת פיבונצ'י.

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 מסוג מספר המחזירה תשובה בוליאנית (כמובן בצורת exp-val) של true כאשר אחד המספרים הוא פי-2 מהשני. אחרת יוחזר false.
למשל, עבור המספרים 6 ו- 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 פרמטר n ומחזירה את n!.

על ידי שימוש באופן החישוב הבא:

f(0)=1

f(n) = n*f(n-1)

לשם כך השתמשו בטכניקה מסעיף ב', וכן ב- Currying.

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: 2022ג **מועד הגשה:** **26.8.2022**

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

ברצוננו להרחיב שפה זו עם יכולת של הגדרה של מצביע רב מימדי, וגישה באמצעותו לנתונים.

להלן הדקדוק המגדיר את הביטויים החדשים שיש להטמיע בשפה:

ADDR ::= &

addr-exp ()

Expression ::= {*ADDR*}$^+$ (*Expression*)

multiptr-exp (ptrs data)

STAR ::= *

star-exp ()

Expression ::= {*STAR*}$^+$ *identifier*

star-exp (stars id)

הסבר על מרכיבי הביטויים ואופן פעולתם:

ביטוי **multiptr-exp** מאפשר הגדרת מצביע רב מימדי (מצביע למצביע) על נתון מסוים.

הביטוי מורכב מ-ptrs שהיא רשימת **addr-exp** שאורכה מציין את המימדיות של המצביע.
מרכיב data הוא ביטוי שערך עליו מצביעים הצורה רב מימדית.

ביטוי **multiptr-exp** אמור להחזיר מצביע (reference) למצביע ....למצביע לנתון המיוצג ע"י data.

ביטוי star-exp מאפשר פניה אל הנתון המוצבע ע"י המצביע הרב מימדי ששמו נתון ע"י id או אל אחד המצביעים בדרך אל הנתון. הביטוי מורכב מ-stars שהיא רשימה של star-exp. אורכה של הרשימה מציין את כמות המצביעים שיש להתקדם, הערך המוחזר הוא מה שמאוחסן במקום אליו הגענו.
אם אורכה של הרשימה גדול יותר מהמימד של המצביע תודפס הודעת שגיאה.

להלן דוגמת שימוש:

```
let  p = &&&(6)
in
  let q= **p
  in
     let  num = *q
     in
       -(num, 2)
```

בדוגמה זו מוגדר מוגדר מצביע תלת מימדי p (כלומר, מצביע למצביע למצביע ל- 6).
המצביע q הוא מצביע מימדי המצביע לעבר 6. ו-num הוא 6
לכן ערכו של הביטוי הוא (num-val 4).

ממשו הגדרות אלה בשפת "ויפנה" (EXPLICIT-REFS)

שאלה 2 (50 נקודות)

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

ברצוננו להרחיב שפה זו עם ביטוי חדש בשם seq.

להלן הדקדוק המגדיר את הביטוי החדש שברצוננו להוסיף:

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

seq-exp (id exps)

לביטוי **seq-exp** המרכיבים הבאים:

- **id** – משתנה שכבר הוגדר קודם לכן. (זו תהיה שגיאה לבצע ביטוי זה על משתנה לא קיים)
- **exps** – רשימת ביטויים (לפחות אחד) המופרדים בפסיק.

שלבי חישוב ביטוי seq-exp הם:
למשתנה המצויין ע"י id יבוצעו מספר השמות בצורה סדרתית. ההשמות הם ערכי הביטויים הרשומים ברשימה exps. בתחילה המשתנה יקבל את ערכו של הביטוי הראשון, לאחר מכן את ערכו של הביטוי השני, וכן הלאה. בכל חישוב ביטוי, משתמשים בערכו העדכני של המשתנה id עד כה.
ביטוי זה מחזיר את ערכו של האחרון של המשתנה המצויין ע"י id.

להלן דוגמא:

```
let z= 10
in let y= seq (z [-(z,1), zero? (z), if z then 3 else 6])
   in  y
```

בדוגמה זו, מוגדר משתנה z שערכו 10.
לאחר מכן, מוגדר משתנה y המקבל את ערכו של ביטוי seq.
ביטוי ה-seq משנה בתחילה את ערכו של z להיות 9 (ערכו של הביטוי הראשון), לאחר מכן ערכו של z משתנה לערכו של הביטוי השני, שהוא ביטוי בוליאני, ולכן z משנה את ערכו ל-false ולבסוף, המשתנה z מקבל את ערכו של הביטוי האחרון, במקרה זה 6.
ולכן המשתנה y מקבל ערך של (num-val 6) בהחסן (store) וזה גם ערכה של כל הדוגמה.

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: 2022ג **מועד הגשה:** **2.9.2022**

- 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):

```
let p = proc (w:?) (w 7)
in
  (p proc (x:?) -(x,7))
```

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.