# עקרונות שפות תכנות - תרגיל 2

# טיפוסים, רקורסיות, מימושים עצלים וחרוצים

**תאריך פרסום**: 27.4.2017

**תאריך הגשה**: 10.5.2017

**אופן ההגשה:**

ההגשה (ב-2 החלקים) **בזוגות**, בלימוד בלבד (עד 23:55)!

**בחלק התאוריטי –**  יש להגיש קובץ או מסמך סרוק בלבד

**בחלק התכנותי -** יש להגיש **קובץ** **scheme** **יחיד** ובו כל ההגדרות. שימו לב כי התכניות שלכם נבדקות ע"י סקריפט ולכן חייבות להיות ללא שגיאות תחביר, ולהחזיר **בדיוק** את הפלטים הנדרשים

# חלק תאורטי (50 נקודות)

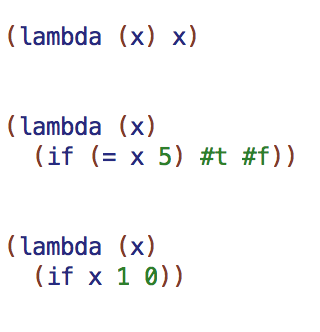
1. **טיפוסים (21 נק')**
   1. (3 נק') הסבירו מדוע יש צורך בטיפוס פולימורפי (משתנה טיפוס).

יש צורך בטיפוס פולימורפי כיוון שקיימות פרוצדורות אשר מקבלות ארגומנטים מסוגים שונים (סימון: T). כלומר, לדוגמא פרוצדורה אחת יכולה לפעול גם על מספרים וגם על boolean.

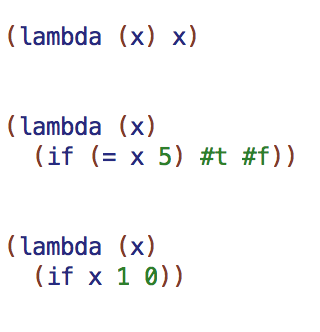
כתבו דוגמא לפונקציה פולימורפית אשר הטיפוס שלה מכיל **שלושה** טיפוס פולימורפיים **שונים**.

(lambda (w x y) (x))

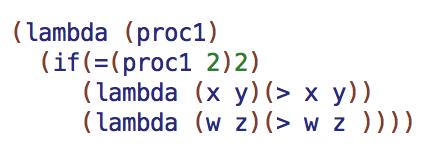
* 1. (2 נק') אילו 2 בנאי טיפוסים למדנו (הכוונה **לסימנים** היוצרים טיפוסים מורכבים מטיפוסים אטומיים)? כתבו את המשמעות של כל אחד מהם.
     + \* : tuple composite types
     + -> : procedure composite types
  2. (8 נק') כתבו בכל תת-סעיף פרוצדורה שהטיפוס שלהם הוא הטיפוס המופיע בתת הסעיף. יש לוודא כי כל הטיפוסים ניתנים להסקה מהביטויים עצמם.
     + [Number->Boolean]



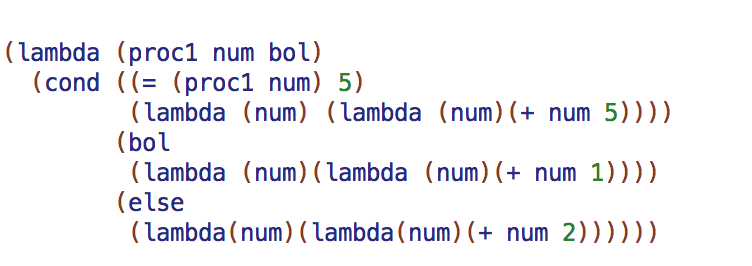
* + - [Boolean->Number]



* + - [ [Number->Number] -> [Number\*Number->Boolean]]



* + - [ [Number->Number] \*Number\*Boolean -> [Number->[Number->Number]]



* 1. (8 נק') מהם הטיפוסים של הביטויים הבאים?

(lambda (x) (if a x x))

[T->T]

(if a (lambda (n) (+ n 1)) (lambda (n) (- n (- n 1))))

[Number->Number]

(lambda (a b)

(+ (a 2) (b (b a))))

[[Number-> Number] \*[[[Number->Number]-> Number] Union [Number -> Number]] -> Number]

(lambda (a b c) (a b c))

[[T1\*T2 -> T[3\*T1\*T2->T3]

1. **חוזים (14 נק')**
   1. ( 2 נק') מהו ההבדל המרכזי בין pre-conditions לבין post-conditions?

בpre-condition התנאי צריך להתקיים לפני הפעלת הפרוצדורה ובpost-condition התנאי צריך להתקיים אחרי הפעלת הפרוצדורה.

איזה מהם קשור יותר לאחריות של מממש הפרוצדורה?

post-condition

* 1. (6 נק') כתבו חוזה מלא לפרוצדורה cylinder\_volume המקבלת רדיוס עיגול וגובה, ומחשבת נפח של גליל

Signature: cylinder\_volume(radius, height)

Purpose: To compute the volume of a cylinder whose radius is the ‘radius’ and height is ‘height)

Type: [Number\*Number -> Number]

Example: (cylinder\_volume 4 8) should produce 401.92

Pre-conditions: radius >= 0, height >= 0

Post-condition: result = PI \* radius2 \* height

Tests: (cylinder\_volume 4 8) 🡺 401.92

Definition:

(define (cylinder\_volume radius height)

(\* 3.14 (\* radius radius) height))

* 1. (6 נק') כתבו חוזה מלא לפרוצדורה make\_apply\_n מתרגיל 1

Signature: (make\_apply\_n f n)

Purpose: Apply a function ‘f’ on number ‘n’ times using predefined procudures twice and thrice. Display a string and print “!” is ‘n’ isn’t 1,2 or 3.

Type:[ [T1->T2]\*Number -> T2]

Example: ((make\_apply\_n add5 3) 2) should produce 17

Preconditions: n is a number, f is a procedure with 1 argument

Post-condition: result= f(x) or (twice f) or (thrice f) or wrong value for n: ‘n’!

Tests: ((make\_apply\_n add5 1) 2) 🡺 7

((make\_apply\_n add5 2) 2) 🡺 12

((make\_apply\_n add5 3) 2) 🡺 17

((make\_apply\_n add5 33) 2) 🡺 “wrong value for n: 33!”

Definition:

(define (make\_apply\_n f n)

(cond ((= n 1) f )

((= n 2) (twice f))

((= n 3) (thrice f))

(else (display "wrong value for parameter n:")(display n) (lambda (stam) (display "!")))))

1. **רקורסיות (10 נק')**
   1. הסבירו מדוע תהליך איטרטיבי ב-Scheme הוא גם רקורסיבי.

בScheme לא ניתן לממש איטרציות (לולאות) ולכן גם תהליך אטרקטיבי הוא רקורסיבי. לולאה דורשת השמות ומצב מה שאי אפשר לדמות בתכנות פונקציונאלי.

* 1. כתבו פרוצדורה **רקורסיבית** (לא-אטרקטיבית) המקבלת n חיובי כקלט ורצה בזמן **קבוע** (לכל n)

(define (func n)

(if (<= n 2) 2 (func 2)))

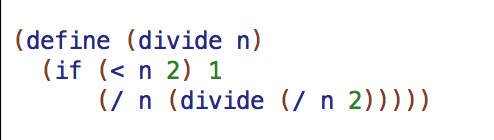
* 1. התבוננו בפרוצדורה f משאלה 1 בחלק התכנותי. כתבו מהי סיבוכיות הזמן והמקום של פרוצדורה זו כפונקציה של n – במימוש רקורסיבי (לא-איטרטיבי) ובמימוש איטרטיבי. **הסבירו** בקצרה.
     + מימוש רקורסיבי (לא איטרטיבי)
       - זמן: מעריכי, בכל frame מתבצעת קראה לפונקציה 3 פעמים(O(3n)).
       - מקום: ליניארי, ברגע נתון נשמר בזיכרון מספר קבוע של מסלולים שלכל היותר באורך n => O(n)
     + מימוש איטרטיבי:
       - זמן: ליניארי, יש צורך בלחשב את כל האיברים עד לאיבר הn.
       - מקום: קבוע, רק ערכי משתני הלולאה נשמרים.
  2. האם אתם מסכימים עם הקביעה שפרוצדורה איטרטיבית רצה תמיד במקום ליניארי?

כן, פרוצדורה איטרטיבית רצה תמיד במקום ליניארי מכיוון שהיא תלויה בגודל הקלט וגודל הקלט לא גדול מn.

ומה לגבי זמן ליניארי?

לא, קיימים פרוצדורות איטרטיביות שרצות בזמן גדול מליניארי לדוגמא: מיון בועות.

* 1. כתבו פרוצדורה רקורסיבית (לא-איטרטיבית) הרצה בזמן לוגריתמי בגודל הקלט. אין להשתמש בפונקציית log או בפונקציה דומה.



1. **מימושים חרוצים ועצלים (8 נק')**
   1. תארו בקצרה 2 מצבים שהם כדאי יותר לבצע מימוש עצל ו-2 מצבים בהם כדאי יותר לבצע מימוש חרוץ

מימוש עצל :

1. במצב של if כדי שלא נצטרך לעשות את שני החלקים שלו.
2. כשרוב העבודה היא בזמן ריצה ולא צריך לתכנן הרבה לפני.

מימוש חרוץ:

1. כאשר עלות זמן הריצה הוא יקר.. נרצה קודם לחשב את כל החלקים ואז להריץ.
2. כאשר פעולות חוזרות על עצמן נרצה לחשב אותם פעם אחת.
   1. האם נעזרים ב-let במימושים חרוצים או עצלים בדרך כלל? מדוע?

במימושים חרוצים, כדי שהחישובים יהיו מוכנים כבר בזמן קומפילציה.

* 1. הסבירו בקצרה, מדוע בדוגמת ה-if-proc שראינו בשיעור חובה לקלוט ב-2 הארגומנטים פרוצדורות. כי בפונקציית if יש שני אפשרויות שתלויות באם התנאי יתקיים או לא. ובשביל כל אפשרות אנחנו צריכים פרוצדורה משלה (לא תמיד אותו דבר).
  2. נתונים 2 ביטויים – hard-exp1 ו-hard-exp2. כתבו קריאה נכונה למימוש העצל של הפרוצדורה- if-proc שראינו בכיתה (שימו לב כי הועלתה גרסא מעודכנת בתאריך 27.4) כדי להחזיר פרוצדורה בשם f-hard של ארגומנט יחיד – condition, שתעריך רק אחד מהביטויים לפי ערכו של הארגומנט. **הנחיה**: עטפו כל אחד מהביטויים ב-lambda ללא ארגומנטים.

הקוד שלכם צריך להתחיל בביטוי הבא:







(define f-hard (lazy-if-proc (lambda() hard-exp1)(lambda() hard-exp2))condition))

# חלק תכנותי (50 נק')

**1. (14 נק')** נתונה הפונקציה הבאה, המוגדרת מעל הטבעיים:

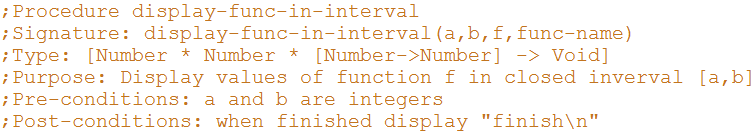


1. ממשו את הפונקציה בעזרת פרוצדורה בשם f היוצרת תהליך רקורסיבי מסוג רקורסיית עץ. כתבו לפרוצדורה חוזה הכולל חתימה, טיפוס, מטרה, תנאי התחלה, תנאי סיום, ובדיקות.
2. ממשו את הפונקציה בעזרת פרוצדורה f-iter היוצרת תהליך איטרטיבי. כתבו חוזה חלקי המכיל רק את טיפוס הפרוצדורה.

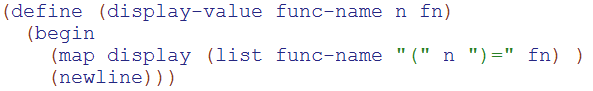
**2. (36 נק')** לעתים קיים צורך, כמו בשאלה הקודמת, להשוות בין ערכים שמחזירות שתי פרוצדורות f ו-g. לשם כך נתן להגדיר אינטרבל [a,b] (נתחיל באינטרבל שלמים) ולבדוק כי מתקיימים כל השיוויונים שלהלן :

שתי דרכים אפשריות לעשות זאת הן (1) הדפסת הערכים שכל אחת מהפרוצדורות מחזירה באינטרבל [a,b] והשוואה ידנית בין זוגות הערכים (2) השוואה של זוגות הערכים בעזרת פרוצדורה שלישית לקבלת תשובה **בינארית** (ערך בוליאני) לגבי הזהות בין הפרוצדורות. (3) השוואה בין זוגות הערכים לקבלת תשובה **נומרית** לגבי **מידת** הזהות בין הפרוצדורות. בשאלה זו נממש את שלוש הדרכים ואף נשתמש בחלק מהפרוצדורות כדי לחשב את ערכו של אינטגרל של פונקציה בשיטת המלבנים.

1. ממשו פרוצדורה **איטרטיבית** בשם display-func-in-interval המקבלת פרוצדורה f ואת קצוות האינטרבל [a,b] ומבצעת **הדפסה** של הערכים המוחזרים על ידי הפרוצדורה כאשר היא מופעלת על כל הערכים השלמים באינטרבל . הפרוצדורה נתונה לפי החוזה הבא:



לצורך הדפסת ערכים בודדים , יש להשתמש בפרוצדורה הבאה:



הערה: לצורך שאלה זו הניחו כי f עצמה היא איטרטיבית

1. ממשו פרוצדורה **איטרטיבית** בשם equal-procedures? המקבלת שתי פרוצדורות f, g (הניחו כי הן איטרטיביות) ו-2 קצוות של אינטרבל שלמים [a,b] מחזירה #t אם ורק אם כל זוגות הערכים ששתי הפרוצדורות מחזירות באינטרבל שווים כלומר מתקיימים כל השיוויונים שבתיאור לעיל.

יש לממש את הפרוצדורה תחת האילוצים הבאים:

1. כאמור, הפונקציה צריכה להיות איטרטיבית (בהנחה כי f ו-g הן כאלה)
2. אין להפעיל אף אחת מהפרוצדורות יותר מפעם אחת כל ערך באינטרבל
3. אין להשוות בין הערכים ו- יותר מפעם אחת לכל
4. אין להשתמש באופרטור and

**כתבו חוזה חלקי המכיל רק את טיפוס הפרוצדורה**

1. ממשו פרוצדורה איטרטיבית בשם average-procedure-difference הפועלת בדומה לפרוצדורה מהסעיף הקודם, אך מחזירה את **ממוצע** ההפרשים בין הערכים שמחזירות הפרוצדורות באינטרבל, המוגדר ע"י הנוסחא:

יש להשתמש ב-letrec להגדרת הפרוצדורה המבצעת את האיטרציות.

**כתבו גם חוזה מפורט לפרוצדורה**. בידקו את הפרוצדורה על פונקציות ליניארית כגון ו- כדי לוודא שהיא מתנהגת כצפוי, והוסיפו את הבדיקות שלכם לחוזה

1. כתבו פרוצדורה בשם procedure-subtraction-floats, המתנהגת בדומה לזו שבסעיף הקודם, אך פועלת באינטרבל כללי (לאו דווקא של מספרים שלמים). הקלטים הם שתי פרוצדורות f,g שני קצוות אינטרבל a,b (שאינם בהכרח שלמים), ו גודל הצעד step (שגם אינו בהכרח שלם). הפרוצדורה תחשב את **סכום** הפרשי הערכים של הפונקציות בנקודות לפי הנוסחא הבאה:

**כתבו חוזה חלקי המכיל רק את טיפוס הפרוצדורה**

1. השתמשו בפרוצדורה מהסעיף הקודם כדי לחשב את סכום ההפרשים כאשר :

כתבו גם את הביטוי המחשב את הערך, וגם את התוצאה שקיבלתם בהערה אחרי קוד הפרוצדורה התחילו את ההערה במילים: answer for part 5 of question 2

1. כתבו פרוצדורה בשם calculate-definite-integral המקבלת כקלט פרוצדורה f, קצוות אינטרבל a ו-b, וגודל הצעד step , ומחשבת **בעזרת הפרוצדורה מהסעיף הקודם** את ערכו של האינטגרל המסויים של f בין a ל-b בשיטת המלבנים, המתקבל מהנוסחא:

ניתן להניח כי f חיובית בכל האינטרבל.

השתמשו ב-test שבו הפונקציה היא , קצוות האינטרבל הם 0 ו-1, ו-step קטן (למשל 0.0001) ובידקו כי הפרוצדורה מחזירה ערך קרוב ל- , ובtest- נוסף דומה שבו הפונקציה היא והתוצאה קרובה ל .

**כתבו חוזה מפורט** והוסיפו את ה-tests לחוזה.

# בהצלחה