



מטה מל"מ
המרכז הישראלי לחינוך מדעי טכנולוגי
על שם עמוס דה-שליט



משרד החינוך
המזכירות הפדגוגית
האגף לתכנון ולפיתוח תכניות לימודים



מגוון – מחקר ופיתוח בהוראת מדעי-המחשב
המחלקה להוראת הטכנולוגיה והמדעים
הטכניון – מכון טכנולוגי לישראל
ומוסד הטכניון למחקר ופיתוח בע"מ

תכנות פונקציונלי

פרדיגמה תכנותית נוספת

ספר שני

פרק תמי"ט: פונקציות רקורסיביות

**החומרים ניתנים להורדה ושימוש בחינם במסגרת בית הספר.
חל איסור על הפצה מסחרית של החומרים.**

פיתוח היחידה:

פרופ' אורי לירון - ראש הפרויקט

תמי לפידות - ראש צוות הפיתוח

דלית לוי

תמר פז

יעוץ אקדמי: ד"ר שאול מרקוביץ

פרק חמישי: פונקציות רקורסיביות

16.....	מעבדה.....
16.....	מאלגוריתם לפונקציה רקורסיבית.....
23.....	רקורסיה לינארית.....
28.....	פונקציות מספריות.....
31.....	מיפוי וסינון.....
34.....	פונקציות רקורסיביות לוגיות.....
35.....	כלים לעיבוד רשימות.....
38.....	עיבוד רשימות לא שטוחות.....
38.....	עיבוד רשימות לא שטוחות.....
42.....	סיכום.....
43.....	מודלים של פונקציות רקורסיביות.....
45.....	דרכים שונות לתאור פונקציה רקורסיבית.....
51.....	תרגילים נוספים: מבני נתונים מופשטים.....

חלק ראשון: מאלגוריתם לפונקציה רקורסיבית

דוגמה ראשונה: סכום האברים ברשימה

נפתח אלגוריתם רקורסיבי לחישוב סכום האברים ברשימת מספרים.

למשל, סכום האברים ברשימה (1 4 3 10 2 5) הוא 25.

א. מהו סכום האברים ברשימה ריקה?

ב. אם נתון שסכום אברי הרשימה ללא האיבר הראשון הוא S

כתוב ביטוי לחישוב הסכום של כל אברי הרשימה (כולל האיבר הראשון).

ג. נרכז את שני חלקי האלגוריתם ביחד

- אם הרשימה ריקה אזי סכום האברים ברשימה שווה ל-
- אחרת, סכום האברים ברשימה (לא ריקה)

שווה לסכום של ושל

ד. כעת נתרגם את האלגוריתם הרקורסיבי לפונקציה ב-Scheme.

תחילה נכתוב את השלד של הפונקציה.

(define (sum-all L)

(cond

[.....]

[else]))

השלם:

שם הפונקציה -

לפונקציה יש פרמטר אחד ושמו -

גוף הפונקציה מורכב מהוראת cond יחידה.

מדוע יש צורך בהוראת תנאי?

מדוע הכנו שתי שורות להוראת התנאי?

ה. כעת נשלים את השלד של הפונקציה.

התנאי הראשון הוא "אם הרשימה ריקה"

תרגם לשפת Scheme -

הערך המוחזר במקרה כזה הוא

אחרת, סכום האברים ברשימה שווה "לסכום של האיבר הראשון ושל סכום האברים ברשימה ללא הראשון".

כיצד נבטא את סכום האברים ברשימה ללא הראשון?

השלימו את הפונקציה sum-all

```
(define (sum-all L)
  (cond
    [( ) ]
    [else (+ ( ) ( ))]))
```

ו. לחצו על כפתור execute ובדקו מה מחזירה הפונקציה sum-all עבור הרשימות הבאות

(1 2 3 4 5) -

(10 20 30 40) -

(7 -5 12 -6 3) -

דוגמה שנייה: סכום האברים הזוגיים ברשימה

נפתח אלגוריתם רקורסיבי לחישוב סכום האברים הזוגיים ברשימת מספרים. למשל, סכום האברים הזוגיים ברשימה (1 4 3 10 2 5) הוא 16.

א. מהו סכום האברים הזוגיים ברשימה ריקה?

ב. אם נתון שסכום אברי הרשימה הזוגיים ללא האיבר הראשון הוא S כתוב ביטוי לחישוב הסכום של כל אברי הרשימה הזוגיים (כולל האיבר הראשון).

ג. נרכז את חלקי האלגוריתם ביחד

- אם הרשימה ריקה אזי סכום האברים הזוגיים ברשימה שווה ל-
- אם ערכו של האיבר הראשון ברשימה הוא זוגי אזי סכום אברי הרשימה הזוגיים שווה לסכום של ושל
- אחרת, סכום האברים הזוגיים ברשימה שווה ל

ד. כעת נתרגם את האלגוריתם הרקורסיבי לפונקציה ב-Scheme. תחילה נכתוב את השלד של הפונקציה.

(define (sum-even L)

```
(cond
  [ ..... ]
  [ ..... ]
  [ else ..... ]))
```

השלם:

שם הפונקציה -

לפונקציה יש פרמטר אחד ושמו -

גוף הפונקציה מורכב מהוראת cond יחידה.

מדוע יש צורך בהוראת תנאי?

מדוע הכנו שלוש שורות להוראת התנאי?

ה. כעת נשלים את השלד של הפונקציה.

התנאי הראשון הוא "אם הרשימה ריקה"

תרגם לשפת Scheme -

הערך המוחזר במקרה כזה הוא

התנאי השני הוא "אם האיבר הראשון הוא מספר זוגי?"

תרגם לשפת Scheme -

הערך המוחזר במקרה כזה הוא הסכום של האיבר הראשון ושל סכום האיברים ברשימה ללא הראשון.

כיצד נבטא את סכום האברים הזוגיים ברשימה ללא הראשון?

תרגם את הסכום המוחזר במקרה השני לשפת Scheme

אחרת, סכום האברים הזוגיים ברשימה שווה "לסכום האברים הזוגיים ברשימה ללא הראשון"

תרגם לשפת Scheme -

השלימו את הפונקציה sum-even

```
(define (sum-even L)
  (cond
    [( ) ]
    [( ) (+ ( ) ( ))]
    [else ( ) ]))
```

ו. לחצו על כפתור execute ובדקו מה מחזירה הפונקציה sum-even עבור הרשימות הבאות

(2 3 4 5 2) -

(10 20 30 40) -

(7 -5 12 -2 3) -

דוגמה שלישית: הוצאת איבר מרשימה

נפתח אלגוריתם רקורסיבי להוצאת איבר כלשהו מרשימה.

למשל, אם נוציא מהרשימה (10 20 30 40) את האיבר השני, נקבל את הרשימה (10 30 40)
אם נוציא מהרשימה (what a nice day) את האיבר השלישי, נקבל את הרשימה (what a day)

א. כתבו פונקציה without-1 שמקבלת רשימה ומחזירה אותה ללא האיבר הראשון.
למשל, בתגובה להוראה ((without-1 '(10 20 30 40))) נקבל את הרשימה (20 30 40)

ב. כתבו פונקציה without-2 שמקבלת רשימה ומחזירה אותה ללא האיבר השני.
למשל, בתגובה להוראה ((without-2 '(10 20 30 40))) נקבל את הרשימה (10 30 40)

ג. לפני שנכתוב את הפונקציה without-3 נבדוק האם ניתן להעזר בפונקציות הקודמות.

נניח שהרשימה המקורית היא (10 20 30 40 50 60)

ונבדוק מה יחזירו הפונקציות הקודמות

$(\text{without-1 } '(10\ 20\ 30\ 40\ 50\ 60)) \Rightarrow (20\ 30\ 40\ 50\ 60)$

$(\text{without-2 } '(10\ 20\ 30\ 40\ 50\ 60)) \Rightarrow (10\ 30\ 40\ 50\ 60)$

כעת אנחנו רוצים לכתוב פונקציה שתוציא מהרשימה את האיבר השלישי שלה.

כלומר,

$(\text{without-3 } '(10\ 20\ 30\ 40\ 50\ 60)) \Rightarrow (10\ 20\ 40\ 50\ 60)$

כיצד נוכל להעזר בפונקציות הקודמות?

במקום להוציא את האיבר השלישי מהרשימה

נוציא את האיבר השני מהרשימה ללא האיבר הראשון ונקבל

```
(without-2 (rest '(10 20 30 40 50 60)))
```

```
⇒ (without-2 '(20 30 40 50 60))
```

```
⇒ (20 40 50 60)
```

כל שנותר לנו הוא להחזיר את האיבר הראשון. כלומר,

```
(cons 10 [without-2 (rest '(10 20 30 40 50 60))])
```

```
⇒ (cons 10 '(20 40 50 60))
```

```
⇒ (10 20 40 50 60)
```

במילים אחרות, כדי להוציא מרשימה את האיבר השלישי שלה, אפשר להחזיר את האיבר הראשון לרשימה המתקבלת מהוצאת האיבר השני מהרשימה ללא האיבר הראשון.

הפונקציה בשלמותה תראה כך -

```
(define (without-3 L)
```

```
(cons (first L) [without-2 (rest L)]))
```

הגדירו את הפונקציה על המחשב ובידקו אותה.

ד. באופן דומה, השלימו את הפונקציה שמוציאה מהרשימה את האיבר הרביעי שלה.

```
(define (without-4 L)
```

```
(cons (first L) [ _____ ]))
```

הגדירו את הפונקציה על המחשב ובידקו אותה.

ה. נתבונן כעת בשתי הפונקציות האחרונות:

```
(define (without-3 L)
  (cons (first L) [without-2 (rest L)]))
```

```
(define (without-4 L)
  (cons (first L) [without-3 (rest L)]))
```

נסחו במילים שלכם מה משותף לשתי הפונקציות האחרונות.

ו. נכתוב את הפונקציה הכללית `without` שמוציאה איבר כלשהו (במקום n) מהרשימה `L`.

```
(define (without n L)
  (cond
    [(= n 1) (rest L)]
    [else (cons (first L) {without (sub1 n) (rest L)})]))
```

הסבירו מדוע התנאי הראשון הוא $(= n 1)$

הסבירו מדוע במקרה הראשון מוחזרת שארית הרשימה `(rest L)`

הגדירו את הפונקציה `without` על המחשב ובידקו אותה.

חלק שני: רקורסיה לינארית

תרגיל 1

כתבו פונקציה שמקבלת רשימה של מספרים ומחזירה את מספר האברים שערכם אי-זוגי ברשימה. למשל, בתגובה להוראה (count-odd '(4 60 3 11 2 9 10 1 48)) יוחזר 4.

תרגיל 2

א. כתבו פונקציה שמקבלת רשימה ומחזירה את המספר הראשון שמופיע ברשימה (ידוע שיש לפחות אחד). למשל, בתגובה להוראה (first-number '(a b 15 c d 93 e f)) יוחזר 15

ב. כתבו פונקציה שמקבלת רשימה ומחזירה את המספר האחרון שמופיע ברשימה. למשל, בתגובה להוראה (last-number '(a b 15 c d 93 e f)) יוחזר 93
 רמז: אפשר להעזר בפונקציה מהסעיף הקודם ולכתוב פונקציה לא רקורסיבית.

תרגיל 3

כתבו פונקציה שמקבלת רשימה ואיבר item ומחזירה את המיקום (הראשון) של item בתוך הרשימה. למשל, בתגובה להוראה (place 'e '(a b c d e f)) יוחזר 5

תרגיל 4

כתבו פונקציה שמקבלת רשימת מספרים ומחזירה את המספר הגדול ביותר שמופיע ברשימה (מקסימום).

תרגיל 5

הפונקציה הבאה מקבלת רשימת מספרים. בדקו מה היא מחזירה ותנו לה שם משמעותי.

```
(define (sod L)
```

```
  (cond
```

```
    [(empty? L) '()]
```

```
    [else (cons (* 2 (first L)) (sod (rest L)))]))
```

תרגיל 6

א. כתבו פונקציה רקורסיבית שמקבלת רשימה של מספרים חיוביים ומחזירה רשימה של שורשי המספרים. (תזכורת: הפונקציה sqrt מקבלת מספר ומחזירה את השורש הריבועי שלו).

למשל, עבור הרשימה (1 25 0 64 100) תוחזר הרשימה (1 5 0 8 10).

ב. כתבו פונקציה רקורסיבית שמקבלת רשימה של רשימות ומחזירה רשימה של איבריהם הראשונים. למשל, עבור ((dad and mom) (good morning) (my cat)) תוחזר הרשימה (my good dad).

הפונקציות בתרגיל 6 מקבלות רשימה (רשימת קלט) ומחזירות רשימה (רשימת פלט).

לכל איבר ברשימת הקלט מותאם איבר ברשימת הפלט. התאמה כזו נקראת **פונקצית מיפוי**.

תרגיל 7

כתבו פונקציות רקורסיביות המבצעות את המיפויים הבאים:

$$1. ((a\ b\ c)\ (1\ 2)\ (3\ 4\ 5)) \Rightarrow ((c\ b\ a)\ (2\ 1)\ (5\ 4\ 3))$$

$$2. ((a\ b\ c)\ (x\ y)\ (10\ 15\ 20\ 25)) \Rightarrow (3\ 2\ 4)$$

תרגיל 8

הפונקציה הבאה מקבלת רשימת מספרים.

```
(define (sod L)
  (cond
    [(empty? L) '()]
    [(negative? (first L)) (sod (rest L))]
    [else (cons (sqrt (first L)) (sod (rest L)))]))
```

בדקו מה מחזירה הפונקציה עבור הרשימות הבאות

(64 -4 100 25 -25) -

(1 49 -9 -16 81 4) -

תנו לפונקציה שם משמעותי -

תרגיל 9

א. כתבו פונקציה שמקבלת רשימה כלשהי ומחזירה רשימה רק של המספרים. למשל:

(only-numbers '(1 a (one two three) 2.5 hello -3 97)) ⇒ (1 2.5 -3 97)

ב. כתבו פונקציה שמקבלת רשימה כלשהי ובונה רשימה חדשה ללא הרשימות שבה.

הפונקציות בתרגילים 8-9 מקבלות רשימה (רשימת קלט) ומחזירות רשימה (רשימת פלט).
לכל איבר ברשימת הקלט שמקיים תנאי מסוים,
מותאם איבר ברשימת הפלט. התאמה כזו נקראת **פונקצית סינון**.

תרגיל 10

פונקציה מהסוג **ללא הראשון** היא מקרה פרטי של פונקצית סינון. הפונקציה מחזירה את הרשימה ללא האיבר הראשון ברשימה שמקיים תנאי כלשהו. כתבו פונקציה שמקבלת רשימה ומחזירה את הרשימה ללא האיבר הראשון שהוא מספר זוגי.

תרגיל 11

פונקציה דומה לפונקציה מסוג **ללא הראשון** היא פונקציה מסוג **החלף ראשון** שמחזירה רשימה שבה מוחלף האיבר הראשון המקיים תנאי מסוים בנתון כלשהו. למשל, כתבו פונקציה שמחליפה את המספר הראשון ברשימה במילה number .

תרגיל 12

א. כתבו פונקציה שמקבלת שני מספרים ורשימה ריקה. הפונקציה תחזיר את הרשימה כשהיא מכילה את כל המספרים שבין המספר הקטן והמספר הגדול. לדוגמה, $(3\ 4\ 5\ 6)$ \Leftrightarrow $(\)$ (between 3 6)

ב. כתבו פונקציה שמקבלת שני מספרים, ומחזירה רשימה שמכילה את כל המספרים שבין המספר הקטן והמספר הגדול. לדוגמה, $(3\ 4\ 5\ 6)$ \Leftrightarrow $(\)$ (between 3 6)

חלק שלישי: פונקציות מספריות

תרגיל 13

א. בדקו מה מחזירה הפונקציה הבאה עבור (f 4) .

(define (f n)

```
(cond
  [(= n 1) 1]
  [else (+ (f (- n 1)) (* 2 n))]))
```

$$f(1) = 1$$

$$f(n) = f(n-1) + 2*n$$

ב. שנו את הפונקציה המקורית ארבע פעמים לפי העמודה הימנית והשלימו את הטבלה הבאה:

תאור רקורסיבי (נוסחה)	השינוי בפונקציה	(f 4) לאחר השינוי
$f(1) = 2$ $f(n) = f(n-1) + 2*n$		
$f(0) = 3$ $f(n) = f(n-1) + 2*n$		
$f(1) = 1$ $f(n) = f(n-1) + n/3$		
$f(0) = 1$ $f(1) = 1$ $f(n) = f(n-2) + 2*n$		

תרגיל 14

א. בדקו מה מחזירה הפונקציה הבאה עבור (nice 3 2)

(define (nice a b)

```
(cond
  [(= a 0) 1]
  [else (* 2 (nice (- a 1) (+ b 1)))]))
```

נסחו במילים שלכם מה מבצעת הפונקציה.

ב. אם נשנה את תנאי העצירה כך $(\text{cond } [(= a 0) 5])$ כיצד ישפיע השינוי הזה על הפונקציה? בדקו את השערתכם.

ג. אם נשנה את תנאי העצירה כך $(\text{cond } [(< a 1) 1])$ כיצד ישפיע השינוי הזה על הפונקציה? בדקו את השערתכם.

ד. אם נשנה את תנאי העצירה כך $(\text{cond } [(< a 1) b])$ כיצד ישפיע השינוי הזה על הפונקציה? בדקו את השערתכם.

ה. אם נשנה את תנאי העצירה כך $(\text{cond } [(> b 1) 1])$ כיצד ישפיע השינוי הזה על הפונקציה? בדקו את השערתכם.

ו. מה צריך להיות תנאי העצירה כך שעבור b שלילי, הפונקציה תחזיר $5 \cdot 2^{|b|}$? הריצו את הפונקציה המעודכנת כדי לבדוק. למשל, (nice 50 -5) צריכה להחזיר את המספר 160.

ז. נשנה את תנאי העצירה כך $(\text{cond } [(\text{or } (< a 1) (> b 1)) 1])$ אילו ערכים של a ו- b יגרמו עכשיו לעצירת הפונקציה? מה מבצעת הפונקציה כעת?

ח. הטבלה הבאה מסכמת את השינויים בתנאי העצירה של הפונקציה nice. השלימו אותה.

תנאי העצירה	תחום ערכי a	תחום ערכי b	מה מחזירה הפונקציה
$((= a 0) 1)$	a מספר שלם $a \geq 0$	כל המספרים (הפונקציה כלל לא משתמשת ב-2)	
$((= a 0) 5)$			$5 \cdot 2^a$
$((< a 1) 1)$			
$((< a 1) b)$			
$((> b 1) 1)$		b מספר שלם	$b > 1$ 1 $b = 1$ 2 $b < 1$ $2^{ b +2}$
$((> b -1) 5)$			
$((\text{or } (< a 1) (> b 1)) 1)$			

חלק רביעי: מיפוי (MAP) וסינון (FILTER)

תרגיל 15

כתבו פונקציה שמקבלת רשימה של מספרים ומחזירה רשימה של מכפלתם ב-2.
 לדוגמה, בתגובה להוראה (double '(2 1.5 18 6)) הפונקציה תחזיר (4 3 36 12)

תרגיל 16

כתבו פונקציה שמקבלת רשימה של רשימות ומחזירה את כל הרשימות שאורכן גדול מ-3.
 לדוגמה, בתגובה להוראה (big3 '((x y) (a b c d e) (1 2 3 4)))
 הפונקציה תחזיר את הרשימה ((a b c d e) (1 2 3 4))

תרגיל 17

כתבו פונקציה שמקבלת רשימת מספרים ומחזירה את רשימת הריבועים של כל המספרים הגדולים מ-8.
 לדוגמה, בתגובה להוראה (sq-big8 '(12 5 8 10)) הפונקציה תחזיר את הרשימה (144 100)
 תזכורת: הביטוי (expt x 2) מחזיר את x^2

תרגיל 18

כתבו פונקציה שמקבלת שתי רשימות של מספרים, ומחזירה רשימה של מכפלות האיברים התואמים. לדוגמה, בתגובה להוראה $((3\ 2\ 10\ 7))$ ' (mult ' $(2\ 5\ 8\ 10)$ תוחזר הרשימה $(6\ 10\ 80\ 70)$ שימו לב: חשוב שהרשימות יהיו באותו אורך

תרגיל 19

כתבו פונקציה שמקבלת שלוש רשימות של מספרים, ומחזירה רשימה של ממוצעי האיברים התואמים. לדוגמה, בתגובה להוראה $((4\ 0\ 8\ -2))$ ' (ave ' $(1\ 2\ 3\ 4)$ ' $(10\ 10\ 10\ 10)$ הפונקציה תחזיר את הרשימה $(5\ 4\ 7\ 4)$

תרגיל 20

כתבו פונקציה שמקבלת שתי רשימות של ציונים ומשווה בין ציונים במקומות התואמים. הפונקציה תחזיר רשימה שמורכבת מהמילים better worse equal בהתאם לתוצאות השוואה. לדוגמה, בתגובה להוראה $((75\ 59\ 100\ 60))$ ' (compare ' $(100\ 59\ 80\ 60)$ הפונקציה תחזיר את הרשימה $(better\ equal\ worse\ equal)$ שימו לב: חשוב שהרשימות יהיו באותו אורך

תרגיל 21

המטרה בתרגיל זה היא לבנות פונקצית סינון כללית - **filter**, שהשלד שלה מופיע בהמשך. כדי לסייע בבניית הפונקציה הכללית נביא שתי פונקציות:

- פונקציה filter1 מקבלת רשימה כלשהי ומחזירה רק את המספרים.

```
(define (filter1 L)
  (cond
    [(empty? L) '()]
    [(number? (first L)) (cons (first L) (filter1 (rest L)))]
    [else (filter1 (rest L))]))
```

- פונקציה filter2 מקבלת רשימה כלשהי ומחזירה רק את הרשימות שאורכן בדיוק 2.

```
(define (filter2 L)
  (cond
    [(empty? L) '()]
    [(length2? (first L)) (cons (first L) (filter2 (rest L)))]
    [else (filter2 (rest L))]))
```

```
(define (length2? L)
  (equal? (length L) 2))
```

א. השלימו את הגדרת פונקצית הסינון הכללית.

```
(define (filter ..... L)
  (cond
    [(empty? L) '()]
    [( ..... (first L)) (cons (first L) (filter ..... ))]
    [else ( ..... )]))
```

ב. הריצו במחשב כדי לבדוק, למשל כך - (filter number? '(a 3 b 4 500))

חלק חמישי: פונקציות רקורסיביות לוגיות

תרגיל 22

א. בדקו מה מחזירה הפונקציה הבאה ותנו לה שם משמעותי

```
(define (sod L)
  (cond
    [(empty? L) #t]
    [(not (number? (first L))) #f]
    [else (sod (rest L))]))
```

ב. כתבו פונקציה בוליאנית שמקבלת רשימת מספרים ובודקת האם כולם זוגיים. לדוגמה,

(all-even? '(2 88 46 -10)) ⇒ #t , (all-even? '(2 88 43 -10)) ⇒ #f

ג. כתבו פונקציה בוליאנית שמקבלת רשימה ובודקת האם זו רשימה שטוחה. לדוגמה,

(flat? '(2 shalom 47.4 -10)) ⇒ #t , (flat? '(2 (good day) 47.4 -10)) ⇒ #f

תזכורת: רשימה שטוחה היא רשימה שאף אחד מאיבריה אינו רשימה.

ד. כתבו פונקציה בוליאנית שמקבלת רשימה ובודקת האם יש בה לפחות מספר אחד. לדוגמה,

(has-number? '(this is 1 example)) ⇒ #t , (has-number? '(no numbers here)) ⇒ #f

חלק שישי: כלים לעיבוד רשימות

תרגיל 23

כלי שימושי לעיבוד רשימות הוא הפונקציה upto ("עד ל-").

הפונקציה מקבלת רשימה L ואינדקס n ומחזירה את n האיברים הראשונים ברשימה (עד לאיבר ה-n, כולל איבר זה). לדוגמה,

$(\text{upto } ('(\text{alon gila roni miri moti yael}) 3)) \Rightarrow (\text{alon gila roni})$

$(\text{upto } ('(10 (2.5 3.5) \text{ x hello}) 2)) \Rightarrow ((10 (2.5 3.5)))$

כתבו את הפונקציה.

תרגיל 24

כלי שימושי נוסף הוא הפונקציה from ("מ-").

הפונקציה מקבלת רשימה L ואינדקס n ומחזירה את איברי הרשימה מהמקום ה-n ואילך. לדוגמה,

$(\text{from } ('(\text{alon gila roni miri moti yael}) 3)) \Rightarrow (\text{roni miri moti yael})$

$(\text{from } ('(10 (2.5 3.5) \text{ x hello}) 2)) \Rightarrow ((2.5 3.5) \text{ x hello})$

כתבו את הפונקציה.

תרגיל 25

א. השתמשו בכלים from, upto כדי להגדיר פונקציה חדשה - (between L a b), אשר מחזירה את איברי הרשימה L מהמקום ה-a עד המקום ה-b. דוגמאות:

(between '(alon gila roni miri moti yael) 3 5) \Rightarrow (roni miri moti)

(between '(10 (2.5 3.5) x hello) 2 3) \Rightarrow ((2.5 3.5) x)

שימו לב: הפונקציה between איננה רקורסיבית.

ב. השתמשו בכלים from, upto כדי להגדיר פונקציה חדשה - (insert L x n), אשר מכניסה את האיבר x למקום ה-n ברשימה L. דוגמאות:

(insert '(alon dalya roni miri) 'gadi 3) \Rightarrow (alon dalya gadi roni miri)

(insert '(10 (2.5 3.5) x hello) '(a b c) 2) \Rightarrow (10 (a b c) (2.5 3.5) x hello)

ג. הגדירו את הפונקציה insert כפונקציה רקורסיבית, ללא שימוש בכלים from, upto.

תרגיל 26

כתבו פונקציה לעדכון רשימה. הפונקציה מקבלת רשימה, איבר לעדכון ואיבר חדש שצריך לבוא במקומו, ומחזירה את הרשימה המעודכנת. ניתן להגדיר את הפונקציה באופן רקורסיבי או באמצעות כלים לעיבוד רשימות. דוגמאות:

(update '(alon gila roni miri) 'gila 'sofi) \Rightarrow (alon sofi roni miri)

(update '(a (x y z) b 3) 3 'c) \Rightarrow (a (x y z) b c)

(update '(a b a 3) 'a 'c) \Rightarrow (c b a 3)

שימו לב - בדוגמה האחרונה האיבר a מופיע פעמיים ברשימה, אבל הוא מוחלף ב-c רק בפעם הראשונה.

תרגיל 27

הגדירו פונקציה חדשה butn שמקבלת רשימה L ואינדקס n ומחזירה את הרשימה ללא האיבר המופיע ברשימה במקום ה-n. הוסיפו לפונקציה בדיקה שהרשימה מכילה לפחות n איברים. לדוגמה,

(butn '(alon gila roni miri) 2) \Rightarrow (alon roni miri)

תרגיל 28

הפונקציות list, append הן פונקציות נוספות לבנית רשימות. בדקו את הפונקציות הללו וסכמו את ההבדלים בין שלוש הפונקציות list, append ו-cons. למשל,

(cons '(hello friend) '(15 20 25))

(list '(hello friend) '(15 20 25))

(append '(hello friend) '(15 20 25))

חלק שביעי: עיבוד רשימות לא שטוחות

תרגיל 29

לפניכם שתי פונקציות שמקבלות רשימות לא שטוחות של מספרים.

```
(define (sum-all L)
  (cond
    [(empty? L) 0]
    [(list? (first L))
     (+ (sum-all (first L)) (sum-all (rest L)))]
    [else (+ (first L) (sum-all (rest L)))]))
```

```
(define (count-all-even L)
  (cond
    [(empty? L) 0]
    [(list? (first L))
     (+ (count-all-even (first L)) (count-all-even (rest L)))]
    [(even? (first L)) (add1 (count-all-even (rest L)))]
    [else (count-all-even (rest L))]))
```

הגדירו אותן על המחשב ובדקו את ההוראות הבאות:

```
(sum-all '(1 (2 20) (3 30 300)))
(sum-all '((1 2 3) (4 5 6 7) (8 9 10)))
(count-all-even '(1 (2 20) (3 30 300)))
(count-all-even '((1 2 3) (4 5 6 7) (8 9 10)))
```

הסבירו מה מבצעות שתי הפונקציות:

תרגיל 30

א. כתבו פונקציה שמקבלת רשימה לא שטוחה ומחזירה את סכום המספרים האי-זוגיים המופיעים ברשימה או בתת-אבריה. לדוגמה $(10\ 11\ 12)$ $(9\ (2\ 3\ 4\ 5\ 6))$ $(1\ (2\ 3\ 4\ 5\ 6))$ (sum-all-odd תחזיר 29 (כי $1+3+5+9+11$))

ב. כתבו פונקציה שמקבלת רשימה לא שטוחה ומכפילה את כל מספריה פי 2. לדוגמה,
 $(16\ 8\ 4\ 2)$ $(30\ 14\ (20\ 40\ 60))$ $(6\ (20\ 40\ 60)) \Rightarrow (3\ (10\ 20\ 30)\ 15\ 7\ (8\ 4\ 2\ 1))$ (double-all

תרגיל 31

א. כתבו פונקציה שמוחקת מרשימה לא שטוחה את כל ההופעות של איבר מסוים x .
לדוגמה, בעקבות ההוראה

(without-all '(3 (1 2 3 4) (15 7 8) 20 (-5 3 15) 66) 3)

נקבל את הרשימה הבאה שנמחקו ממנה כל ההופעות של המספר 3

((1 2 4) (15 7 8) 20 (-5 15) 66)

ב. כתבו פונקציה שמקבלת רשימה לא שטוחה של מספרים חיוביים ומחזירה את המספר הגדול ביותר המופיע ברשימה או בתת-אבריה. לדוגמה,

(max-all '((1 2 3) (90 9 19) 89 63 (5 23 1 -5 7))) \Rightarrow 90

ג. כתבו פונקציה שמקבלת רשימה לא שטוחה ומבצעת היפוך של כל אבריה. לדוגמה,

(reverse-all '(100 (1 2 3 4) (5 6 7) 55)) \Rightarrow (55 (7 6 5) (4 3 2 1) 100)

תרגיל 32

א. בדקו מה מחזירה הפונקציה הבאה ותנו לה שם משמעותי

```
(define (sod L)
  (cond
    [(empty? L) #t]
    [(list? (first L)) (and {sod (first L)} {sod (rest L)})]
    [(not (number? (first L))) #f]
    [else (sod (rest L))]))
```

ב. כתבו פונקציה בוליאנית שמקבלת רשימה לא שטוחה ובודקת האם כל המספרים המופיעים בה הם זוגיים. לדוגמה,

```
(all-even? '(2 (12 44 900) (88 46) -10 62)) ⇒ #t
(all-even? '(2 (12 44 9) (88 43) -10 62)) ⇒ #f
```

דף סיכום: פונקציות רקורסיביות

פונקציה רקורסיבית היא פונקציה שקוראת לעצמה. או ליתר דיוק, גורמת להפעלה חוזרת של אותה הפונקציה אבל בדרך-כלל עם ערכים שונים של פרמטרים אקטואליים.

מרכיבים בסיסיים בפונקציה רקורסיבית:

- תנאי עצירה (מקרה בסיסי) שפועל על אחד מהפרמטרים של הפונקציה
- תנאי העצירה מגדיר מה מחזירה הפונקציה במקרה הפשוט ביותר (המקרה הבסיסי).
- קריאה רקורסיבית או צעד רקורסיבי שגורם להפעלה החוזרת.
- שינוי הפרמטרים בקריאה הרקורסיבית צריך להיות "בכיוון" של תנאי העצירה אם אין שינוי בפרמטר המתאים, הפונקציה (התהליך) לא תעצר.

סוגי רקורסיה:

- רקורסית קצה (tail recursion) - יוצרת תהליך חישובי איטרטיבי (לולאה).
- רקורסיה אמיתית (מלאה) - יוצרת תהליך רקורסיבי מלא.
- רקורסיה לינארית (חד-ממדית) מכילה קריאה רקורסיבית יחידה.
- רקורסיה מורכבת (רב-ממדית) מכילה יותר מקריאה רקורסיבית אחת.

הצגנו מספר תבניות לפונקציות רקורסיביות. בעמודים 43-44 תוכלו למצוא דוגמאות של התבניות השונות.

רקורסיה ניתן לתאר במספר דרכים שכל אחת מהן מדגישה היבט אחר. בעמודים 45-50 תוכלו למצוא שתי דוגמאות לתיאורים שונים של פונקציות רקורסיביות.

מודלים של פונקציות רקורסיביות

```
(define (sum-all L)
```

```
  (cond
```

```
    [(empty? L) 0]
```

```
    [else (+ (first L) (sum-all (rest L)))]))
```

```
(define (sum-even L)
```

```
  (cond
```

```
    [(empty? L) 0]
```

```
    [(even? (first L)) (+ (first L) (sum-even (rest L)))]
```

```
    [else (sum-even (rest L))]))
```

```
(define (double L)
```

```
  (cond
```

```
    [(empty? L) '()]
```

```
    [else (cons (* 2 (first L)) (double (rest L)))]))
```

```
(define (roots L)
```

```
  (cond
```

```
    [(empty? L) '()]
```

```
    [(negative? (first L)) (roots (rest L))]
```

```
    [else (cons (sqrt (first L)) (roots (rest L)))]))
```

```
(define (all-numbers? L)
  (cond
    [(empty? L) #t ]
    [(not (number? (first L))) #f ]
    [else (all-numbers? (rest L))]))
```

```
(define (has-number? L)
  (cond
    [(empty? L) #f ]
    [(number? (first L)) #t ]
    [else (has-number? (rest L))]))
```

```
(define (count-atoms-deep L)
  (cond
    [(empty? L) 0 ]
    [(list? (first L)) (+ (count-atoms-deep (first L))
                          (count-atoms-deep (rest L)))]
    [else (+ (first L) (count-atoms-deep (rest L)))]))
```

```
(define (count-even-deep L)
  (cond
    [(empty? L) 0 ]
    [(even? (first L)) (add1 (count-even-deep (rest L)))]
    [(list? (first L)) (+ (count-even-deep (first L))
                          (count-even-deep (rest L)))]
    [else (count-even-deep (rest L))]))
```

דרכים שונות לתיאור פונקציה רקורסיבית - דוגמה ראשונה

נתבונן בפונקציה fact שמקבלת מספר n ומחזירה את העצרת שלו. לדוגמה:

$$(\text{fact } 3) \Rightarrow 1 * 2 * 3 = 6$$

הגדרת הפונקציה

```
(define (fact n)
  (cond
    [(= n 0) 1]
    [else (* n (fact (sub1 n)))]))
```

תאור מילולי של הפונקציה עצרת n!

הערך של עצרת עבור מספר טבעי כלשהו n הוא:

אם n שווה ל-0 הערך הוא 1.

אחרת הערך הוא המכפלה של n בערך של עצרת עבור n-1.

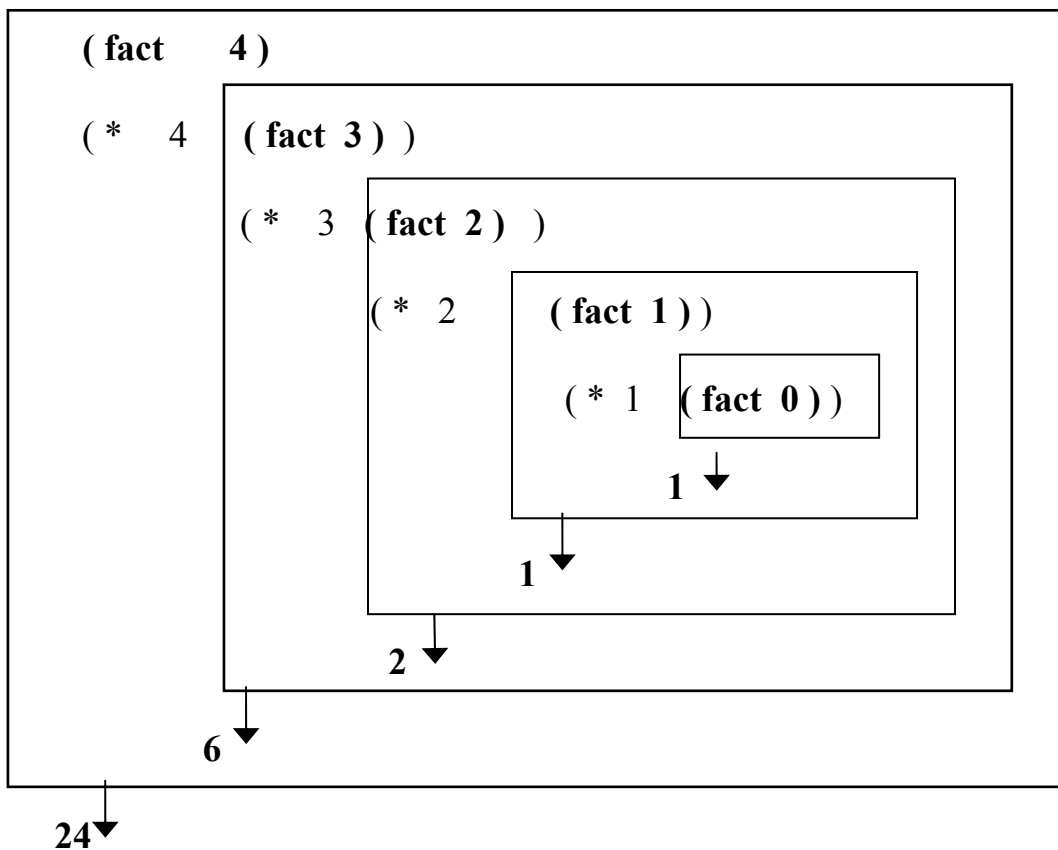
מודל ההצבה

```

( fact 4 ) =
( * 4 ( fact 3 ) ) =
( * 4 ( * 3 ( fact 2 ) ) ) =
( * 4 ( * 3 ( * 2 ( fact 1 ) ) ) ) =
( * 4 ( * 3 ( * 2 ( * 1 ( fact 0 ) ) ) ) ) =
( * 4 ( * 3 ( * 2 ( * 1 1 ) ) ) ) =
( * 4 ( * 3 ( * 2 1 ) ) ) =
( * 4 ( * 3 2 ) ) =
( * 4 6 ) =
24

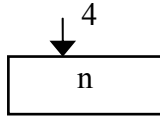
```

מודל המסגרות (גישת Top-Down)



מודל האנשים הקטנים

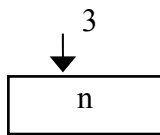
נבקש מפנינה לחשב עבורנו את הפונקציה (fact 4)



לפנינה כיס בשם n .

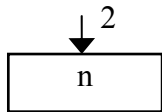
פנינה קוראת את הגדרת הפונקציה. היא מתבוננת בכיסה ומבינה שהתנאי לא מתקיים (n לא שווה לאפס). לכן היא עוברת לחלק ה- else וניגשת לביצוע $(n \text{ (fact (sub1 n))})$. עליה לחשב למעשה את הביטוי (3 (fact 3)) . היא קוראת לפנחס לבצע עבורה את (fact 3).

לפנחס כיס אחד.

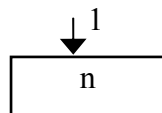


הוא קורא את הגדרת הפונקציה, מתבונן בכיסו ומגלה שהתנאי לא מתקיים. לכן הוא עובר לשורת ה- else ומבין שעליו לבצע $(n \text{ (fact (sub1 n))})$, או על פי הערך שנמצא בכיסו - (3 (fact 2)) . הוא מבקש מפרח לבצע את (fact 2).

גם פרח צריכה לבצע את הביטוי $(n \text{ (fact (sub1 n))})$ שמופיע בשורת ה- else. היא מבקשת מפרי שיבצע עבורה את (fact 1).



פרי בודק שהערך בכיסו (1) אינו אפס, ומבצע את הביטוי בשורת ה- else. לשם כך הוא נעזר בפאולה, שמתבקשת לבצע עבורו את (fact 0).



פאולה בודקת את התנאי ומגלה כי המספר שבכיסה מקיים את התנאי. לפיכך היא מחזירה לפרי את המספר 1.

פרי משתמש בערך שפאולה החזירה, מציב 1 במקום (fact 0) ומבצע (1 (fact 1)) . כעת הוא יכול להחזיר לפרח שקראה לו לעזרתה את התוצאה 1.

פרח מציבה את הערך 1 במקום הפונקציה שביקשה מפרי לבצע, היא מחשבת את הביטוי (2 (fact 1)) ומחזירה לפנחס את המספר 2.

פנחס מציב 2 במקום (fact 2) שביקש מפרח לבצע.

הוא יכול עתה לסיים את חישוב הביטוי (3 (fact 2)) ומחזיר לפנינה את המספר 6.

פנינה מציבה את המספר 6 במקום הפונקציה (fact 3) שביקשה מפנחס לבצע.

היא מחשבת את הביטוי (6 (fact 4)) ומחזירה את המספר 24.

דרכים שונות לתיאור פונקציה רקורסיבית - דוגמה שנייה

נתבונן בפונקציה count-atom שמקבלת רשימה L ומחזירה את מספר האטומים שבה. לדוגמה:

(count-atom '(alon gila (roni miri) moti (yael 16))) \Rightarrow 3

(count-atom '((10 100) (3 9) squares)) \Rightarrow 1

(count-atom '(1948 1977 1999 2000)) \Rightarrow 4

(count-atom '((days and weeks))) \Rightarrow 0

הגדרת הפונקציה

```
(define (count-atom L)
  (cond
    [(empty? L) 0]
    [(atom? (first L)) (+ 1 (count-atom (rest L)))]
    [else (count-atom (rest L))]))
```

תאור מילולי של הפונקציה

מספר-האטומים ברשימה ריקה L הוא 0.

מספר-האטומים ברשימה כלשהי L הוא:

אם האיבר הראשון של L הוא אטום

אז **מספר-האטומים** ב-L יהיה שווה ל**מספר-האטומים** ברשימת השארית (ללא הראשון) ועוד 1

אחרת **מספר-האטומים** ב-L שווה ל**מספר-האטומים** ברשימת השארית (ללא הראשון)

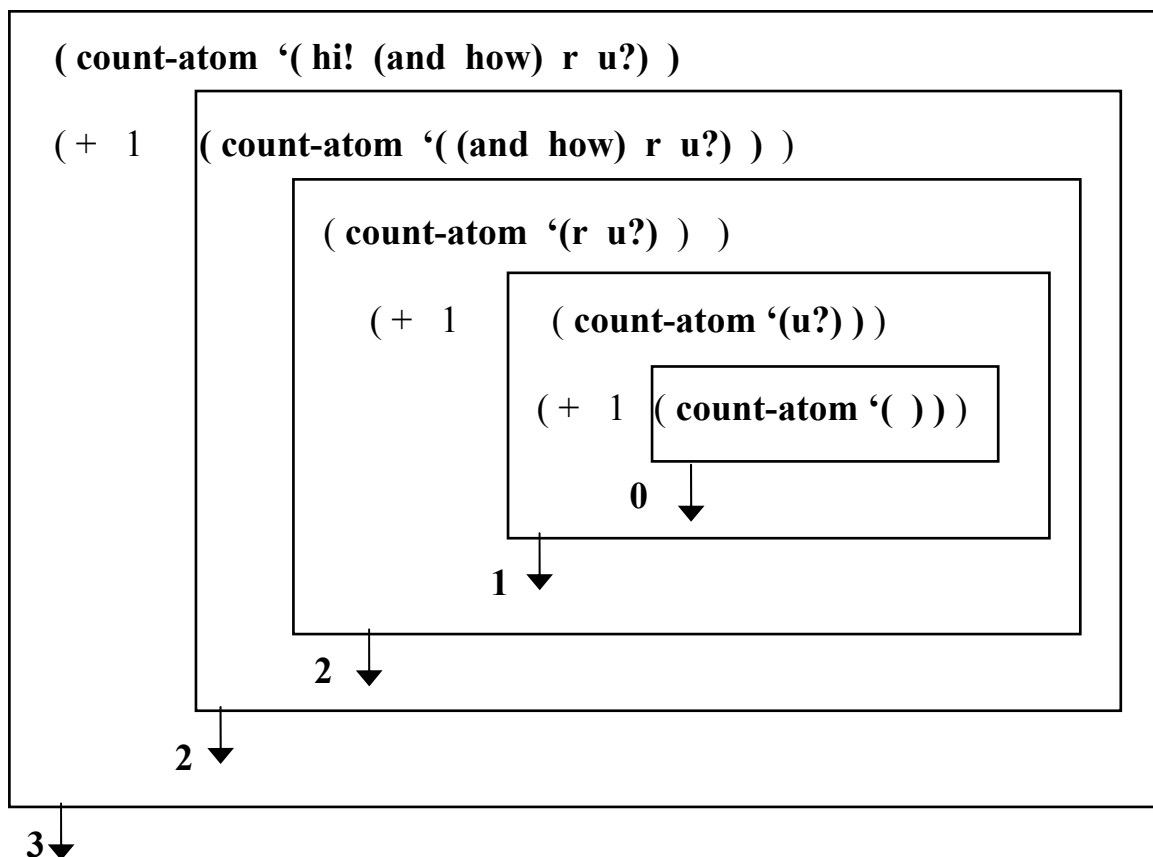
מודל ההצבה

```

(count-atom '(hi! (and how) r u?)) =
(+ 1 (count-atom '((and how) r u?))) =
(+ 1 (count-atom '(r u?))) =
(+ 1 (+ 1 (count-atom '(u?)))) =
(+ 1 (+ 1 (+ 1 (count-atom '())))) =
(+ 1 (+ 1 (+ 1 0))) =
(+ 1 (+ 1 1)) =
(+ 1 2) =
3

```

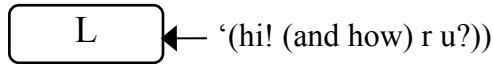
מודל המסגרות (גישת Top-Down)



מודל האנשים הקטנים

נבקש מקרן לחשב עבורנו את הפונקציה `(count-atom '(hi! (and how) r u?))`

לקרן כיס בשם L.

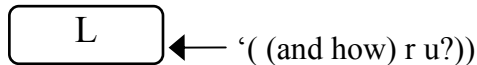


קרן קוראת את הגדרת הפונקציה.

היא מתבוננת בכיסה ומבינה שהתנאי הראשון לא מתקיים (L לא ריקה). לכן היא עוברת לבדיקת התנאי השני ומתבוננת באיבר הראשון ברשימה שבכיסה. איבר זה הוא אטום ולפיכך קרן ניגשת לביצוע `(+ 1 (count-atom (rest L)))`. עליה לבצע למעשה `(+ 1 (count-atom '((and how) r u?)))`.

היא קוראת לכפיר לבצע עבורה את `(count-atom '((and how) r u?))`.

לכפיר כיס אחד.



הוא קורא את הגדרת הפונקציה.

התנאי הראשון לא מתקיים ולכן הוא עובר לבדיקת האיבר הראשון ברשימה שבכיסו, כפי שמורה התנאי השני. גם תנאי זה לא מתקיים (האיבר הראשון הוא רשימה ולא אטום) ולכן עובר כפיר לשורת ה- else וצריך לבצע `(count-atom (rest L))`. מאחר ואין זו פונקציה בסיסית הוא קורא לכרמלה לבצע אותה. הוא נותן לכרמלה את הרשימה `(r u?)` וכרמלה מכניסה אותה לכיסה.

כרמלה קוראת את הגדרת הפונקציה. התנאי הראשון לא מתקיים, אולם בתנאי השני מתברר לה שהראשון ברשימה שבכיסה הוא אטום. לכן היא צריכה לבצע `(+ 1 (count-atom (rest L)))`. היא קוראת לכוכב ומסרת לו את הרשימה `(u?)`

גם **כוכב** מגלה מיד שהראשון ברשימה שלו הוא אטום. הוא צריך לבצע `(+ 1 (count-atom '()))` ומבקש מכרמי לעזור ולחשב את `(count-atom '())`.

בתנאי הראשון בודק **כרמי** אם הרשימה שבכיסו ריקה, ומכיוון שכך הוא מחזיר 0 לכוכב.

כוכב משתמש בערך שכרמי החזיר, מציב 0 במקום `(count-atom '())` ומבצע `(+ 1 0)`. כעת הוא יכול להחזיר לכרמלה שקראה לו לעזרתה את התוצאה 1.

כרמלה מציבה את הערך 1 במקום הפונקציה שביקשה מכוכב לבצע,

היא מחשבת את הביטוי: `(+ 1 1)` ומחזירה לכפיר את המספר 2.

כפיר קיבל מכרמלה את הערך שביקש והוא מחזיר אותו כפי שהוא לקרן.

קרן מציבה את המספר 2 במקום הפונקציה שביקשה מכפיר לבצע.

היא מחשבת את הביטוי: `(+ 1 2)` ומחזירה את המספר 3.

תרגילים נוספים: מבני נתונים מופשטים (Abstract Data Type)

תרגיל 33

מחסנית היא מבנה נתונים המנוהל לפי מדיניות של אחרון-נכנס-ראשון-יוצא LIFO. לדוגמה, אם במחסנית נמצא כבר המספר 17 ועכשיו נכניס אליה גם את המספר 4 אז כאשר נוציא את האיבר שנמצא בראש המחסנית נקבל את 4 ואילו 17 ישאר בתוך המחסנית.

כדי שיהיה אפשר לעבוד עם מחסנית גם ב-Scheme נגדיר את הפונקציות הבאות:

- הפונקציה `empty-stack?` מקבלת מחסנית, מחזירה `#t` אם המחסנית ריקה (ללא אברים) ו-`#f` אחרת.
 - הפונקציה `top` מקבלת מחסנית ומחזירה את האיבר הנמצא בראש המחסנית.
 - הפונקציה `push` מקבלת מחסנית ואיבר `x`.
- הפונקציה מחזירה מחסנית חדשה שאליה הוכנס `x` בראש המחסנית.
- הפונקציה `pop` מקבלת מחסנית ומחזירה אותה ללא האיבר בראש המחסנית.

כתבו את הפונקציות בשפת Scheme

תרגיל 34

תור הוא מבנה נתונים המנוהל לפי מדיניות של ראשון-נכנס-ראשון-יוצא FIFO. לדוגמה, אם בתור נמצא כבר המספר 17 ועכשיו נכניס אליה גם את המספר 4 אז כאשר נוציא את האיבר שנמצא בראש התור נקבל את 17 ואילו 4 ישאר בתוך התור.

כדי שיהיה אפשר לעבוד עם תור גם ב-Scheme נגדיר את הפונקציות הבאות:

- הפונקציה `empty-queue?` מקבלת תור, מחזירה `#t` אם התור ריק (אין בו אברים) ו-`#f` אחרת.
- הפונקציה `first-in-queue` מקבלת תור ומחזירה את האיבר הנמצא בראש התור.
- הפונקציה `enter` מקבלת תור ואיבר `x`. הפונקציה מחזירה תור חדש שאילו הוכנס `x`.

כתבו את הפונקציות בשפת Scheme

תרגיל 35

מערך חד-ממדי הוא מבנה נתונים שמאפשר לגשת אל כל אחד מנתוניו בעזרת אינדקס (מציין). למשל, הנה מערך חד-ממדי של 5 אברים:

	1	2	3	4	5
	32	19	-5	763	100

כדי שיהיה אפשר לעבוד עם מערך חד-ממדי גם ב-Scheme נגדיר את הפונקציות הבאות:

- הפונקציה `get` מקבלת מערך ומיקום מסוים ומחזירה את האיבר הנמצא במיקום המבוקש.
- הפונקציה `update` מקבלת מערך, איבר x ומיקום.
- הפונקציה מחזירה מערך חדש שבו מופיע האיבר x במיקום המבוקש (במקום האיבר הקודם).
- הפונקציה `init` מקבלת מערך ומאתחלת אותו (את כל האברים שלו) לערך מבוקש `value`.

כתבו את הפונקציות בשפת Scheme

תרגיל 36

מעריך דו-ממדי (מטריצה) הוא מבנה נתונים שניתן לגשת לכל אחד מאבריו בעזרת שני אינדקסים. למשל, הנה מעריך דו-ממדי:

	1	2	3	4	5
1	72	70	68	66	64
2	-1	-2	-3	-4	-5
3	30	400	11	21	0

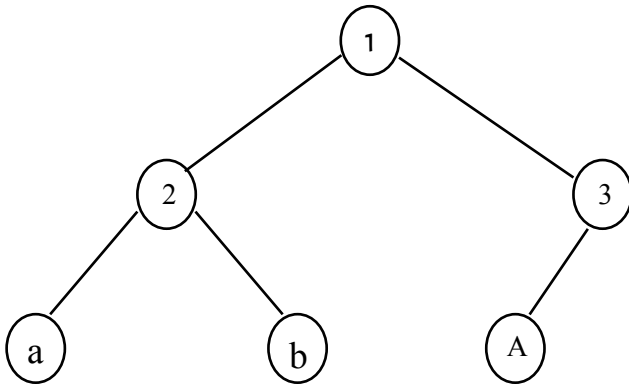
כדי שיהיה אפשר לעבוד עם מעריך דו-ממדי גם ב-Scheme נגדיר את הפונקציות הבאות:

- הפונקציה `get` מקבלת מעריך דו ממדי ושני אינדקסים ומחזירה את האיבר הנמצא במיקום המתאים.
- הפונקציה `update` מקבלת מעריך דו ממדי, איבר x ומיקום.
- הפונקציה מחזירה מעריך דו ממדי חדש שבו מופיע האיבר x במיקום המבוקש (במקום האיבר הקודם).
- הפונקציה `init` מקבלת מעריך דו ממדי ומאתחלת אותו לערך מבוקש `value`.
- הפונקציה `row` מקבלת מעריך ומספר ומחזירה את כל השורה שזה האינדקס שלה (כמעריך חד-ממדי).
- הפונקציה `column` מקבלת מעריך ומספר ומחזירה את כל העמודה שזה האינדקס שלה.
- הפונקציה `main-diagonal` מקבלת מעריך ומחזירה את האלכסון הראשי שלו.

תרגיל 37

עץ הוא מבנה נתונים המורכב מאברים המכונים "צמתים", כאשר לכל צומת יכולים להיות צמתים שהם "בנים" שלו. צמתים שאין להם בנים נקראים "עלים".

לדוגמה,



בעץ הזה יש שורש שערכו 1. לשורש הזה יש שני בנים.

ערכו של הבן השמאלי הוא 2 וערכו של הבן הימני הוא 3.

לעץ יש 3 עלים וערכיהם - a, b, A

כדי שיהיה אפשר לעבוד עם עץ גם ב-Scheme נגדיר את הפונקציות הבאות:

- הפונקציה empty-tree? מקבלת עץ, מחזירה #t אם העץ ריק ו-#f אחרת.
- הפונקציה root מקבלת עץ ומחזירה את השורש שלו.
- הפונקציה left-son מקבלת עץ ומחזירה את הבן השמאלי שלו (איבר או תת-עץ).
- הפונקציה right-son מקבלת עץ ומחזירה את הבן הימני שלו (איבר או תת-עץ).
- הפונקציה in? מקבלת עץ ואיבר x. הפונקציה מחזירה #t אם האיבר שייך לעץ ו-#f אחרת.
- הפונקציה leaf? מקבלת איבר בעץ ובודקת האם הוא עלה.
- הפונקציה replace-root מקבלת עץ ושורש חדש ומחליפה את השורש של העץ.
- הפונקציה replace-left מקבלת עץ ותת-עץ שמאלי ומחליפה אותו.
- הפונקציה replace-right מקבלת עץ ותת-עץ ימני ומחליפה אותו.
- הפונקציה find-first-leaf מקבלת עץ ומחזירה את העלה הראשון שהיא מוצאת בעץ.

כתבו את הפונקציות בשפת Scheme

תרגיל 38

קבוצה היא אוסף נתונים ללא חזרות וללא חשיבות לסדר ביניהם. לדוגמה, $\{ 6 32 7 -59 0 1 \}$ היא קבוצה.

כדי שיהיה אפשר לעבוד עם קבוצות גם ב-Scheme נגדיר את הפונקציות הבאות:

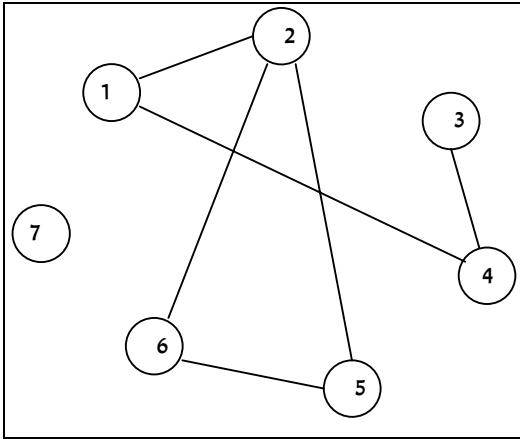
1. פעולות על קבוצה יחידה

- הפונקציה empty-set? מקבלת קבוצה, מחזירה #t אם הקבוצה ריקה ו-#f אחרת.
- הפונקציה in? מקבלת קבוצה ואיבר x. הפונקציה מחזירה #t אם האיבר שייך לקבוצה ו-#f אחרת.
- הפונקציה item מקבלת קבוצה ומחזירה איבר אקראי מתוך הקבוצה.
- הפונקציה insert מקבלת קבוצה ואיבר x. הפונקציה מחזירה קבוצה חדשה שבה מופיע גם האיבר x.
- הפונקציה delete מקבלת קבוצה ואיבר x. הפונקציה מחזירה את הקבוצה ללא x.
- הפונקציה order מקבלת קבוצה ומחזירה את מספר האיברים בקבוצה.

2. פעולות על שתי קבוצות

- הפונקציה intersection מקבלת שתי קבוצות ומבצעת חיתוך שלהן. כלומר, הפונקציה מחזירה קבוצה חדשה שמורכבת רק מהאיברים שמופיעים בשתי הקבוצות.
- הפונקציה union מקבלת שתי קבוצות ומבצעת איחוד שלהן. כלומר, הפונקציה מחזירה קבוצה חדשה שמורכבת מכל האיברים שמופיעים בשתי הקבוצות.
- הפונקציה contain? מקבלת שתי קבוצות ומחזירה #t אם הקבוצה הראשונה מוכלת בשנייה
- (כלומר כל האברים של הקבוצה הראשונה מופיעים בקבוצה השנייה), אחרת יוחזר #f.
- הפונקציה same? מקבלת שתי קבוצות ומחזירה #t אם הן שוות, אחרת יוחזר #f.
- הפונקציה complement מקבלת שתי קבוצות ומחזירה את קבוצת המשלים. כלומר, הפונקציה מחזירה קבוצה חדשה שמורכבת מכל האברים של הקבוצה השנייה שאינם מופיעים בקבוצה הראשונה.

כתבו את הפונקציות בשפת Scheme

תרגיל 39

גרף הוא מבנה נתונים שמורכב מצמתים וקשתות המחברות ביניהם. לדוגמה,

בגרף הזה יש 7 צמתים ו-6 קשתות.

צמתים 1 ו-2 מחוברים ביניהם בקשת.

צמתים 1 ו-3 אינם מחוברים בקשת.

כדי שיהיה אפשר לעבוד עם גרפים גם ב-Scheme נגדיר את הפונקציות הבאות:

- הפונקציה `empty-graph?` בודקת האם הגרף ריק.
- הפונקציה `nodes` מקבלת גרף ומחזירה את רשימת כל הצמתים.
- הפונקציה `edges` מקבלת גרף ומחזירה את רשימת כל הקשתות. (קשת תצוין ע"י שני הצמתים שהיא מחברת).
- הפונקציה `connected?` מקבלת גרף ושני צמתים, היא בודקת האם הם מחוברים בקשת ישירה.
- הפונקציה `add-edge` מקבלת גרף ושני צמתים ומוסיפה ביניהם קשת (אם לא היתה קודם).
- הפונקציה `nodes-number` מקבלת גרף ומחזירה את מספר הצמתים בגרף.
- הפונקציה `edges-number` מקבלת גרף ומחזירה את מספר הקשתות בגרף.
- הפונקציה `legal-route?` מקבלת מסלול בגרף ובודקת האם הוא מסלול חוקי.
- הפונקציה `sub-graph` מקבלת גרף ומסלול. אם המסלול חוקי היא יוצרת גרף שמורכב ממנו

כתבו את הפונקציות בשפת Scheme

