מטלה מספר 5 – שפות תכנות והידור (Scheme)

הנחיות כלליות

- הגשה דרך מערכת ההגשות
- התרגיל צריך לרוץ על שרתי האוניברסיטה u2.
- יש לצרף בהגשה את הקובץ Ex5.scm.
- בסוף הקובץ יש להוסיף את השורה **(load "test.scm")**
- בראש הקובץ יש להוסיף הערה בה רשום:
  - שם הסטודנט
  - ת.ז.
  - מספר קבוצה
  - שם משתמש
- הערה ב- scheme היא ע"י ;;

במטלה אנו נתרגל את השפה scheme. יש לוודא שאתם משתמשים בגרסא R5RS שנלמדה בכיתה.

- ממומלץ שתשתמשו בפעולות map/reduce/filter שנלמדו בשיעור.
- מותר להשתמש בפונקציות הבאות בלבד:
  - string->list
  - string=?
  - char=?
  - car
  - cdr
  - cons
  - length
  - null?
  - pair?
  - list-ref
  - reverse
  - append
  - list
- אסור להשתמש ב- set! (או כל רעיון אימפרטיבי במטלה), הגשה עם חלק אימפרטיבי לא תיבדק!
- אין צורך לבדוק את תקינות הטיפוס וכמות הפרמטרים שמתקבלים בפונקציה.

חלק 1:

- **הפונקציה ends-with**

קלט: suffix (string)

str (string)

פלט: מחזיר true אם ה- suffix הוא סיפא של המחרוזת str, אחרת false

```
(ends-with "" "bazinga!")    ;; true
(ends-with "nga!" "bazinga!") ;; true
(ends-with "bazinga!!" "bazinga!") ;; false
(ends-with "bazidga!" "bazinga!") ;; false
```

(רמז.. יש להפוך את המחרוזת לרשימה של תווים)

- **הפונקציה mul-of-pairs**

קלט: suffix (string)

ls – רשימה של pairs, כאשר כל pair מכיל (string, number)

פלט: מחזיר את מכפלת ערכי הזוגות אשר suffix הוא סיפא של ה- string בזוג (אם אין זוגות כאלו יוחזר 1)

```
(mul-of-pairs "a" (list (cons "a" 2) (cons "b" 2) (cons "aa" 3) (cons "ca"
4))) ;; =24
(mul-of-pairs "aa" (list (cons "a" 1) (cons "b" 2) (cons "aa" 33) (cons "ca"
4))) ;; =33
(mul-of-pairs "aa" '()) ;; =1
```

- **הפונקציה merge**

קלט: ls1 (רשימה מטיפוס כלשהוא)

ls2 (רשימה מטיפוס כלשהוא)

פלט: מחזיר רשימה חדשה המורכבת ממיזוג שתי הרשימות (כאשר האיבר הראשון מ- ls1 לאחר מכן האיבר הראשון מ- ls2 וחוזר חלילה)

(אם רשימה אחת קצרה מהשנייה, נשרשר את הרשימה הארוכה יותר לסוף הרשימה החדשה)

```
(merge '(1 2) (list "a" "b")) ;; (1 "a" 2 "b")
(merge '(1 2) (list "a"))     ;; (1 "a" 2)
(merge '(1) (list "a" "b"))   ;; ((1 "a" "b"))
(merge '() (list "a" "b"))   ;; ("a" "b")
```

- **הפונקציה rotate**

קלט: ls – רשימה (מכל טיפוס)

n – מספר שלם (גדול או שווה ל-0)

פלט: הפונקציה מחזירה רשימה, אשר עברה n סיבובים של ls בכיוון השעון

```
(rotate '(1 2 3 4) 1) ;; (4 1 2 3)
(rotate '(1 2 3 4) 2) ;; (3 4 1 2)
(rotate '(1 2 3 4) 4) ;; (1 2 3 4)
(rotate '(1 2 3 4) 5) ;; (4 1 2 3)
(rotate '(1 2 3 4) 0) ;; (1 2 3 4)
(rotate '(1 2 3 4) 23);; (2 3 4 1)
(rotate '() 3) ;; ()
```

- **הפונקציה quicksort**

קלט: comp – פונקציית השוואה, המקבלת 2 פרמטרים: x,y ומחזירה מספר שלילי אם (x<y), 0 אם הם שווים, ומספר חיובי (אחרת)

פלט: הפונקציה מחזירה פונקציית מיון (עפ"י אלגוריתם quick-sort) אשר מקבל רשימה וממיין את ערכיה עפ"י הפונקציה comp.

(הפונקציה comp היא תקינה, והרשימה המתקבלת היא תקינה)

```
(define ascsort (quicksort (lambda (x y) (- x y)))) ;; will sort numbers in
ascending order
(define descsort (quicksort (lambda (x y) (- y x)))) ;; will sort numbers in
descending order
(define charsort (quicksort (lambda (x y) (- (char->integer x) (char->integer
y))))) ;; will sort characters in ascending order. char->integer returns the
ascii code of a character

(ascsort '(1 6 -4 5 12 7))  ;;(-4 1 5 6 7 12)
(descsort '(1 6 -4 5 12 7)) ;;(12 7 6 5 1 -4)
(charsort '(#\x #\a #\g #\i #\w #\s)) ;;(#\a #\g #\i #\s #\w #\x)
```

חלק 2:

בחלק זה נצטרך לממש פונקציות אשר מייצגות רצף אינסופי.
עבור הרצף נגדיר 2 פונקציות עזר:
hd – מקבל רצף, ומחזיר את ראש הרצף.
tail – מקבל רצף ומחזיר רצף המייצג את המשך הרצף.

(הפונקציות הללו ישמשו אותנו לאורך כל החלק... ולכן יש לחשוב על מימוש יעיל לפונקציות הנ"ל)

- **פונקציית seq**

קלט: n – מספר שלם חיובי

פלט: פונקציה המייצגת את הרצף (n, n+1, n+2, ...)

```
(define s (seq 3)) ;; the sequence 3, 4, 5, 6, ...
(hd s) ;; 3
(define s1 (tail s)) ;; the sequence 4, 5, 6, ...
(hd s1) ;; 4
(hd (tail s1)) ;; 5
(define s2 (tail (tail (tail s1)))) ;; the sequence: 7, 8, 9, ...
(hd s2) ;; 7
(hd (tail (tail (tail s2)))) ;; 10
(define s3 (seq -15)) ;; the sequence -15, -14, -13, ...
(hd (tail s3)) ;; -14
```

שימו לב שלא ניצור <u>אוסף</u> (משמע, לא נאחסן את כל הערכים של הרצף ברשימה), אלא ערכי הרצף יתגלו לאחר כל הפעלה של tail ו- hd.

- **פונקציית seq-gen**

קלט: n – מספר שלם חיובי

g – פונקציה אשר יוצרת את המספר הבא ברצף.

פלט: פונקציה המייצגת את הרצף (n, g(n), g(g(n))....)

```
(define s1 (seq-gen 3 (lambda (n) (- n 1)))) ;; the sequence 3, 2, 1, 0, ...
(define s2 (seq-gen 2 (lambda (n) (* n n)))) ;; the sequence 2, 4, 16, 256,...
(hd s1) ;; 3
(hd (tail s1)) ;; 2
(hd (tail (tail s1))) ;; 1

(hd s2) ;; 2
(hd (tail s2)) ;; 4
(hd (tail (tail s2))) ;; 16
```

- **פונקציית cyclic-seq**

קלט: ls – רשימה לא ריקה מטיפוס כלשהוא

פלט: בהינתן n – אורך הרשימה ls, ו- ls[i] הוא האיבר ה- i ברשימה.

הפונקציה מחזירה את הרצף הבא:

ls[0], ls[1], ..., ls[n-1], ls[0], ls[1]

```
(define s (cyclic-seq '(1 2 3 4)))
(hd s) ;; 1
(hd (tail s)) ;; 2
(hd (tail (tail (tail s)))) ;; 4
(hd (tail (tail (tail (tail s))))) ;; 1
(hd (tail (tail (tail (tail (tail s)))))) ;; 2
```

חלק 3:

פונקציית make-dictionary

קלט: אין.

פלט: מחזירה dictionary ריק.

dictionary מאחסן ערכים של key-value (כאשר ה- key הוא string וה- value הוא מטיפוס כלשהוא).

פעולות על dictionary

- בהינתן dictionary הקרוי d, כדי להוסיף רשומה חדשה ל- d, אנו יכולים להפעיל את d עם הרשומה באופן הבא:

(define newD (d (cons "a" 5)))

ובכך newD יהיה ה- dictionary החדש עם הרשומה החדשה.
במקרה ורשומה **כבר קיימת** יש להחליפה בערך החדש של הזוג.

- כדי לקבל את ערכו של key ב- d , אנו נפעיל את d עם המחרוזת המייצגת את ה- key.

  (d "a") => יחזיר את הערך 5

  אם ה- key לא קיים תוחזר רשימה ריקה (משמע ()')
  (נניח כי לא קיים ערך שהוא רשימה ריקה).

- כדי לקבל את כל ערכי ה- dictionary , נפעיל את d עם הערך ()' משמע (d '())

  אין חשיבות לסדר האיברים ב-dictionary.

```
(define d (make-dictionary)) ;; creates an empty dictionary (bind it to d)
(define d1 (d (cons "a" 1))) ;; d1 is a dic with a single entry ("a", 1)
(d1 "a") ;; = 1
(define d2 (d1 (cons "b" 15))) ;; d2 is a dic with ("a", 1), ("b", 15)
(d2 "a") ;; = 1
(d2 "b") ;; = 15
(define d3 (d2 (cons "a" 66))) ;; d2 is a dic with ("a", 66), ("b", 15)
(d3 "a") ;; = 66
(d3 '()) ;; = (("a" . 66) ("b" . 15))
(define d4 (make-dictionary)) ;; creates an empty dictionary (bind it to d4)
((d4 (cons "a" 99)) "a") ;; = 99
```