



# חלק יבש

## שאלה 1:

התבוננו בקטע הקוד הבא של שפת תכנות כלשהי וענו על הסעיפים הבאים, בתוספת הסבר קצר.

שימו לב:

- תשובה ללא הסבר לא תיבדק!
- הניחו שהקוד אינו מכיל שגיאות סינטקטיות, ושהוא רץ ומסתיים ללא שגיאות על כל קלט אפשרי.
- כמו כן, הניחו שהפונקציה `input_string_read` מקבלת קלט מהמשתמש ומחזירה ערך מטיפוס `.string`.

```
x = 3;

s = read_string_input();

if (isnumber(s) and strToInt(s) > 0)

    x = s + 1.3;

else

    x = "negative input";

print(x);
```

א. האם השפה היא `typed`?

ב. האם בשפה יש `strong typing` או `weak typing`?

ג. האם בשפה יש `explicit typing` או `implicit typing`?

## שאלה 2:

למדו את שפת GO מתוך התיעוד הרשמי שלה:

א. מהם בנאי הטיפוסים בשפה? האם יש שם בנאים שלא מופיעים בקורס? האם יש בנאים תיאורטיים שנלמדו

בקורס ולא מופיעים בשפה? מיהם?

ב. מהם הטיפוסים האטומיים?

ג. נתח את שפת GO לפי הקריטריונים שנלמדו בקורס לסיווג מערכת טיפוסים.

# חלק רטוב

## חלק 1 - מילון

את הפתרון לחלק זה יש לשמור בקובץ dict.sml.

בתרגיל זה נגדיר טיפוס חדש עבור מבנה הנתונים מילון, והפעולות עליו. מילון או, מערך אסוציאטיבי, הוא מבנה נתונים המכיל קישורים בין מפתחות לערכים. כלומר בהינתן מילון d ומפתח k - ייתכנו בדיוק שתי אפשרויות: (1) המפתח k לא קיים במילון d או (2) הערך עבור המפתח k במילון d הוא v.

הוסיפו את ההגדרה הבאה לקובץ הפתרון שלכם:

```
datatype ('a, 'b) dictionary =  
  Nil  
  | Dict of {key: 'a, value: 'b} list;  
exception ItemNotPresent;
```

1. ממשו את הפונקציה **insert** המקבלת מילון, מפתח וערך, ומחזירה מילון חדש לאחר הוספת קישור חדש בין המפתח והערך שהועברו.  
שימו לב: לא ייתכן שיהיו במילון שני מפתחות זהים, ולכן אם המפתח כבר קיים בתוך המילון, עליכם להחליף את הערך הקיים בערך החדש שהתקבל כקלט.

```
val insert = fn : ('a,'b) dictionary -> 'a -> 'b -> ('a,'b)  
dictionary
```

דוגמאת הרצה:

```
- val d = Nil;  
val d = Nil : ('a,'b) dictionary  
  
- val d = insert d 2 #"b";  
val d = Dict [{key=2,value=#"b"}] : (int,char) dictionary  
  
- val d = insert d 1 #"a";  
val d = Dict [{key=2,value=#"b"},{key=1,value=#"a"}] : (int,char)  
dictionary  
  
- val d = insert d 3 #"c";  
val d = Dict  
[{key=2,value=#"b"},{key=1,value=#"a"},{key=3,value=#"c"}]  
: (int,char) dictionary
```

הערה: אין חשיבות לסדר האיברים בתוך המילון. כלומר, הדרישה היחידה היא שלאחר שהכנסנו זוג חדש של מפתח וערך, יהיה ניתן למצוא את הערך לפי המפתח הזה באמצעות הפונקציה **find**, שאותה נממש בסעיף הבא.

2. ממשו את הפונקציה **find** המקבלת מילון ומפתח ומחזירה את הערך המתאים למפתח זה במילון. במידה והמפתח לא קיים תיזרק חריגת **ItemNotPresent**.

```
val find = fn : ('a,'b) dictionary -> 'a -> 'b
```

דוגמאת הרצה:

```
- find d 2;  
val it = #"b" : char  
  
- find d 4;  
uncaught exception ItemNotPresent
```

3. ממשו את הפונקציה **remove** המקבלת מילון ומפתח, ומחזירה מילון חדש לאחר מחיקת הקישור בין המפתח והערך שלו במילון. במידה והמפתח לא קיים במילון תיזרק חריגת `ItemNotPresent`. שימו לב: במידה והאיבר שהוסר היה האיבר היחיד המילון יש להחזיר את המילון הריק כ-`Nil`.

```
val remove = fn : ('a,'b) dictionary -> 'a -> ('a,'b)  
dictionary
```

דוגמאת הרצה:

```
- insert Nil "a" [1,2];  
val it = Dict [{key="a",value=[1,2]}] : (string,int list)  
dictionary  
  
- remove it "a";  
val it = Nil : (string,int list) dictionary  
  
- remove it "a";  
uncaught exception ItemNotPresent
```

4. ממשו את הפונקציה **keys** המקבלת מילון ומחזירה את רשימת כל המפתחות הקיימים במילון. במידה והמילון ריק יש להחזיר רשימה ריקה.

```
val keys = fn : ('a,'b) dictionary -> 'a list
```

דוגמאת הרצה:

```
- keys d;  
val it = [2,1,3] : int list
```

5. ממשו את הפונקציה **values** המקבלת מילון ומחזירה את רשימת כל הערכים הקיימים במילון. במידה והמילון ריק יש להחזיר רשימה ריקה.

```
val values = fn : ('a,'b) dictionary -> 'b list
```

דוגמאת הרצה:

```
- values d;  
val it = ["b",#"a",#"c"] : char list
```

הערה: מאחר ואין חשיבות לסדר בין האיברים במילון גם אין חשיבות לסדר בין האיברים ברשימות

אותם הפונקציות keys ו-values מחזירות. בדוגמא הנתונה, כל רשימה המכילה פרמוטציה של האיברים 1,2,3 הינה פלט תקין עבור הרצת השורה keys d.

## חלק 2 - MLISP

את פתרון שאלה זו עליכם להגיש בקובץ mlisp.sml

**שימו לב! ניתן ואף רצוי להשתמש בפונקציות מתרגילי בית הקודמים בשאלה זו. אל תוסיפו את המימוש שלהם לקובץ זה ובפרט כל פונקציות והכרזות העזר שלכם צריכות להיות מוסתרות! במהלך בדיקת התרגיל אנחנו נוסיף מימוש (שלנו) עבור הפונקציות שנדרשתם לממש בתרגילי בית הקודמים.**

### מוטיבציה

בתרגילי בית הקודמים התעסקנו במימוש את הפונקציה tokenize ובמימוש מבנה הנתונים של הסביבה של תוכנית MLisp. בתרגיל זה נשתמש בפלט הפונקציה tokenize על מנת לבנות ביטוי (S-Expression) שניתן להזינו לפונקציה EVAL (שתמומש בתרגיל בית הבא). כלומר בתרגיל זה נממש פונקציה התקרא parse שתקבל כקלט רשימת tokens ותפלוט טיפוס בשם SExp (שנגדיר אותו בעצמינו כפי שלמדנו בתרגול על Datatypes).

### מערכת הטיפוסים של Mlisp

כפי שלמדתם בהרצאה, שפת LISP היא untyped language. וקיים בה טיפוס יחיד בשם S-Expression. כפי שאתם יודעים S-Expression הוא עץ בינארי שהעלים שלו הם אטומים. בשפת MLisp שלנו קיימים שלושה סוגי אטומים:

- NIL - רשימה ריקה.
  - NUMBER - מספר שלם.
  - SYMBOL - שם בתוכנית המקושר לערך בסביבה.
- הוסיפו את הגדרת הטיפוס הבאה לקובץ הפתרון שלכם:

```
datatype Atom =  
  SYMBOL of string | NUMBER of int | NIL;
```

כעת נרצה להגדיר את הטיפוס SExp,

הוסיפו את הגדרת הטיפוס הבאה לקובץ הפתרון שלכם:

```
datatype SExp =  
  ATOM of Atom | CONS of (SExp * SExp);
```

## פונקציות דיבוג (לא להגשה)

שימו לב כי חלק זה נועד בכדי לעזור לכם לדבג את התרגיל ואין חובה לממש אותו

1. כתוב פונקציה בשם isListable שמקבלת ערך מטיפוס SExp ומחזירה true אם ורק אם ניתן לכתוב את הערך כאטום, כ- NIL, או כרשימה שמכילה רשימות, אטומים ו NIL.

החתימה:

```
val isListable = fn : SExp -> bool;
```

2. כתוב פונקציה שגם מתרגמת את הרשימה המקוננת למחרוזת, תוך שימוש בהזחות (אינדנטציה של שני רווחים). עבור הקלט:

```
(a b (c d))
```

יצא הפלט:

```
a
  b
  c
    d
```

החתימה:

```
val withIndentations = fn : SExp -> string;
```

כלומר את הפריט הראשון ברשימה יש להכניס בשורה נפרדת, ואת כל פרטי הרשימה בהזקה (אינדנטציה) של שני רווחים לעומת הפריט הראשון ברשימה.

דוגמאת הרצה:

```
CONS (
  ATOM (SYMBOL ("a")),
  CONS (
    ATOM (SYMBOL ("b")), ATOM (NIL)) );
val it = CONS (ATOM (SYMBOL "a"), CONS (ATOM (SYMBOL "b"),
ATOM NIL)): SExp
withIndentations (it);
val it = "a\n  b\n": string
print(it);
a
  b
```

```
val it = (): unit
```

2. הכן מחרוזת המייצגת את הרשימה בכתוב פונקציונלי, כלומר עבור הקלט:

```
(a b (c d))
```

יצא הפלט:

```
a(b,c(d))
```

החתימה:

```
val asFunctional = fn : SExp -> string;
```

דוגמת הרצה:

```
CONS (
  ATOM (SYMBOL ("a")),
  CONS (
    ATOM (SYMBOL ("b")), ATOM (NIL) );
val it = CONS (ATOM (SYMBOL "a"), CONS (ATOM (SYMBOL "b"),
ATOM NIL)): SExp
asFunctional (it);
val it = "a(b)": string;
```

הערות:

- הניחו כי הקלט לפונקציות withIndentations ו-asFunctional הינו רשימה מקוננת, כלומר אם נעביר אותו לפונקציה isListable, הפונקציה תחזיר true.
  - אתם רשאים להשתמש בפונקציה המובנית Int.toString בכדי להמיר מספר שלם למחרוזת.
  - שימו לב כי לאחר שתממשו את הפונקציה parse תוכלו להשתמש בפונקציות הדיבוג בצורה הבאה:
    - withIndentations (parse (tokenize "(a b)"));
- ```
val it = "a\n b\n" : string
```

## מימוש הפונקציה parse (החל מנקודה זו הסעיפים הם חובה- ויבדקו בהגשה)

ממשו את הפונקציה :

```
val parse = fn : string list -> SExp
```

כאשר הקלט שלה הוא רשימת tokens המייצגת תוכנית והפלט שלה הוא התוכנית המפורסרת באופן הרקורסיבי הבא:

- מספר x הוא:

ATOM(NUMBER(x))

- מחרוזת s היא:

ATOM(SYMBOL(s))

- רשימה (רצף SExp המופרדים ברווחים הנמצאים בין סוגריים) (x1 x2 x3 ... xn) הוא :  
CONS(x1, CONS(x2, CONS(x3, CONS(..., ATOM(NIL)))))

ניתן להניח כי במקרה שבו מדובר ברשימה- הסוגריים מאוזנים.

ניתן להבין את אופן פעולת parse בקלות רבה ע"י דוגמאות ההרצה הבאות:

דוגמאות הרצה:

דוגמא 1:

```
CONS (
  ATOM (SYMBOL ("+")) ,
  CONS (
    ATOM (NUMBER (2)) ,
    CONS (
      ATOM (NUMBER (3)) , ATOM (NIL)
    )
  )
)
) = (parse (tokenize "(+ 2 3)"));
- val it = true : bool
```

דוגמא 2:

```
CONS (
  ATOM (SYMBOL ("+")) ,
  CONS (
    ATOM (NUMBER (1)) ,
```



```

CONS (
  CONS (
    ATOM (SYMBOL ("+" ) ,
    CONS (
      ATOM (NUMBER (1) ) ,
      CONS (
        ATOM (NUMBER (3) ) , ATOM (NIL)
      )
    )
  ) , ATOM (NIL)
)
)
) = (parse (tokenize "(+ 1 (+ 1 3))"));
- val it = true : bool

```

:3 דוגמא

```

CONS (
  ATOM (SYMBOL ("+" ) ,
CONS (
  CONS (
    ATOM (SYMBOL ("+" ) ,
    CONS (
      ATOM (NUMBER (1) ) ,
      CONS (
        ATOM (NUMBER (3) ) , ATOM (NIL)
      )
    )
  )
  , CONS (
    ATOM (NUMBER (1) ) , ATOM (NIL)
  )
)
) = (parse (tokenize "(+ (+ 1 3) 1)"));
- val it = true : bool

```

:4 דוגמא

```

CONS (
  ATOM (SYMBOL ("define" ) ,

```

```

CONS (
  ATOM (SYMBOL ("r"))
  , CONS (
    ATOM (NUMBER (5)) , ATOM (NIL)
  )
)
) = (parse (tokenize "(define r 5)"));
- val it = true : bool

```

## דוגמה 5:

```

ATOM (SYMBOL "a") = (parse (tokenize "a"));
- val it = true : bool

```

למען מניעת בלבול נרשום כאן גם את הפלט של tokenize בכל אחד מהמקרים בכדי שלא תסתמכו על מימוש שגוי שלכם מהתרגיל בית 1 ותוכלו לתקן אותו בהתאם על מנת לאפשר לכם לעבוד בקלות על תרגיל זה:

```

- tokenize "(+ 2 3)";
val it = ["(", "+", "2", "3", ")"] : string list
- tokenize "(+ 1 (+ 1 3))";
val it = ["(", "+", "1", "(", "+", "1", "3", ")", ")", ")"] : string list
- tokenize "(+ (+ 1 3) 1)";
val it = ["(", "+", "(", "+", "1", "3", ")", "1", ")", ")"] : string list
- tokenize "(define r 5)";
val it = ["(", "define", "r", "5", ")", ")"] : string list

```

## הנחיות

- בתרגיל זה ניתן להשתמש רק בחומר שנלמד בשפת ML עד (וכולל) תרגול 8. אין להשתמש באף פונקציה או תכונה של השפה שלא נלמדה בתרגולים.
- רשימת הקבצים שצריכים להופיע בתוך קובץ ה-`zip` היא:  
`dry.pdf, dict.sml, mlisp.sml`
- בכל קובץ קוד, הוסיפו בשורה הראשונה הערה המכילה את השם, מספר ת"ז וכתובת הדואר האלקטרוני של המגישים מופרדים באמצעות רווח.
- על החלק היבש להיות מוקלד, אין להגיש סריקה או צילום של התשובות לחלק זה.
- שם קובץ ההגשה יהיה `EX4_ID1_ID2.zip` כאשר ID1, ID2 הם מספרי ת.ז. של המגישים בסדר עולה ממש. אם לא ניתן לסדר את מספרי ת"ז בסדר שכזה יש לפנות למשרד הפנים.
- אין צורך להגיש ניירת הסמסטר. תא הקורס לא יבדק במהלך הסמסטר, אז אנא חסכו בנייר.

- בודקי התרגילים מאוד אוהבים Memes. שתפו את תחושותיכם במהלך פתירת התרגיל באמצעות Meme מתאים על דף השער בהגשה - אולי יצא מזה משהו מעניין!

**בהצלחה!**

## **תיקונים והבהרות**

- 13.12.20 - הבהרה לגבי תקינות קלט MLISP.
- 18.12.20 נוספה דוגמה ל-parse.