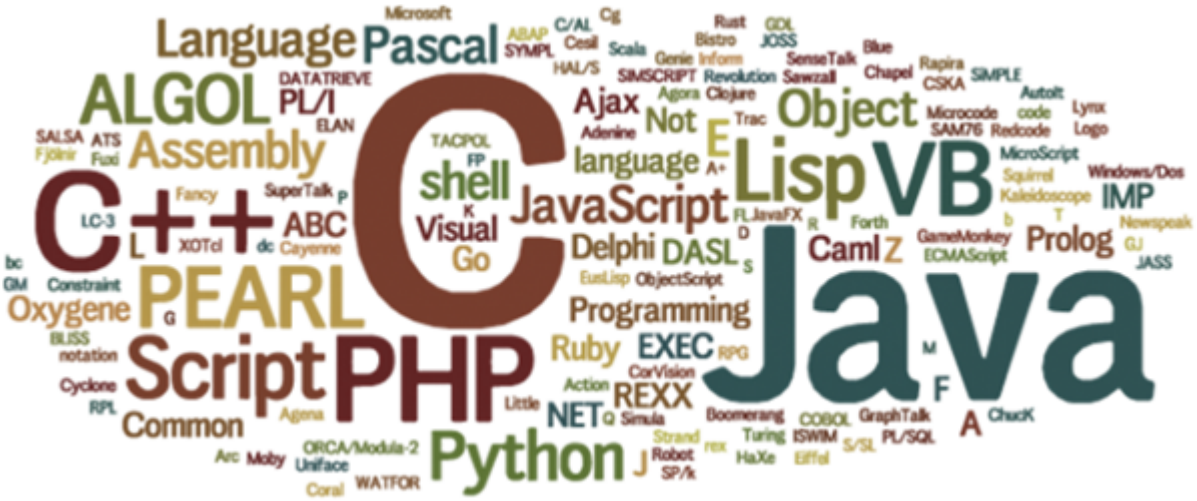
שפות תכנות, 236319

פרופ' י. גיל

חורף 2020-2021



תרגיל בית 2

תאריך פרסום: 10/11/2020
מועד אחרון להגשה: 24/11/2020
מועד אחרון להגשה מאוחרת: 28/11/2020
מתרגל אחראי: מרח גומייד
אי-מייל: marah@campus.technion.ac.il
בפניה בדוא"ל, נושא ההודעה (subject) יהיה "PL-EX2" (ללא המירכאות).

תרגיל בית זה מורכב משני חלקים, חלק יבש וחלק רטוב.
לפני ההגשה, **ודאו שההגשה שלכם תואמת את הנחיות ההגשה בסוף התרגיל.**
תיקונים והבהרות יפורסמו בסוף מסמך זה, אנא הקפידו להתעדכן לעתים תכופות.

חלק יבש

קרא את הפרק הבא בתיעוד של שפת התכנות D, המתאר את המבנה הלקסיקלי של השפה (כלומר אלו אסימונים יש בה)

א. כתוב ביטוי רגולרי (ניתן להשתמש בכתיב של SED) עבור `DecimalFloat`
ב. בשפת C, יש מילים שמורות שהן מזהים. בשפת פסקל, יש מזהים מוגדרים מראש. באיזו מבין שתי השיטות משתמשת שפת D? הסבר.
ג. מיין את המילים השמורות של שפת D לפי הקטגוריות הבאות:

1. Names:
   1.1. Names of types
   1.2. Names of values
   1.3. Names of operators
   1.4. Add your category/categories here
2. Forbidden: can never be used
3. Punctuation (all other keywords)

חזור בקצרה על שפת Common Lisp, כאן

ד. האם יש סוגי הערות שקיימים ב-D שאין ב- Common Lisp? אם כן, אלו הם?
ה. האם יש סוגי הערות שקיימים ב Common Lisp אך לא ב-D? אם כן, אלו הם?
ו. הסבר את השווה והשונה בין self evaluating symbols ובין מזהים שמורים, ומזהים מוגדרים מראש.
הרחב הסבר זה כדי לכלול KEYWORDS של Common Lisp, ומדוע KEYWORDS אלו אינם מילים שמורות.

ז. צייר את מבנה ה S-EXPRESSION של הגוף של הפונקציה print-list בהגדרה הבאה:

```
(defun print-list (list)
  (dolist (i list)
    (format t "item: ~a~%" i)))
```

ח. מהם סוגי הליטרלים של Common Lisp לפי המסמך (ולפי המסמך בלבד)? כדאי להציץ בדוגמאות, ולבדוק

חלק רטוב

בחלק זה עליכם להסתיר פונקציות עזר בעזרת let ו-local.

ניתן להניח תקינות קלט (אלא אם צוין אחרת)
כפי שצוין בהנחיות: בתרגיל זה ניתן להשתמש רק בחומר שנלמד בשפת ML עד תרגול 4 (כולל). אין להשתמש באף פונקציה או תכונה של השפה שלא נלמדה בתרגולים.

משימה 1 :

שימו לב : במשימה זו אסור להשתמש בפונקציה List.nth.

א. ממשו את הפונקציה **valAt** אשר מקבלת רשימה ואינדקס ומחזירה את האיבר באינדקס ה-i ברשימה.

```
val valAt = fn : 'a list -> int -> 'a
```

דוגמאות הרצה:

```
- valAt [1,2,3,4] 3;
val it = 4 : int
- valAt [1,2,3,4] 1;
val it = 2 : int
```

ב. כתבו פונקציה אשר מקבלת רשימה של מספרים ורשימה של אינדקסים וסוכמת את הרשימה של המספרים באינדקסים הנתונים ברשימת האינדקסים. כלומר נסכום רק את האיברים אשר האינדקס שלהם מופיע ברשימת האינדקסים.

```
val sumAtIndices = fn : int list -> int list -> int
```

דוגמאות הרצה:

```
- sumAtIndices [1,~3,9,6,~35,15,18,0,4] [1, 7, 2];
val it = 6 : int
- sumAtIndices [14,7,15,1,6,3,4,7,2,1,5,13] [6,8,1];
val it = 13 : int
```

משימה 2 - Anagrams

Anagram (אנגרמה) : היא יצירת מילה חדשה מערבוב אותיותיה של מילה קיימת, או משפט חדש מערבוב אותיות של משפט קיים.

בשאלה זו תממשו פונקציה המקבלת שתי מחרוזות ובודקת אם אחת המחרוזות היא אנגרמה לשנייה.

שימו לב, אתם נדרשים לתמוך באנגרמות שהן **case insensitive**.

הנחה: המחרוזות בשאלה זו מורכבות רק מאותיות אנגלית (קטנות וגדולות).

ע"מ לעשות זאת, עליכם לממש את הפונקציות הבאות:

א. ממשו את הפונקציה **toLower** המקבלת מחרוזת, ומחזירה את אותה מחרוזת כאשר כל האותיות הגדולות במחרוזת הפכו לאותיות קטנות.

```
val toLower = fn : string-> string
```

דוגמאות הרצה:

```
- toLower “AnaGRam” ;
val it = “anagram” : string
- toLower “anagram” ;
val it = “anagram” : string
```

ב. ממשו את הפונקציה **countOccurrs** המקבלת tuple של מחרוזת ואות אנגלית קטנה, הפונקציה תחזיר את מספר ההופעות של האות במחרוזת.

```
val countOccurrs = fn : string * char -> int
```

דוגמאות הרצה:

```
- countOccurrs (“AnaGRam”,#”a”) ;
val it = 3 : int
- countOccurrs (“AnaGRam”,#”b”) ;
val it = 0 : int
```

ג. ממשו את הפונקציה **getAllOccurrs** המקבלת מחרוזת, הפונקציה מחזירה רשימה כך שכל איבר ברשימה הוא tuple של אות אנגלית קטנה ומספר ההופעות של האות במחרוזת שקיבלה הפונקציה, כאשר ה- tuples יהיו מסודרים לפי סדר האותיות באנגלית.

```
val getAllOccurrs = fn : string -> (char * int) list
```

דוגמאות הרצה:

```
- getAllOccurrs("AnaGRam") ;
val it = [
(#"a",3),(#"b",0),(#"c",0),(#"d",0),(#"e",0),(#"f",0
),(#"g",1),(#"h",0),(#"i",0),(#"j",0),(#"k",0),(#"l"
,0),... ] : (char * int) list
```

הערה: אם אתם רוצים לראות את כל התוצאה, אתם יכולים להשתמש בפקודה :

Control.Print.printLength := 30;

ד. ממשו את הפונקציה **areAnagrams** המקבלת שתי מחרוזות ומחזירה true אם אחת המחרוזות היא אנגרמה של השנייה, ו- false אחרת.

```
val areAnagrams = fn : string * string -> bool
```

דוגמאות הרצה:

```
- areAnagrams ("AnaGRam","maraaNg") ;
val it = true : true
```

משימה 3 – MLISP

את הפתרון לחלק זה יש לשמור בקובץ **mlisp.sml**.

מוטיבציה

בתרגיל בית הקודם מימשנו את הפונקציה tokenize. מטרתה הייתה חלוקת הקוד בlisp לtokens. בתרגיל זה נמשיך לכתוב פונקציות עזר עבור מימוש גרסה של מפרש של lisp. אך הפעם נתעסק בייצוג של מבנה הסביבה.

סביבה כפי שמוגדר בהרצאות היא אוסף כל הכריכות (bindings) בין ערכים לשמות בתוכנית. נרצה בסופו של דבר לממש מילון המקשר בין string לבין ערך. ויתרה מכך נרצה לממש מחסנית של סביבות כדי שמפרש שלנו יאפשר לקנן סביבות ובכך לתמוך בהגדרות פונקציות עם פרמטרים.

ייצוג הסביבה ב ML

סביבה בודדת תהיה מיוצגת ע"י **פונקציה** שמקבלת **string** ומחזירה את הערך המקושר לאותו string. סביבה של תוכנית lisp היא מחסנית (list) של פונקציות שכל אחת מהן מייצגת סביבה. חיפוש כריכה בסביבת התוכנית משמעותו לעבור על כל הסביבות במחסנית ולחפש את הכריכה.

חריגות

חריגות (exceptions) עדיין לא נלמדו בתרגולים בשלב זה, אבל על מנת לממש מילון נהיה חייבים להגדיר ולזרוק חריגות במקרים מסויימים. לכן נציג כעת כיצד להגדיר חריגה בML.

(בהמשך הקורס יהיה תרגול שלם שיוקדש לחריגות אז אל דאגה).

למשל הגדרת חריגה בשם Undefined:

```
exception Undefined;
```

זריקת חריגה זו מבוצעת באופן הבא:

```
raise Undefined
```

הפונקציה הבאה תזרוק את החריגה במקרה שבו n הוא 0:

```
fun foo n = if n = 0 then raise Undefined else "OK";
```

הגדירו בתחילת קובץ הפתרון שלכם את שתי החריגות:

```
exception Undefined;
```

```
exception Empty;
```

סעיף 1

הגדירו את הפונקציה:

```
val initEnv = fn : unit -> string -> 'a
```

הפונקציה initEnv מקבלת unit ותחזיר **פונקציה** שתייצג סביבה ריקה. כלומר סביבה שאין בה bindings ולכן לכל string שהיא (הפונקציה) תקבל היא תזרוק את החריגה Undefined.

דוגמאות הרצה:

```
- val env : (string -> int) = initEnv();

val env = fn : string -> int

- env "x";


uncaught exception Undefined

  raised at: mlisp_ex2.sml:16.42-16.51

- env "message";


uncaught exception Undefined

  raised at: mlisp_ex2.sml:16.42-16.51
```

סעיף 2

ממשו את הפונקציה:

```
val define = fn : string -> (string -> 'a) -> 'a -> string ->
'a
```

הפונקציה מקבלת בצורת curried את הפרמטרים הבאים:

- מחרוזת המייצגת את שם המקושר לכריכה.

- פונקציה המייצגת את הסביבה הישנה.
- הערך שמקושר בכריכה.

הערה: במידה וקיימות מספר הפעלות של define עבור אותו השם המקושר לכריכה, על ההפעלה האחרונה ביותר להיות היחידה שמשפיעה (כלומר עליה לדרוס / להסתיר את הקישורים של ההפעלות הקודמות עבור שם זה).

והיא תחזיר פונקציה שמייצגת את הסביבה לאחר הגדרת הכריכה.

דוגמאות הרצה:

```
- val env : (string -> int) = initEnv();

val env = fn : string -> int

- val env = define "x" env 5;

val env = fn : string -> int

- env "x";

val it = 5 : int

- env "y";


uncaught exception Undefined

  raised at: mlisp_ex2.sml:16.42-16.51

- val env = define "y" env 7;

val env = fn : string -> int

- env "x";

val it = 5 : int

- env "y";

val it = 7 : int

- env "z";


uncaught exception Undefined
```

```
  raised at: mlisp_ex2.sml:16.42-16.51
-
```

סעיף 3

הגדירו את הפונקציה הבאה בתוכנית שלכם.

```
fun emptyNestedEnv () = [initEnv ()];
```

פונקציה זו מחזירה רשימה (מחסנית הסביבות) שמכילה את הסביבה הריקה בלבד.

ממשו את **הפונקציות** הבאות:

```
val pushEnv = fn : 'a -> 'a list -> 'a list

val popEnv = fn : 'a list -> 'a list

val topEnv = fn : 'a list -> 'a
```

הפונקציות האלה מבצעות פעולות פשוטות על מחסנית הסביבות.

pushEnv : הוספת סביבה כלשהי לראש המחסנית.

popEnv : הסרת הסביבה שנמצאת בראש המחסנית.

topEnv : תחזיר את הסביבה שבראש המחסנית.

שימו לב שאלו הפעולות הסטנדרטיות שאתם מכירים והטיפוס הוא רשימה גנרית.

הפונקציות **topEnv** ו **popEnv** יזרקו את החריגה Empty במקרה שהמחסנית ריקה.

סעיף 4

ממשו את הפונקציה:

```
val defineNested = fn : string -> (string -> 'a) list -> 'a ->
(string -> 'a) list
```

הפונקציה הזו מגדירה כריכה חדשה על הסביבה שבראש המחסנית.

הפרמטרים:

1. מחרוזת המייצגת את שם המקושר לכריכה.
2. מחסנית סביבות.
3. הערך שמקושר בכריכה.

כלומר, הפונקציה משנה את הערך עבור הכריכה שבראש המחסנית להיות הערך שבפרמטר מספר 3, אך לא משנה את אף אחת מהכריכות האחרות.

הערה: במידה ומחסנית הסביבות היא ריקה צריכה להיזרק השגיאה Empty.

סעיף 5

ממשו את הפונקציה:

```
val find = fn : string -> (string -> 'a) list -> 'a
```

פונקציה זו מקבלת שם ומחסנית סביבות ומחזירה את הערך המקושר לשם זה (בסביבה החיצונית ביותר).

כעת שימו לב, ייתכן ותיתקלו בחריגות שזרקתם בפונקציות מהסעיפים הקודמים (רמז רמז), על מנת לתפוס חריגה בשפת ML עליכם להשתמש במילה השמורה handle.

למשל, על מנת לתפוס את החריגה Empty שהגדרתם בתחילת התרגיל נצטרך לכתוב:

handle Empty => <how to handle the exception>

כאשר תופסים חריגה, עלינו להגדיר איך להתמודד איתה.

למשל אם נרצה שכאשר נתפוס את החריגה Empty נפלוט רשימה ריקה, נצטרך לכתוב:

handle Empty => []

אם מחסנית הסביבות שהתקבלה היא ריקה, עליכם לזרוק את החריגה Undefined.

בנוסף, אם לשם שהתקבל בקלט אין אף ערך מקושר בסביבה, עליכם לזרוק את החריגה Undefined.

דוגמאות הרצה:

```
- val env : (string -> string) list = emptyNestedEnv ();

val env = [fn] : (string -> string) list
```

```
- val env = defineNested "message" env "Hello, World";
val env = [fn] : (string -> string) list
- find "message" env;
val it = "Hello, World" : string
- val env = pushEnv (initEnv ()) env;
val env = [fn,fn] : (string -> string) list
- find "message" env;
val it = "Hello, World" : string
- val env = defineNested "message" env "NOPE";
val env = [fn,fn] : (string -> string) list
- find "message" env;
val it = "NOPE" : string
- val env = popEnv env;
val env = [fn] : (string -> string) list
- find "message" env;
val it = "Hello, World" : string
```

הנחיות

- בתרגיל זה ניתן להשתמש רק בחומר שנלמד בשפת ML עד (וכולל) תרגול 4. אין להשתמש באף פונקציה או תכונה של השפה שלא נלמדה בתרגולים.
- רשימת הקבצים שצריכים להופיע בתוך קובץ ה-zip היא:
  `dry.pdf,hw2.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 EX2_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:

1. בחלק הרטוב, ניתן להניח תקינות קלט.
2. 15.11.20 : תיקון בחלק היבש: בשאלה צריך לנתח את השפה Common Lisp ולא השפה Mini Lisp (שהיא תת שפה מאוד מצומצמת של Common Lisp) שראיתם בהרצאות ובתרגולים. כדי לעשות זאת היעזרו בלינק המצורף בשאלה.

F.A.Q:

1. מה הכוונה בקלט תקין במשימה 1 בחלק הרטוב?

ז"א שכהפונקציה מקבלת רשימה ואינדקס בסעיף א', אפשר להניח שהאינדקס הוא חוקי (כלומר הוא מספר בין 0 ל- גודל הרשימה -1), לכן רשימה ריקה בסעיף הראשון לא נחשבת כקלט תקין.

לגבי סעיף ב' אפשר להניח שרשימת האינדקסים תכיל רק אינדקסים חוקיים (כמו שהסברתי לעיל) , ובמקרה של רשימה ריקה , רשימת האינדקסים החוקית היחידה הינה רשימה ריקה.

2. איך הפונקציה find אמורה לחפש את השם במשימה 3 ברטוב ?

הפונקציה מתחילה לחפש מהסביבה החיצונית ביותר, אם היא לא מוצאת בה את השם הנדרש, היא ממשיכה לחפש בסביבה הבאה.

אם השם מקושר לשני ערכים שונים בסביבות שונות, הפונקציה תחזיר את הערך הראשון שתמצא (כלומר הערך שהוגדר בסביבה הכי חיצונית), אפשר לראות את זה בדוגמא, כשהוגדרו שני ערכים שונים בשתי סביבות שונות לשם message, במקרה הזה הוחזר הערך "NOPE" שהוגדר בסביבה הכי חיצונית.