שפות תכנות, 236319

חורף 2020-2021

תרגיל בית 5

תאריך פרסום: 29.12.2020
מועד אחרון להגשה: 12.1.2021
מועד אחרון להגשה מאוחרת: 16.1.2021
מתרגל אחראי: מרח גומייד
אי-מייל: marah@campus.technion.ac.il
בפניה בדוא"ל, נושא ההודעה (subject) יהיה "PL-EX5" (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

שאלה 1- RTTI

1. קראו את פרק 5.6 בחוברת השקפים בנושא **RTTI**.
מהו מנגנון RTTI? סכמו בשני משפטים.
2. האם בשפת C יש מנגנון RTTI?
אם כן - תארו אותו.
אם לא - הסבירו למה.
3. הסבירו: כיצד מנגנון איסוף אשפה משתמש במנגנון RTTI?
4. תארו שימוש אחר (שאינו איסוף אשפה) למנגנון RTT בשפות הדוגלות ב static typing.

שאלה 2- מערכים וזיכרון

1. ציינו את הדומה והשונה בין static array ו dynamic array.
2. ציינו את הדומה והשונה בין stack based array ו associative array.
3. בחרו אלגוריתם ניהול זיכרון אוטומטי שלא נראה בכיתה. הסבירו את האלגוריתם לשלביו, ציינו את היתרונות שלו, את החסרונות שלו (ביחס לאלגוריתמים האחרים שהוצגו, וגם ביחס לאפשרות של ניהול זיכרון ידני).

שאלה 3- אוסף שאלות קצרות

1. מה ההבדל בין type error ל pseudo type error, תנו דוגמה לכל אחד.
2. ציינו את כל שיטות שערוך שראינו בקורס, כתבו הסבר קצר על כל אחת ותנו דוגמת קוד קצרה בסינטקס דמוי C, שעבורה היה פלט שונה עבור כל שיטת שערוך שראינו.
3. הסבירו את משמעות הביטוי- "בנאי אורתוגונלי".

חלק רטוב

את פתרון שאלה זו עליכם להגיש בקובץ `hw5.sml`

בחלק זה עליכם להסתיר פונקציות עזר בעזרת let ו-local

משימה 1 : סדרה חשבונית

במשימה זו אסור להשתמש בנוסחאות מתמטיות של סדרה חשבונית, עליכם לפתור את השאלה ע"י שימוש ברצפים.

לכל אורך השאלה, נשתמש בהגדרה הבאה של רצף (כפי שנלמד בתרגולים):

```
datatype 'a seq = Nil | Cons of 'a * (unit-> 'a seq);
```

תזכורת: סדרה חשבונית היא סדרה של מספרים, שבה ההפרש בין כל שני איברים עוקבים הוא קבוע, האיבר הכללי של סדרה חשבונית נתון ע"י הנוסחה:

$$a_n = a_1 + d(n-1)$$

כאשר a_1 הוא האיבר הראשון בסדרה, ו- d הוא ההפרש.

א. ממשו פונקציה בשם **arithmeticSeq** המקבלת את האיבר הראשון וההפרש של סדרה חשבונית, ומחזירה את הסדרה החשבונית כרצף.

```
val arithmeticSeq = fn : int -> int -> int seq
```

דוגמת הרצה:

```
- arithmeticSeq 3 3;
val it = Cons(3,fn) : int seq
```

הסבר: הקריאה לעיל אמורה להחזיר את הסדרה הבאה :

```
3,6,9,12,15,.....
```

ב. ממשו פונקציה בשם **getSubSeq** המקבלת סדרה חשבונית (כלומר האיבר הראשון וההפרש של סדרה חשבונית), אינדקס התחלה s ואינדקס סיום e, הפונקציה תחזיר רשימה המכילה את תת- הסדרה שמתחילה באינדקס s ומסתיימת באינדקס e מתוך הסדרה המקורית.

שימו לב: **1.** אינדקסים של הסדרה מתחילים מ- 1 ולא 0.

2. אפשר להניח שמתקיים: s,e >0, s<=e

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

דוגמת הרצה:

```
- getSubSeq 3 3 2 5;
val it = [6,9,12,15] : int list
```

ג. ממשו פונקציה בשם **getKDivElems** המקבלת סדרה חשבונית(כלומר האיבר הראשון וההפרש של סדרה חשבונית), מספר **אי-שלילי** n, ומספר **חיובי** k.
הפונקציה תחזיר את n האיברים הראשונים בסדרה שהאינדקס שלהם מתחלק במספר k.
שימו לב: **1.** אינדקסים של הסדרה מתחילים מ- 1 ולא 0.
2. אפשר להניח שמתקיים: k>0, n>=0.

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

דוגמת הרצה:

```
- getKDivElems 3 3 4 2;
val it = [6,12,18,24] : int list
```

משימה 2 : עצים עצלים

בשאלה זו אין להשתמש בפונקציות שנלמדו בכיתה.

נגדיר עץ בינארי עצל באופן הבא: הוסיפו את ההגדרה הבאה בתחילת קובץ הפתרון שלכם.

```
datatype 'a lazyTree = tNil | tCons of 'a * (unit -> 'a
lazyTree) * (unit -> 'a lazyTree);
```

הבנאי tCons מקבל כפרמטר ראשון את תוכן הצומת, כפרמטר שני פונקציה המחזירה את תת העץ השמאלי, וכפרמטר שלישי פונקציה המחזירה את תת העץ הימני.

1. ממשו את הפונקציה:

```
lazyTreeFrom : int -> int lazyTree
```

אשר מקבלת את ערך השורש ובונה עץ עצל אין סופי שערכיו מצייתים לחוקיות הבאה:
לצומת שערכו X מתקיים שערך בנו השמאלי הוא 2*X וערך בנו הימני הוא 2*X+1.
למשל עבור ההרצה הבאה:

```
- lazyTreeFrom(1);
val it = tCons (1,fn,fn) : int lazyTree
```

נקבל נקבל עץ שבשורש שלו יש 1, ערך בנו השמאלי הוא 2, ערך בנו הימני הוא 3, וכך הלאה.

2. ממשו את הפונקציה:

```
lazyTreeMap : ('a -> 'b) * 'a lazyTree -> 'b lazyTree
```

אשר מקבלת פונקציה ועץ עצל (סופי או אינסופי) ומחזירה עץ עצל לאחר שהפעילו את הפונקציה על כל איברי העץ.

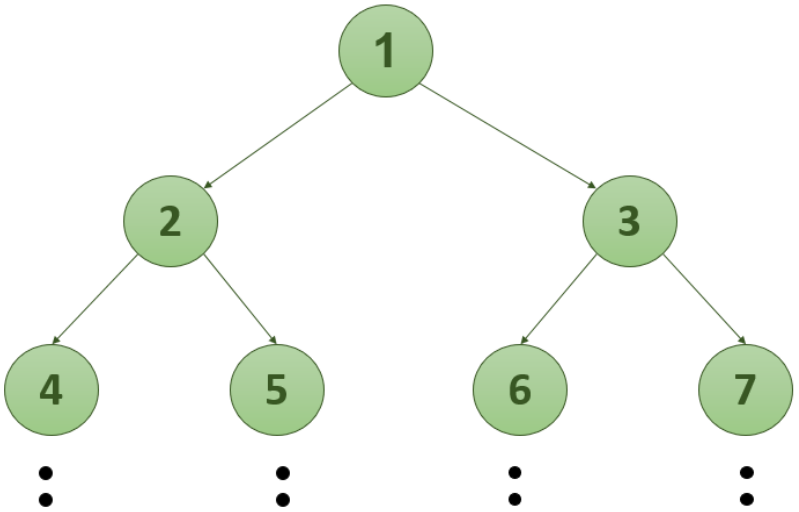
3. ממשו את הפונקציה:

```
lazyTreeFilter : ('a -> bool) * 'a lazyTree -> 'a lazyTree
```

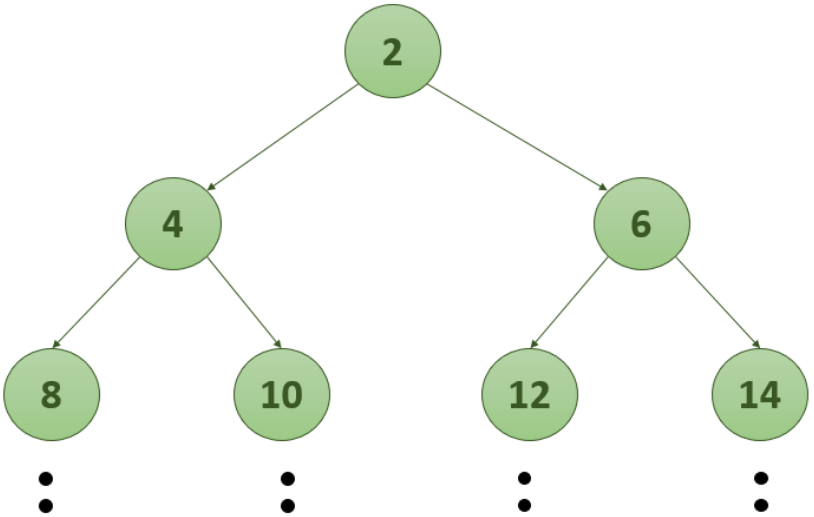
אשר מקבלת פרדיקט ועץ עצל (סופי או אינסופי) ומחזירה עץ עצל אותו בונה באופן הבא:
הפונקציה עוברת על איברי העץ (באופן עצל) ולכל צומת שמספק את הפרדיקט משאירה אותו.
במקרה והצומת אינו מספק את הפרדיקט, מסירה את כל תת העץ (כולל את הצומת המדובר).

דוגמת הרצה:

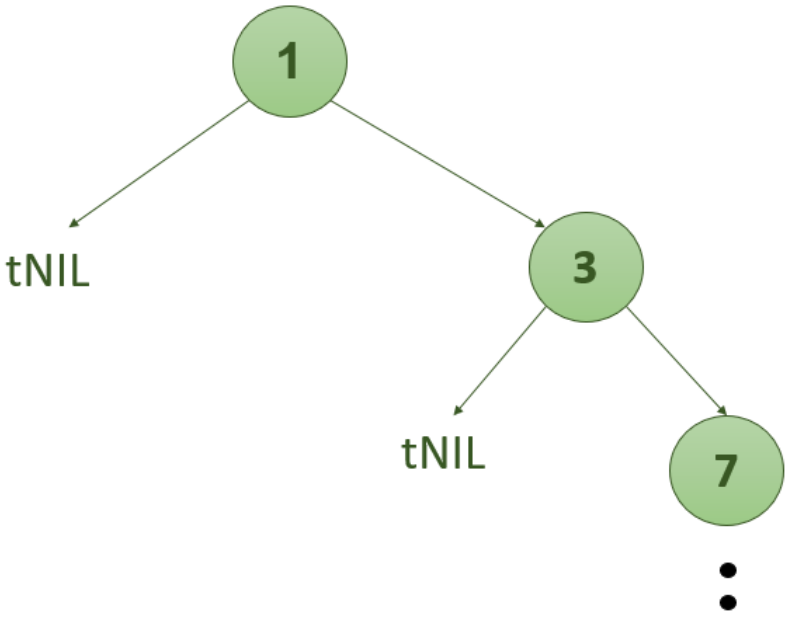
```
- lazyTreeFrom(1);
```




```
- lazyTreeMap ((fn(x)=>2*x),lazyTreeFrom(1));
```



```
- lazyTreeFilter ((fn(x)=>((x mod 2)=1)),lazyTreeFrom(1));
```



משימה 3: MLisp

את פתרון שאלה זו עליכם להגיש בקובץ `mlisp_ex5.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

הגענו לחלקו האחרון של מימוש המפרש של שפת MLisp. בחלק זה נרצה לממש את הפונקציה eval כך שהיא שתקבל סביבה וS-Expression ותחזיר לנו:

- את השערוך של הביטוי תחת הסביבה.
- את הסביבה החדשה לאחר פעולת השערוך.

בעזרת הפונקציה הזו יחד עם הפונקציה parse שמימשנו בתרגילי בית הקודמים ניתן כעת לממש את הפונקציה REPL ולקבל מפרש אינטראקטיבי מלא לשפת MLisp. אנחנו נסתפק בתרגיל זה רק במימוש תקין של פונקציית eval ולא נממש את הפונקציה REPL (אתם כמובן יכולים בזמנכם החופשי ללמוד כיצד לקבל קלט מהמשתמש ולהדפיס פלט לקונסול בשפת ML ולכתוב מימוש מלא לREPL).

הפונקציה eval

הגדירו את החריגה הבאה בתוכנית שלכם:

```
exception MlispError;
```

חריגה זו תשמש אתכם **בכל** סוגי השגיאות. **אסור** לכם לזרוק כל חריגה אחרת.

הגדירו את הפונקציה eval שמקבלת ביטוי מטיפוס SExp וסביבה מקוננת (כפי שהגדרנו בתרגילים הקודמים) והיא מחזירה טאפל של שיערוך הביטוי מעל הסביבה ואת הסביבה המעודכנת. החתימה שלה:

```
val eval = fn : SExp -> (string -> SExp) list -> SExp *
(string -> SExp) list
```

אופן פעולת הפונקציה מתואר בהמשך התרגיל.

שיערוך ביטויים בשפת MLisp

כפי שציינו בתרגיל בית 4, קיים טיפוס יחיד בשפת MLisp והוא S-Expression. נרצה כעת להגדיר כיצד על הפונקציה eval לשערך את הביטויים הללו.

1. אם הביטוי הוא אטום אזי:
   a. אם הוא NIL או שהוא מצייג מספר אז השיערוך שלו יהיה האטום עצמו.
   b. אם הוא symbol אז השיערוך שלו יהיה השיערוך של הערך המתאים לו בסביבה (בהמשך נגדיר כיצד להגדיר כריכות בסביבה).

2. אם הוא רשימה אזי זו הפעלה של פונקציה:
   a. האיבר הראשון ברשימה הוא שם הפונקציה.
      i. אם הפונקציה הוגדרה ע"י המשתמש יש לשלוף אותה מהסביבה.
      ii. אם הפונקציה הוגדרה ע"י מתכנני השפה (אתם) אין צורך לשמור אותה בסביבה ואתם יכולים להכניסה כחלק ממימוש הפונקציה eval. בהמשך נגדיר את הפונקציות שמוגדרות מראש בשפה.
   b. האיברים הבאים ברשימה הם הפרמטרים האקטואליים.
      i. לפני הפעלת הפונקציה יש להכניס כריכות בין הפרמטרים האקטואליים לבין הפרמטרים הפורמליים בscope חדש כלומר יש להכניס סביבה חדשה למחסנית הסביבות.
   c. לאחר מכן יש לשערך את הביטוי המגדיר את הפונקציה על מחסנית הסביבות המעודכנת.
   d. בסוף יש להוציא את הסביבה החדשה ממחסנית הסביבות, כלומר, להסיר את הכריכות בין הפרמטרים הפורמליים לפרמטרים האקטואליים של הפונקציה.
3. כל דבר אחר הוא שגיאה ועליכם לזרוק עליו `MlispError`.

Fear Not, אנחנו נציג שלל דוגמאות הרצה בהמשך.

הפונקציות המוגדרות מראש בMLisp

1. הפונקציה +

מחברת שני מספרים.

דוגמת הרצה:

```
- val (res,env) = eval (parse (tokenize "(+ 2 3)"))
(emptyNestedEnv ());
val res = ATOM (NUMBER 5) : SExp
val env = [fn] : (string -> SExp) list
```

2. הפונקציה -

מחסרת שני מספרים.

דוגמת הרצה:

```
- val (res,env) = eval (parse (tokenize "(- 2 3)"))
(emptyNestedEnv ());
val res = ATOM (NUMBER ~1) : SExp
val env = [fn] : (string -> SExp) list
```

3. הפונקציה *

מכפילה שני מספרים.

דוגמת הרצה:

```
- val (res,env) = eval (parse (tokenize "(* 2 3)"))
(emptyNestedEnv ());
val res = ATOM (NUMBER 6) : SExp
val env = [fn] : (string -> SExp) list
```

4. הפונקציה div

מחלקת שני מספרים.

דוגמת הרצה:

```
- val (res,env) = eval (parse (tokenize "(div 10 2)"))
(emptyNestedEnv ());
val res = ATOM (NUMBER 5) : SExp
val env = [fn] : (string -> SExp) list
```

5. הפונקציה cons

יוצר רשימה בMLisp, מקבל שני פרמטרים: הראשון ראש הרשימה והשני זנב הרשימה.
הפונקציה משוערכת לרשימה המתאימה.

דוגמאות הרצה:

```
- val (res,env) = eval (parse (tokenize "(cons 1 2)"))
(emptyNestedEnv ());
val res = CONS (ATOM (NUMBER 1),ATOM (NUMBER 2)) : SExp
val env = [fn] : (string -> SExp) list
- val (res,env) = eval (parse (tokenize "(cons 1 (cons 2
3))")) (emptyNestedEnv ());
val res = CONS (ATOM (NUMBER 1),CONS (ATOM (NUMBER 2),ATOM
(NUMBER 3))) : SExp
val env = [fn] : (string -> SExp) list
- val (res,env) = eval (parse (tokenize "(cons 1 (cons 2 (cons
3 nil)))")) (emptyNestedEnv ());
val res =
  CONS
    (ATOM (NUMBER 1),CONS (ATOM (NUMBER 2),CONS (ATOM (NUMBER
3),ATOM NIL)))
  : SExp
val env = [fn] : (string -> SExp) list
```

שימו לב! בMLisp יש שני סוגי רשימות:

1. רשימה תקינה: כזו שמסתיימת בnil.

2. רשימה לא תקינה: כזו שלא מסתיימת בnil.

ניתן ליצור את שניהם בשפה כפי שמצויין בדוגמאות.

6. הפונקציה car

מחזירה את ראש הרשימה.

דוגמת הרצה:

```
- val (res,env) = eval (parse (tokenize "(car (cons 1 (cons 2
nil)))")) (emptyNestedEnv ());
val res = ATOM (NUMBER 1) : SExp
val env = [fn] : (string -> SExp) list
```

7. הפונקציה cdr

מחזירה את זנב הרשימה.

דוגמת הרצה:

```
- val (res,env) = eval (parse (tokenize "(cdr (cons 1 (cons 2
nil)))")) (emptyNestedEnv ());
val res = CONS (ATOM (NUMBER 2),ATOM NIL) : SExp
val env = [fn] : (string -> SExp) list
```

8. הפונקציה define

פונקציה זו מגדירה כריכה בין שם לערך או מגדירה כריכה בין שם לפונקציה (שזה בעצם אותו דבר. פונקציה זה גם ערך בMLisp). הפלט של הפונקציה הוא nil אבל היא מחזירה סביבה חדשה שבה מוגדרת הכריכה.

בהגדרת כריכה בין שם לערך שאינו פונקציה define מקבלת שני פרמטרים:

1. שם הערך.
2. הערך עצמו.

הבהרה:בהגדרת כריכה בין שם לערך שאינו פונקציה, אנחנו נבדוק בתרגיל רק מקרים של define בקבוע מספרי שלם. אם כבר מימשתם ותמתכם ביותר מזה, מעולה ואתם יכולים להגיש בלי שינוי. אבל הטסטים יכללו רק מקרים שבהם עושים define למספר שלם.

בהגדרת כריכה בין שם לפונקציה define מקבלת שלושה פרמטרים:

3. שם הפונקציה.
4. רשימת הפרמטרים הפורמליים.
5. הביטוי המייצג את הפונקציה (יכול להכיל כמובן אזכור לפרמטרים הפורמליים).

דוגמת הרצה:

```
- val (res,env) = (eval (parse (tokenize "(define pi 3)"))
env);
val res = ATOM NIL : SExp
val env = [fn] : (string -> SExp) list
- val (res,env) = (eval (parse (tokenize "(define area (r) (*
pi (* r r)))")) env);
val res = ATOM NIL : SExp
val env = [fn] : (string -> SExp) list
- val (res,env) = (eval (parse (tokenize "(area 1)")) env);
val res = ATOM (NUMBER 3) : SExp
val env = [fn] : (string -> SExp) list
- val (res,env) = (eval (parse (tokenize "(area 2)")) env);
val res = ATOM (NUMBER 12) : SExp
val env = [fn] : (string -> SExp) list
```

הסבר:

הקוד

```
(define area (r) (* pi (* r r)))
```

מכיל הגדרה של הפונקציה area. היא מקבלת פרמטר אחד בשם r. ומחשבת ביטוי כלשהו שיכול להכיל את r. בהפעלת הפונקציה:

```
(area 1)
```

נוצרת כריכה בין r לערך 1 ואז משוערך הביטוי.

שימו לב שבניגוד לML אם מישהו ישנה את ערכו של pi, כלומר יגדיר אותו מחדש אזי הקישור הזה ישתנה בהפעלות הבאות של הפונקציה. מה זה אומר על שפת MLisp?

דוגמת הרצה:

```
- val (res,env) = (eval (parse (tokenize "(define pi 3)"))
env);
val res = ATOM NIL : SExp
val env = [fn] : (string -> SExp) list
- val (res,env) = (eval (parse (tokenize "(define area (r) (*
pi (* r r)))")) env);
val res = ATOM NIL : SExp
val env = [fn] : (string -> SExp) list
- val (res,env) = (eval (parse (tokenize "(define pi 7)"))
env);
val res = ATOM NIL : SExp
val env = [fn] : (string -> SExp) list
- val (res,env) = (eval (parse (tokenize "(area 1)")) env);
val res = ATOM (NUMBER 7) : SExp
val env = [fn] : (string -> SExp) list
```

הנחיות

- בבתרגיל זה ניתן להשתמש בכל החומר שנלמד בתרגולים. כרגיל אין להשתמש באף פונקציה או תכונה של השפה שלא נלמדה בתרגולים.
- רשימת הקבצים שצריכים להופיע בתוך קובץ ה-zip היא:
  `dry.pdf,hw5.sml,mlisp_ex5.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 EX5_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:

- 30.12 : תיקונים קלים בחתימות ודוגמאות ההרצה של משימה 1 בחלק הרטוב.
- 3.1: 1. הוספת הנחה על הפרמטרים בסעיף 2 ב משימה 1 בחלק הרטוב.
  2. הוספת הערה בתחילת משימה 2 בחלק הרטוב.
  3. הקלה לגבי define במשימה 3 בחלק הרטוב.
- 4.1: 1. תקיון ההבהרה לגבי define במשימה 3 בחלק הרטוב.
  2. הוספת דוגמת הרצה במשימה 2 החלק הרטוב.