שעת קבלה – שפות תכנות מועד ב'

DISCLAIMER

אלה לא פתרונות רשמיים של הקורס וייתכנו בהם טעויות ואי דיוקים
ייתכנו פתרונות טובים או יעילים יותר

אביב 2020 מועד א' חלק ב' – שאלה

1

בשאלה זו נרחיב את המפרש MLISP שפיתחתם בתרגיל הבית, ונוסיף SExp חדש עבור הפונקציה map שפעולתה דומה לזו שבשפת ML. אולם, היות שבמפרש שבנינו עבור MLISP אין תמיכה בפונקציות אנונימיות נוכל להעביר כפרמטר רק את שם הפונקציה שאותה נרצה להפעיל על הרשימה.

שימו לב כי בתרגיל הבית לא הוגדר כיצד יש לממש הגדרה והפעלה של פונקציה ע"י המפרש, ועל כן עליכם לכתוב קוד שלא מסתמך על מימוש ספציפי. רמז: השתמשו במבנה הרקורסיבי של הרשימה ובהפעלה של שהפונקציה eval על מבנה זה.

המבנה של הפונקציה map:

```
(map fname list)
```

הפונקציה תקבל שם של פונקציה המקבלת ארגומנט אחד ורשימה ותחזיר רשימה שהינה הפעלה של הפונקציה על כל אחד מאיברי הרשימה בהתאמה. ניתן להניח את תקינות הארגומנטים לפונקציה.

לדוגמא, נניח כי הרצנו את השורות הבאות במפרש:

```
val (res,env) = (eval (parse (tokenize "(define pi 3)")) env);
val (res,env) = (eval (parse (tokenize "(define area (r) (* pi (* r r)))")) env);
```

ההצהרה

```
val (res,env) = (eval (parse (tokenize "(map area (1 2 3))")) env);
```

תדפיס

```
val res =CONS(ATOM (NUMBER 3),CONS (ATOM (NUMBER 12),CONS (ATOM (NUMBER 27),ATOM NIL))) : SExp
val env = [fn] : (string -> SExp) list
```

אביב 2020 מועד א' חלק ב' – שאלה 1

<u>תזכורת:</u>

החתימות של env ושל eval הן:

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

המבנה אותו SExp מייצג:

- 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)))))
```

אביב 2020 מועד א' חלק ב' – שאלה 1

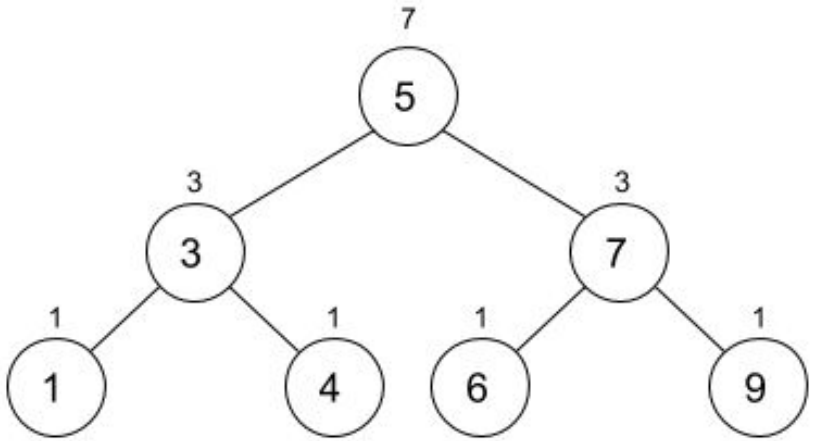
```
fun eval exp env = (case exp of
    ATOM(atom) => (case atom of
        NIL => (ATOM(NIL), env)
    |   NUMBER(number) => (ATOM(NUMBER(number)), env)
    |   SYMBOL("nil") => (ATOM(NIL), env)
    |   SYMBOL("true") => (ATOM(SYMBOL("true")), env)
    |   SYMBOL("false") => (ATOM(SYMBOL("false")), env)
    |   SYMBOL(symbol) => eval (find symbol env) env)
|
(* ... some cases omitted ...*)

|      CONS(
         ATOM (SYMBOL "map"),
         CONS(ATOM (SYMBOL(functionName)),CONS(lst,ATOM(NIL)))
       ) =>

(case lst of
     ATOM(NIL) => (ATOM(NIL),env)
|    CONS(x,xs) => let
          val (first, _) = (eval (CONS(ATOM(SYMBOL(functionName)),CONS(x,ATOM(NIL)))) env)
          val map_exp = CONS(ATOM (SYMBOL "map"),CONS(ATOM(SYMBOL(functionName)),CONS(xs,ATOM(NIL))))     val
(rest, _) = (eval map_exp env)
     in
          (CONS(first,rest),env)
     end
|    _ => raise MlispError)
```

אביב 2020 מועד א' חלק ב' – שאלה 2

עץ דרגות (rank tree) הוא עץ חיפוש בינארי ממוין (אך לא בהכרח מאוזן) בו שמורים בכל צומת זוג ערכים:

1. איבר מטיפוס כלשהו,
2. מספר הצמתים בתת העץ שהצומת הוא שורשו.

לדוגמא, בתרשים מוצג עץ דרגות בו האיברים הם ערכים מסוג int הרשומים בתוך העיגולים (שמציינים צמתים), ומספר הצמתים בכל תת העץ רשום מעל העיגול.



אביב 2020 מועד א' חלק ב' – שאלה 2

<u>סעיף א':</u>

נעזר בהגדרות הבאות:

```
type 'a Node = {Element: 'a, Count: int};
datatype 'a RankTree = Nil | Tree of {Left:'a RankTree, Node:'a Node, Right:'a RankTree};
```

הגדירו את הפונקציה

```
val insert = fn : ('a * 'a -> order) -> 'a RankTree -> 'a list -> 'a RankTree
```

המקבלת פונקציית השוואה, עץ דרגות ורשימת איברים, ומחזירה עץ דרגות חדש המכיל את האיברים ברשימה.

הנחיה: אין להוסיף איברים שכבר נמצאים בעץ. כלומר, אין להוסיף איבר חדש אם קיים איבר בעץ שפונקציית ההשוואה עם האיבר החדש מחזירה EQUAL. תזכורת:

```
datatype order = LESS | EQUAL | GREATER;
```

אביב 2020 מועד א' חלק ב' – שאלה 2

```
local
fun create_node (elem,count) = {Element=elem, Count=count};

fun insert_item cmp Nil item = Tree({Left=Nil, Node = create_node(item,1), Right=Nil})
|   insert_item cmp (Tree({Left=left, Node={Element=elem, Count=count}, Right=right})) item =
     case cmp(item,elem) of
          EQUAL => Tree({Left=left, Node=create_node(elem,count), Right=right})
     |    LESS => Tree({Left=(insert_item cmp left item), Node=create_node(elem, count+1), Right=right})
     |    GREATER => Tree({Left=left, Node=create_node(elem, count+1), Right=insert_item cmp right item})
in
     fun insert cmp tree items = foldl (fn (item,t) => insert_item cmp t item) tree items

end;
```

אביב 2020 מועד א' חלק ב' – שאלה 2

<u>סעיף ב'</u>

האינדקס (rank) של איבר בעץ הוא מקומו בסדרה ממוינת של איברי העץ. לדוגמא, בעץ שבאיור הסדר של האיברים הממוינים הינו 1,3,4,5,6,7,9, והאינדקס של האיבר 7 הינו 6.

נגדיר חריגה.

```
exception NotFound;
```

הגדירו את הפונקציה

```
val rank = fn : ('a * 'b -> order) -> 'b RankTree -> 'a -> int
```

המקבלת פונקציית השוואה, עץ דרגות ואיבר, במידה והאיבר נמצא בעץ הפונקציה מחזירה את האינדקס שלו בעץ אחרת זורקת את החריגה `NotFound`.

הנחיה: בכדי לחשב את האינדקס של איבר בעץ יש לחפש את הצומת של האיבר בעץ (inorder), ולספור הן את הצמתים על מסלול החיפוש בעץ עם איברים הקטנים מאיבר זה והן את הצמתים בתת העץ השמאלי של הצומת.

אביב 2020 מועד א' חלק ב' – שאלה 2

```
local
    fun getCount Nil = 0
        |   getCount (Tree({Left=_, Node={Element=_, Count=count}, Right=_})) = count
in
    fun rank cmp Nil item = raise NotFound
        |   rank cmp (Tree({Left=left, Node={Element=elem, Count=count}, Right=right})) item =
        (case cmp(item,elem) of
            EQUAL => 1 + (getCount left)
        |   LESS => rank cmp left item
        |   GREATER => 1 + (getCount left) + (rank cmp right item))
end;
```

אביב 2020 מועד א' חלק ב' – שאלה 2

סעיף ג':

הגדירו את הפונקציה

```
val find = fn : 'a RankTree -> int -> 'a
```

המקבלת עץ דרגות ואינדקס, במידה וקיים איבר עם אינקס זה בעץ הפונקציה מחזירה איבר זה אחרת זורקת את החריגה `NotFound`.

הנחיה: בכדי למצוא את האיבר בעץ המתאים לאינדקס נתון יש לבדוק האם האיבר בעל אינדקס זה נמצא בתת עץ השמאלי ולחפש את האיבר שם רקורסיבית, אחרת לחפש את האיבר בתת העץ הימני כאשר מחסירים מהאינדקס את מספר הצמתים שלא נמצאים בתת העץ הימני.

אביב 2020 מועד א' חלק ב' – שאלה 2

```
local
    fun getCount Nil = 0
    |       getCount (Tree({Left=_, Node={Element=_, Count=count}, Right=_})) = count
in
    fun find Nil rank = raise NotFound
    |   find (Tree({Left=left, Node={Element=e, Count=count}, Right=right})) rank =
        let
            val right_rank = getCount right
            val left_rank = getCount left
        in
            if ((count - right_rank) = rank) then e
                else if left_rank >= rank then (find left rank)
                else find right (rank - (count - right_rank))
        end
end;
```

אביב 2020 מועד א' חלק ב' – שאלה 2

סעיף ד':

מדוע הטיפוס של פונקציית ההשוואה בפונקציה rank בסעיף ב' הוא a' * 'b -> order ואילו בפונקציה insert בסעיף א' הוא a' * 'a -> order?

אביב 2020 מועד א' חלק ב' – שאלה

3

בעקבות התפרצות הגל השני של מגפת הקורונה בארץ, משרד הבריאות מבקש את עזרתכם במציאת כל האנשים במדינה שצריכים להימצא בבידוד נכון להיום.
לצורך כך משרד הבריאות מספק לנו את רשומות המידע שבידיו, כאשר לכל אדם שנדבק בנגיף קיימת רשומה יחידה עם התאריך (יום, חודש ושנה) שבו האדם ששמו `name` נדבק בנגיף:

```
sick(name, date(Day, Month, Year)).
```

השב"כ מספק לנו רשומה עם התאריך הנוכחי:

```
today(date(Day, Month, Year)).
```

למשל,

```
today(date(16, 7, 2020)).
```

נתונות גם רשומות עם מספר הימים בכל חודש (ניתן להתעלם משנה מעוברת):

```
mdays(1,31)
mdays(2,28)
mdays(3,31)
mdays(4,30)
mdays(5,31)
mdays(6,30)
mdays(7,31)
mdays(8,31)
mdays(9,30)
mdays(10,31)
mdays(11,30)
mdays(12,31)
```

אביב 2020 מועד א' חלק ב' – שאלה 3

<u>סעיף א':</u>

משרד הבריאות מגדיר כל אדם שטרם חלפו שבועיים מאז היום בו חלה כאדם שצריך להיכנס לבידוד. ממשו את הרלציה bidud(X) המסתפקת עבור קלט X (קונקרטי או לא מאותחל) אם ורק אם על X להימצא בבידוד.

לדוגמה, עבור הרשומות הבאות:

```
sick(noa, date(10, 7, 2020)).
sick(alex, date(7, 7, 2020)).
sick(raviv, date(2, 7, 2020)).
sick(daniel, date(3, 6, 2019)).
sick(tomer, date(29, 12, 2019)).
sick(itay, date(30, 12, 2019)).
```

והקריאה:

```
today(date(16, 7, 2020)).
bidud(X(.
```

יתקבל הפלט הבא:

```
X=noa ; X=alex ; false.
```

אביב 2020 מועד א' חלק ב' – שאלה 3

```
within_2_weeks(date(D1,M,Y), date(D2,M,Y)) :- D1 #>= D2, D1 - D2 #< 14.
within_2_weeks(date(D1,1,Y1), date(D2,12,Y2)) :- Y1 #= Y2 + 1,
                        mdays(12,D3),
                        (D1 + (D3 - D2)) #< 14.
within_2_weeks(date(D1,M1,Y), date(D2,M2,Y)) :- M1 #= M2 + 1,
                            mdays(M2,D3),
                            (D1 + (D3 - D2)) #< 14.

bidud(X) :- today(D1), sick(X, D2), within_2_weeks(D1,D2).
```

אביב 2020 מועד א' חלק ב' – שאלה 3

סעיף ב':

כעת משרד הבריאות גילה שאנשים גם נפגשים! הוא רוצה לכלול ברשימת האנשים החייבים בבידוד את כל מי:

- שטרם חלפו שבועיים מאז היום בו חלה, או
- ששני התנאים הבאים נכונים לגביו:
  - פגש אדם שנדבק בקורונה בשבועיים (14 ימים) הצמודים ליום הדבקת האדם.
  - חלפו לכל היותר שבועיים מיום הפגישה.

לשם כך הוא מוסר לידיכם רשומות שאסף השב"כ המתארות מפגש של שני אנשים בתאריך מסוים:

```
meeting(name1, name2, date(Day,Month,Year)).
```

עליכם לממש את shabak(X) (שוב X, קונקרטי או לא מאותחל) כדי למצוא את כל האנשים שצריכים להיות בבידוד נכון להיום לפי המדיניות החדשה של משרד הבריאות.

לדוגמה, עבור הרשומות הבאות:

```
meeting(noa, alex, date(9, 7, 2020)).
meeting(alex, michal, date(10, 7, 2020)).
meeting(noa, guy, date(8, 3, 1920)).
meeting(noa, guy, date(8, 7, 2020)).
meeting(raviv, itay, date(6, 1, 2020)).
meeting(noa, raviv, date(11, 7, 2020)).
meeting(guy, dimion, date(15, 7, 2020)).
meeting(david, dimion, date(15, 7, 2020)).
meeting(alex, noa, date(15, 7, 2020)).

today(date(16, 7, 2020)). shabak(X).
```

והקריאה:

יתקבל הפלט הבא:

```
X = noa;  X = alex ;X = guy ;X = raviv ;X = michal ;false
```

אביב 2020 מועד א' חלק ב' – שאלה 3

```
sym_meeting(X,Y,D) :- meeting(X,Y,D).
sym_meeting(X,Y,D) :- meeting(Y,X,D).

shabak(X) :- bidud(X).
shabak(X) :- today(D1),
             sym_meeting(X,Y,D2),
             within_2_weeks(D1,D2),
             sick(Y,D3),
             (within_2_weeks(D3,D2) ; within_2_weeks(D2,D3)),
             not(bidud(X)).
```

אביב 2020 מועד א' חלק ב' – שאלה 3

<u>סעיף ג':</u>

משרד הבריאות מבקש להרחיב את מעגל הבידוד כך שיכלול גם מעגל שני ומעלה של הדבקות, כלומר, את כל אדם שלגביו מקיים לפחות אחד משלושת התנאים:

- טרם חלפו שבועיים מאז היום בו חלה
- פגש אדם שנדבק בקורונה בשבועיים הצמודים ליום הדבקת האדם, וטרם חלפו שבועיים מיום הפגישה.
- פגש אדם שחייב בבידוד, וטרם חלפו שבועיים מיום הפגישה

עליכם לממש את doomed(X) שתחזיר את שמות כל אלו שעליהם להיכנס לבידוד כעת! (כאשר שוב X, יכול להיות קונקרטי או משתנה).

לדוגמה, עבור אותן רשומות כמו בשני הסעיפים הקודמים, והקריאה:

```
today(date(16, 7, 2020)).
doomed(X).
```

כל צוות הקורס, לדאבוננו, יחויב לשהות בבידוד:

```
X = noa;  X = alex ;X = guy ;X = raviv ;X = michal ;X = dimion; X = david; false.
```

אביב 2020 מועד א' חלק ב' – שאלה 3

```
doomed(X) :- shabak(X).
doomed(X) :- sym_meeting(Y,X,D2),
             today(D1),
             within_2_weeks(D1,D2),
             doomed(Y).
```

חורף 2019-2020 מועד ב' – שאלה 2

בשאלה זו, עליך להשתמש בדקדוק של ליספ, ובפונקציות defun, eq, cond, cons, cdr, car ו atom בלבד. אין להשתמש במספרים בליספ, אלא ברשימות ובביטויי S בלבד.

א. כתוב את הפונקציה SNOC אשר מקבלת שני פרמטרים: ביטוי S אשר שמו יהיה X ורשימה אשר שמה יהיה XS ומחזירה את הרשימה המתקבלת מהוספת X בסוף הרשימה XS. סדר הפרמטרים ל- SNOC יהיה הפוך מזה של הפונקציה CONS. ניתן, אך לא חובה, להגדיר פונקציות עזר.

```
(defun SNOC (XS X) (cond ((eq XS nil) (cons X nil))
                         (t (cons (car XS) (SNOC (cdr XS) X)))))
```

ב. כתוב את הפונקציה LISTABLE אשר מקבלת ביטוי S ומחזירה את הערך המקובל בליספ ל-TRUE אם הביטוי ניתן להכתב בכתיב הרשימות (כלומר כרשימה של אטומים או של רשימות), ואת הערך המקובל בליספ ל FALSE אם הביטוי אינו יכול להכתב בכתיב זה.

```
(defun LISTABLE (S) (cond ((eq S nil) t)
                          ((atom S) nil)
                          (t (LISTABLE (cdr S)))))
```

אביב 2019 מועד ב' – שאלה 3 סעיף ג'

סעיף ג'

עבור כל זוג הגדרות של הטיפוסים σ ו- τ סמנו את התשובה הנכונה ביותר, כך שתוכנית שעברה בהצלחה את בדיקת הטיפוסים לעולם לא תנסה לגשת בזמן ריצה לשדה שלא קיים ברשומה.

(א) מותר ש- σ יהיה subtype של τ
(ב) מותר ש- σ יהיה subtype של τ רק בשפה האוסרת selective update
(ג) מותר ש- σ יהיה subtype של τ רק בשפה המתירה selective update
(ד) אסור ש- σ יהיה subtype של τ

type σ	type τ	תשובה
$\sigma = \{\ell_1: \text{int}, \ell_2: \text{int}\}$	$\tau = \{\ell_1: \text{int}, \ell_2: \text{int}, \ell_3: \text{int}\}$	(א) (ב) (ג) (ד)
$\sigma = \{\ell_1: \text{int}, \ell_2: \{\ell_3: \text{int}, \ell_4: \text{int}\}\}$	$\tau = \{\ell_1: \text{int}, \ell_2: \{\ell_3: \text{int}\}\}$	(א) (ב) (ג) (ד)
$\sigma = \{\ell_1: \text{int}, \ell_2: \text{int}, \ell_3: \text{int}\}$	$\tau = \{\ell_2: \text{int}, \ell_3: \text{int}\}$	(א) (ב) (ג) (ד)
$\sigma = \text{fn}: \{\ell_1: \text{int}, \ell_2: \text{int}, \ell_3: \text{int}\} \to \text{int}$	$\tau = \text{fn}: \{\ell_2: \text{int}, \ell_3: \text{int}\} \to \text{int}$	(א) (ב) (ג) (ד)
$\sigma = \text{fn}: \text{int} \to \{\ell_1: \text{int}, \ell_2: \{\ell_3: \text{int}\}\}$	$\tau = \text{fn}: \text{int} \to \{\ell_1: \text{int}, \ell_2: \{\ell_3: \text{int}, \ell_4: \text{int}\}\}$	(א) (ב) (ג) (ד)

אביב 2018 מועד א' – שאלה 3

סעיף ב'

עבור כל אחת מההכרזות הבאות רישמו אילו **כריכות** (binding) חדשות נוצרות בזמן ההכרזה. לכל כריכה (binding), יש לציין את **השם** (name), את **הטיפוס** (type), ואת **היישות/ערך** (value\entity) שנקשר/ה.

```
fun ++ n = (n := !n + 1; n);
```

ישנה כריכה בין השם ++ לפונקציה המוגדרת שהינה מטיפוס int ref -> int ref

---

```
val f = let val x = ref 42 in fn () => ++ x end;
```

תחילה ישנה כריכה בסביבה פנימית של הפונקציה לבין השם x לערך ref 42 שהינו מטיפוס int ref.

לאחר מכן ישנה כריכה בין השם f לפונקציה המוגדרת שהינה מטיפוס unit->int ref.

אביב 2018 מועד א' – שאלה 3

סעיף ב' – המשך

```
val g = fn () => let val x = ref 42 in ++ x end;
```

ישנה כריכה בין השם g לפונקציה המוגדרת שהינה מטיפוס unit->int ref.

נשים לב כי הכריכה עבור השם x אינה מתרחשת בזמן ההכרזה!

```
local val x = ref 42 in fun h () = ++ x end;
```

נוצרת כריכה בסביבה הפנימית של ה-local בין השם x לערך ref 42 מטיפוס int ref, וכן נוצרת כריכה בין השם h לבין הפונקציה המוגדרת מטיפוס unit->int ref.

אביב 2018 מועד א' – שאלה 3

סעיף ג'

להלן הכרזות נוספות. רשמו אילו כריכות (bindings) חדשות התווספו לסביבה **לאחר סיום ביצוע כל ההכרזות הללו**. לכל כריכה (binding), יש לציין את השם (name), את הטיפוס (type), ואת הישות/ערך (value/entity)שנקשר/ה.

```
val (r,s,t) = (f(),g(),h())
val ((),(),()) = (r:=0,s:=1,t:=2)
val (u,v,w) = (f(),g(),h())
val (x,y,z) = (f(),g(),h());
```

אביב 2018 מועד א' – שאלה 3

סעיף ג' – המשך

נרשום אילו כריכות יש לנו לאחר כל שורה,

```
val (r,s,t) = (f(),g(),h())
val r = ref 43, val s = ref 43, val t = ref 43
val ((),(),()) = (r:=0,s:=1,t:=2)
val r = ref 0, val s = ref 1, val t = ref 2
val (u,v,w) = (f(),g(),h())
val r = ref 1, val s = ref 1, val t = ref 3
val u = ref 1, val v = ref 43, val w = ref 3
val (x,y,z) = (f(),g(),h());
val r = ref 2, val s = ref 1, val t = ref 4
val u = ref 2, val v = ref 43, val w = ref 4
val x = ref 2, val y = ref 43, val z = ref 4
```

תזכורת:

```
fun ++ n = (n := !n + 1; n);
val f = let val x = ref 42 in fn () => ++ x end;
val g = fn () => let val x = ref 42 in ++ x end;
local val x = ref 42 in fun h () = ++ x end;
```

אביב 2018 מועד א' – שאלה 3

סעיף ד'

לנוכח ההכרזות בסעיפים הקודמים, מהו הטיפוס (type) ומה הערך (value) של הביטוי הבא:

```
[!r, !(f()), !t, !r, !s, !(f()) + !(f())];
 Val it = [2, 3, 4, 3, 1, 9] : int list
```

מה היה הטיפוס ומה היה הערך של הביטוי הנ"ל לו היינו מריצים את הקוד מההתחלה בגרסה היפותטית שלML עם תחימה דינאמית (dynamic scoping)? סמנו את התשובה הנכונה ונמקו:

(א) אותו טיפוס ואותו ערך

(ב) אותו טיפוס אך ערך שונה

(ג) שגיאת טיפוס

(ד) שגיאת ריצה

נימוק: _____

כאשר נקרא לפונקציה h לא מוגדר בסביבה שלה ישות בשם x.

אביב 2017 מועד ב' – שאלה

(4) נתון הקוד הבא בשפת ML.

```
val n = 3;
fun foo x =
  let
    fun foo x = x + n
    val n = foo x
  in
    foo x
  end;
```

ונגדיר ארבע גירסאות היפותטיות של ML:

(a) תחימה לקסיקלית (Lexical scoping) ושערוך Applicative order

(b) תחימה דינמית (Dynamic scoping) ושערוך Applicative order

(c) תחימה לקסיקלית (Lexical scoping) ושערוך Normal order

(d) תחימה דינמית (Dynamic scoping) ושערוך Normal order

מה יהיה ערך הביטוי `foo n` בכל אחד מהגרסות הנ"ל של ML? נא למלא את התשובות בטבלה.

`foo n`	Lexical scoping	Dynamic scoping
Applicative order	(a) **6**	(b) **9**
Normal order	(c) **6**	(d) **12**

אביב 2017 מועד ב' – שאלה

(5) נתון הקוד הבא בשפת ML.

```
val t = ref 0;
fun ++ _ = (t := !t + 1; t);
val t = ref 1;
fun bar t = if !t * !t < 5 then !t else !(++ t);
```

מה יהיה ערך הביטוי `bar(++(++ t))` בכל אחד מהגרסות הנ"ל של ML? נא למלא את התשובות בטבלה.

`bar(++(++ t))`	Lexical scoping	Dynamic scoping
Applicative order	(a) **2**	(b) **4**
Normal order	(c) **3**	(d) **4**

תרגיל בית 6 - פרולוג

ג. כתבו פרדיקט בשם transpose מהצורה:

```
transpose_pl(A, B)
```

כאשר לפחות אחד משני הארגומנטים קונקרטי.

הפרדיקט מקבל מטריצות בפורמט של רשימה של רשימות, כאשר כל רשימה פנימית מהווה שורה במטריצה.

הפרדיקט מסתפק כאשר B טרנפוז של A כלומר: $A^T = B$.

דוגמא:

```
?- transpose_pl([[1,2,3],[4,5,6],[7,8,9]], Ts).
Ts = [[1, 4, 7], [2, 5, 8], [3, 6, 9]].
```

תרגיל בית 6 - פרולוג

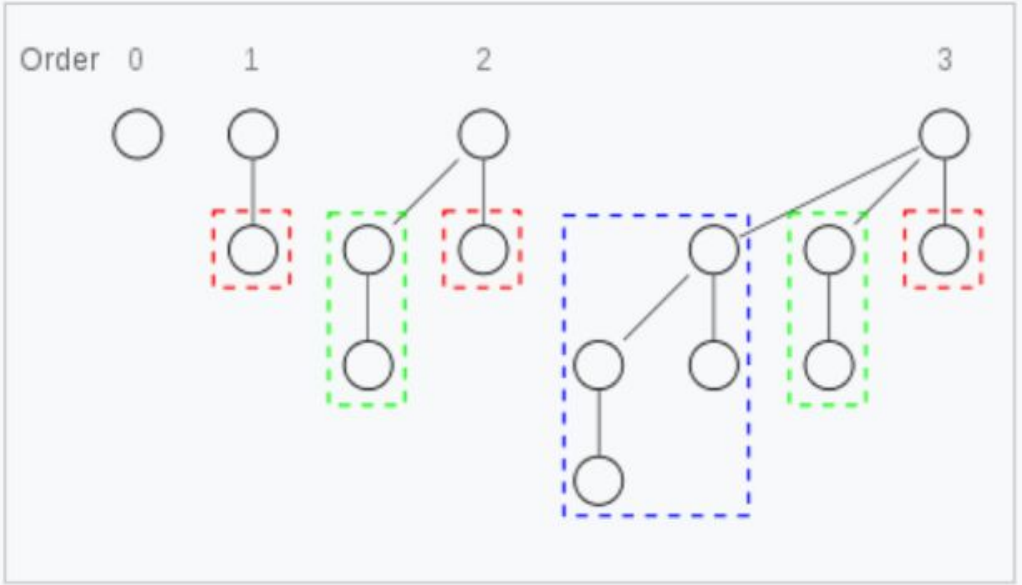
```
transpose_first_row([],[],[]).
transpose_first_row([[X|Xs]|Rest1],[X|Rest2],[Xs|Rest3]) :-
transpose_first_row(Rest1,Rest2,Rest3).

transpose_pl([[]|_],[]).
transpose_pl(Mat,[R|Rs]) :- transpose_first_row(Mat,R,Mat1), transpose_pl(Mat1,Rs).
```

אביב 2019 מועד ב' – שאלה 4

שאלה 4 (15 נק')

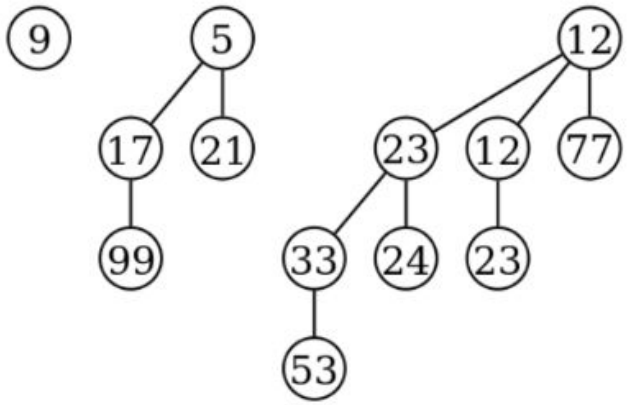
תזכורת

עץ בינומי (binomial tree) הוא עץ בעל שורש, המארגן 2^n צמתים לתת-עצים שכולם בינומיים בעצמם. באופן רקורסיבי, העץ הבינומי מסדר 0, המסומן B_0, מורכב מצומת אחד בלבד; העץ הבינומי B_n, מסדר n, מתקבל מהוספת השורש של עץ בינומי B_{n-1} כצאצא נוסף של השורש של עץ בינומי B_{n-1} אחר. בפרט, B_1 מורכב משורש עם עלה יחיד; B_2 מורכב משורש שיש לו שני צאצאים: עותק של B_1 משמאל, ועלה בודד מימין; וכן הלאה. לפיכך, עץ בינומי B_n מורכב משורש עם בנים $B_0, B_1, B_2, \ldots, B_{n-1}$. באיור, משמאל לימין: עצים בינומיים מגובה 0 עד 3. לכל עץ יש שורש עם תת-עצים מכל הגבהים הנמוכים ממנו.



אביב 2019 מועד ב' – שאלה

ערימה בינומית (binomial heap[1]) היא אוסף של עצים בינומיים. כל עץ בינומי מקיים את "כלל הערימה": ערך כל צומת קטן יותר מערכי כל בניו. דוגמה לערימה בינומית בגודל 13:



הגדרות

לפניכם הגדרות בשפת ML לייצוג עץ בינומי וערמה בינומית:

```
(*  Binomial tree defined as (node value, list of children trees)  *)
datatype 'a tree = Empty | Node of 'a * ('a tree) list;

(*  Binomial heap is a list of (binomial tree, rank) in ascending rank order  *)
type 'a heap = ('a tree * int) list;
```

אביב 2019 מועד ב' – שאלה 4

סעיף א'

ממשו פעולת **מיזוג**: בהינתן שתי ערימות, ממזגים אותן בדומה לחיבור בינארי: עוברים על הדרגות מהקטנה לגדולה, ובכל דרגה, אם יש רק עץ אחד מדרגה כזו מכניסים אותו לערימה המאוחדת. אם יש שניים, מאחדים אותם ומתייחסים אליהם כאל עץ מדרגה גדולה יותר (בדומה לנשא בחיבור רגיל).

```
val merge = fn : int heap -> int heap -> int heap
```

דוגמת הרצה:

```
- val h1 = [(Node (5,[Node (21,[])]),1)];
- val h2 = [(Node (17,[Node (99,[])]),1)];
- val h3 = merge h1 h2;
val h3 = [
  (Node (5,[Node (17,[Node (99,[])]),Node (21,[])]),2)
] : (int tree * int) list
```

אביב 2019 מועד ב' – שאלה 4

```
fun merge [] h1 = h1
|   merge h2 [] = h2
|   merge ((t1,n1)::xs) ((t2,n2)::ys) = if n1 < n2 then ((t1,n1)::(merge xs ((t2,n2)::ys))) else
                        if n2 < n1 then ((t2,n2)::(merge ((t1,n1)::xs) ys)) else
            let
                val (Node(node1,l1)) = t1
                val (Node(node2,l2)) = t2
                val tree = if node1 < node2 then (Node(node1,t2::l1))
                    else (Node(node2,t1::l2))
            in
                merge (merge [(tree,n1+1)] xs) ys
            end;
```

אביב 2019 מועד ב' – שאלה 4

(עוד פתרון ל merge)

```
fun merge_trees (Node(node1,l1)) (Node(node2,l2)) =
    if node1 < node2 then (Node(node1,(Node(node2,l2))::l1))
    else (Node(node2,(Node(node1,l1))::l2));

fun merge_aux [] = []
|   merge_aux [t] = [t]
|   merge_aux [(t1,n1),(t2,n2)] = (case n1 = n2 of
                    false => [(t1,n1),(t2,n2)]
                |   true => [(merge_trees t1 t2, n1 + 1)])
|   merge_aux ((t1,n1)::(t2,n2)::(t3,n3)::xs) = (case n1 = n2 of
                    false => (t1,n1) :: (merge_aux ((t2,n2)::(t3,n3)::xs))
                |   true =>
            if n2 = n3 then (t1,n1) :: (merge_aux ((t2,n2)::(t3,n3)::xs))
            else merge_aux ((merge_trees t1 t2, n1 + 1) :: (merge_aux ((t3,n3)::xs))));
```

אביב 2019 מועד ב' – שאלה 4

```
fun merge_heaps h1 [] = h1
|   merge_heaps [] h2 = h2
|   merge_heaps ((t1,n1)::xs) ((t2,n2)::ys) =
        if n1 < n2 then ((t1,n1)::(merge_heaps xs ((t2,n2)::ys)))
        else if n2 < n1 then ((t2,n2)::(merge_heaps ((t1,n1)::xs)) xs)
        else ((t1,n1)::(t2,n2)::(merge_heaps xs ys));

fun merge h1 h2 = merge_aux (merge_heaps h1 h2);
```

אביב 2019 מועד ב' – שאלה 4

סעיף ב'

ממשו פעולת **הכנסה:** כדי להכניס אלמנט חדש, נבנה ערימה חדשה מדרגה 0 ונמזג אותה עם הערימה הקיימת.

```
val insert = fn : int heap -> int -> int heap
```

דוגמת הרצה:

```
- val h4 = insert h3 9;
val h4 = [
  (Node (9,[]),0),
  (Node (5,[Node (17,[Node (99,[])]),Node (21,[])]),2)
] : (int tree * int) list
```

```
fun insert h elem = merge h [(Node(elem,[]),0)];
```

אביב 2019 מועד ב' – שאלה

סעיף ג'

ממשו פעולת **מחיקת המינימום**: הואיל וכל אחד משורשי העצים בערימה מקיים את תכונת הערימה, הרי שאיבר המינימום בערימה הוא אחד משורשי העצים. כיוון שילדיו של עץ מדרגה i הם שורשים של עצים מדרגות 0,1,...,i-1, ניתן להסתכל על שורשיו כערימה בינומית מסדר i-1. כעת מבצעים מיזוג בין הערימה החדשה הזו לשאר הערימה.

```
val delmin = fn : int heap -> int heap
```

דוגמת הרצה:

```
- delmin h4;
val h4 = [
  (Node (9,[Node (17,[Node (99,[])]),Node (21,[])]),2)
] : (int tree * int) list
```

אביב 2019 מועד ב' – שאלה 4

```
fun min_lst m [] = m
|   min_lst m (x::xs) = if x < m then (min_lst x xs) else (min_lst m xs);

fun create_heap [] _ = []
|   create_heap (t::ts) n = (create_heap ts (n-1)) @ [(t,n)];

fun delmin_aux _ [] _ = []
|   delmin_aux m (((Node(elem,l)),n)::xs) begin_lst = case m = elem of
          false => delmin_aux m xs (begin_lst @ [((Node(elem,l)),n)])
      |   true => (case (null l) of
                true => xs
             |   false => merge (begin_lst@xs) (create_heap l (n-1)));
```

אביב 2019 מועד ב' – שאלה 4

```
fun delmin [] = []
|   delmin h = let
            val head_lst = map (fn ((Node(n,_),_)) => n) h
            val min = min_lst (hd head_lst) (tl head_lst)
        in
            delmin_aux min h []
        end;
```

חורף 10-11 מועד א' – שאלה 4

נתבונן בקטע הקוד הבא ב-ML:

```
exception E;
fun r(x) = if (x mod 2 = 0) then raise E else x;
fun f(a,[])     = 0
  | f(a,x::xs) = (a*x + f(a,xs)) handle E => x+f(a,xs);
val x = 2;
f(r(x), [3,4]);
```

כיתבו מה יהיה פלט התכנית בכל אחד מהוריאנטים הבאים של ML:

א. ML עם static binding, eager evaluation
uncaught exception E

ב. ML עם static binding, normal-order evaluation
val it = 7 : int

ג. dynamic binding, eager evaluation
uncaught exception E

ד. ML עם dynamic binding, normal-order evaluation
val it = 13 : int

חורף 07-08 מועד א' – שאלה 4

נרצה לייצג מערכת קבצים ע"י טיפוס הנתונים הבא:

```
datatype fselem  =  File of string * string |
                    Folder of string * fselem list;
```

הסבר:

מערכת קבצים מכילה קבצים וספריות, כאשר לקובץ (File) יש שם ותוכן, ולספריה (Folder) יש שם ורשימה של אלמנטים המוכלים בה (קבצים או ספריות).

א. (4 נק') עליכם לממש את הפונקציה

```
val find = fn : string -> fselem -> string list
```

המקבלת שם קובץ וספריה התחלתית, ומחזירה את ה – path המלא לקובץ, כלומר את מסלול הספריות מהספריה ההתחלתית ועד הספריה המכילה את הקובץ. המסלול ייוצג ע"י רשימה המכילה את שמות כל הספריות.

במידה ושם הקובץ לא נמצא יש לזרוק את החריגה `NotFound`.

לדוגמא בעבור הקריאה הבאה:

```
find "f" (Folder("a", [Folder("b", [File("f", "some content...")])]));
```

התוצאה תהיה:

```
["a", "b"]
```

ניתן להניח שישנו לכל היותר קובץ יחיד עם אותו שם תחת עץ הספריות הנתון.

חורף 07-08 מועד א' – שאלה 4

```
fun find f_name (File(name,_)) = if name = f_name then [] else raise NotFound
|   find _ (Folder(_,[])) = raise NotFound
|   find f_name (Folder(name,f_lst)) = name :: (find f_name (hd f_lst))
            handle NotFound => find f_name (Folder(name,(tl f_lst)));
```

חורף 07-08 מועד א' – שאלה 4

ג. (6 נק') כתבו את הפונקציה

```
val find_content = fn : string -> fselem -> string list list
```

המקבלת מחרוזת שמייצגת תוכן של קובץ וספריה התחלתית, ומחפשת את כל הקבצים בעלי התוכן הנ"ל. בשונה מסעיף א', יכול להופיע יותר מקובץ אחד כזה, לכן הפונקציה תחזיר רשימה של מסלולי ספריות. כמו כן, הפעם שם הקובץ אינו ידוע מראש, לכן יש להכניסו כחלק מהמסלול.

כלומר, עבור הקריאה הבאה:

```
find_content "abc" (Folder("a", [File("f1", "abc"), Folder("b", [File("f2", "abc")])]));
```

התוצאה תהיה:

```
[["a","f1"], ["a","b","f2"]]
```

חורף 07-08 מועד א' – שאלה 4

```
local
    fun insertFirst x l = map (fn lst => x::lst) l
in
    fun find_content c (File(name,content)) = if c = content then [[name]] else []
    |       find_content c (Folder(_,[])) = []
    |       find_content c (Folder(name,(f::fs))) =
    (insertFirst name (find_content c f)) @ (find_content c (Folder(name,fs)))
end;
```

חורף 07-08 מועד א' – שאלה 4

ה. (6 נק') כתבו את הפונקציה

```
val scorch = fn : fselem -> fselem
```

המקבלת ספריה התחלתית, ומוחקת את כל הקבצים במבנה הספריות הנתון (אבל משאירה את מבנה הספריות כמו שהוא) ומחזירה את התוצאה.

הניחו כי הספרייה ההתחלתית הינה תיקייה (Folder) ולא קובץ (File).

חורף 07-08 מועד א' – שאלה 4

```
fun scorch (Folder(name,[])) = Folder(name,[])
|   scorch (Folder(name,(File(_,_))::fs)) = scorch (Folder(name,fs))
|   scorch (Folder(name,(f::fs))) =
        let
            val (Folder(_,scorched)) = scorch (Folder(name,fs))
        in
            Folder(name,(scorch f)::scorched)
        end;
```