תאריך הבחינה: 5.7.2017 שעה: 09:00

שם המרצה: מני אדלר,מיכאל אלחדד, מירה בלבן, ירון גונן, דנה פישמן

מבחן בקורס: עקרונות שפות תכנות

מס' קורס: 202-1-2051

מיועד לתלמידי: מדעי המחשב והנדסת תוכנה

שנה: ב' סמסטר: ב'

מועד א'

משך הבחינה: 3 שעות

חומר עזר: אסור.

<u>הנחיות כלליות:</u>

1) ההוראות במבחן מנוסחות בלשון זכר, אך מכוונות לנבחנים ולנבחנות כאחד.
2) מבחן הכתוב בעיפרון חלש המקשה על הקריאה**, לא יבדק**
3) נא לוודא כי ב**שאלון** זה X עמודים
4) יש לענות על כל השאלות <u>בגוף המבחן בלבד (בתוך השאלון)</u>. מומלץ לא לחרוג מהמקום המוקצה.
5) מומלץ להשתמש בטיוטא לפני כתיבת התשובה בתוך השאלון.
6) אם אינך יודע את התשובה, ניתן לכתוב "לא יודע" ולקבל 20% מהניקוד על הסעיף/השאלה.
7) הניקוד במבחן הוא כדלהלן:

שאלה 1: טיפוסים ופונקציות high-order	=	25 נק'
שאלה 2: רשימות עצלות	=	26 נק'
שאלה 3: דחיית חישוב ואופרטורים מיוחדים	=	20 נק'
שאלה 4: מודל הסביבות	=	12 נק'
שאלה 5: תכנות לוגי	=	22 נק'

סה"כ:	=	105 נק'

בהצלחה !!!

שאלה 1 [25 נק'] טיפוסים ופונקציות high-order

מבנה הנתונים Map ב-JavaScript מממש את הממשק הבא:

```
let m = new Map();
m.has(x):    determines whether m contains an entry <x,s(x)>
m.set(x, y): adds an entry <x,y> to the map m, or update <x,s(x)> to
             <x,y> if such an entry exists.
m.get(x):    returns the value of x in m if it exists.
```

א. [3 נק'] השלם את הטיפוסים של הפונקציות בעזרת אנוטציות ב-TypeScript:

```
Map<T1,T2>:

has(key: __________): ________________

set(____________________): __________

get(____________________): __________
```

ב. [4 נק']

התבונן בהגדרה הבאה:

```
const memoize = (fn) => {
  let cache = new Map();
  return (x) => {
    if (!cache.has(x)) {
      cache.set(x, fn(x))
    }
    return cache.get(x)
  }
}

const fib = memoize((n:number) : number => {
    return  (n === 0) ? 0 :
            (n === 1) ? 1 :
            fib(n-1) + fib(n-2);
})
```

הסבר מה הפונקציה memoize מחשבת ומה היתרון שהפעלת memoize מעניקה להגדרת fib:

_____

_____

_____

ג. [5 נק'] השלם את הטיפוסים בהגדרת memoize:

```
const memoize : <T1,T2>_____________________________________________ =

<T1,T2>_____________________________________________  => {

  let cache = new Map_________________();

  return (x:______) _________________ => {
    if (!cache.has(x)) {
      cache.set(x, fn(x))
    }
    return cache.get(x)
  }
}
```

ד. [5 נק']

מגדירים פונקציות higher-order לעבודה מעל maps ב-TypeScript:

- **prop**: given a key, return an accessor function for a map that returns the value of this key.
- **propEq**: given a key and a value, return a predicate which tests whether a map has an entry <key,value>.

למשל:

```
const students = [
    {name: 'John', age: 23, dob: {year: 1994, month: 10, day: 10}},
    {name: 'Mary', age: 24, dob: {year: 1993, month: 07, day: 28}}
];

students.map(prop('age'));
==> [23, 24]

students.filter(propEq('name', 'John'));
==> [{name: 'John', age: 23, dob: {year: 1994, month: 10, day: 10}}]
```

השלם את הטיפוסים של הפונקציות ואת הקוד של propEq:

```
const prop : (key:_____________________________________________ =

    (key: _____________) : ____________________________ => {

        return (map: {}) => map[key];
    }

const propEq : <T>(key: ___________________________________ =

    (key: __________, value: ___________) => {

        ____________________________________________________
    }
```

ה. [8 נק']

מרחיבים את הפונקציה prop לפונקציה שיכולה לעבור על מסלול של מפתחות בתוך ערך מסוג map.

- **path**: Returns a function that when supplied an object returns the nested property of that object, if it exists.

כתוב את הפונקציה path והגדר עם טיפוסים מלאים.

```
students.map(path(['dob', 'year']))

==> [ 1994, 1993 ]
```

תזכורת: כאשר עובדים על מערך ב-JavaScript, ניתן להשתמש ב:

```
const arr = [1,2,3];
arr[0]       // ==> 1 (same as car)
arr.length   // ==> 3
arr.slice(1) // [2,3] (same as cdr) - returns a new array
```

```
const path =
    (keys: string[]) : ________________________________ => {

        ____________________________________________________________

        ____________________________________________________________

        ____________________________________________________________

        ____________________________________________________________

        ____________________________________________________________

        ____________________________________________________________

        ____________________________________________________________

        ____________________________________________________________
}
```

שאלה 2 [26 נק'] רשימות עצלות

א. **[3 נק']** שימוש ברשימות עצלות

1. ציין, או הדגם, שני מקרים בהם רצוי להשתמש ברשימות עצלות. הסבר את היתרונות.

_____

_____

_____

2. מהו החיסרון בשימוש ברשימות עצלות?

_____

_____

_____

ב. **[7 נק']** למערכת ריאקטיבית יש מספר ערוצי קלט שבהם נקלטים אירועים (events) באופן המשכי. על מנת לתחזק את המערכת הוחלט להתממשק עם ערוץ קלט דרך רשימה עצלה, ולתחזק את המערכת כרשימה של רשימות עצלות לכל ערוצי הקלט. לצורך קליטת אירוע מפעילים פונקציה לקריאה מערוץ: `(read-channel i)` קוראת את האירוע הבא בערוץ `i`. לצורך הדוגמה, נניח שהאירועים המתקבלים מקריאת הערוצים הם מספרים.

```
(define make-event-feed
  (lambda (i)
    (cons-lzl (read-channel i) (lambda () (make-event-feed i)))))

> (make-event-feed 1)
; Channel 1: 1, 2…
'(1 . #<procedure:...>)

(define start-reactive-system
  (lambda (n)
    (map make-event-feed
         (enumerate-interval 1 n))))

> (start-reactive-system 3)
; Channel 1: 1, 2…
; Channel 2: 1, 2…
; Channel 3: 100, 200…
'((1 . #<procedure:...>)
  (1 . #<procedure:...>)
  (100 . #<procedure:...>))
```

המימוש שלעיל לא מאפשר להבחין, בהינתן ערך של אירוע, באיזה ערוץ האירוע הופיע. תקן את המידול של המערכת הריאקטיבית, כך שליד כל אירוע יצוין מספר הערוץ שבו הוא הופיע:

```
> (start-reactive-system 3)
; Channel 1: 1, 2
; Channel 2: 1, 2
; Channel 3: 100
'(((1 . 1) . #<procedure:...>)
  ((2 . 1) . #<procedure:...>)
  ((3 . 100) . #<procedure:...>))
```

לצורך התיקון, השתמש בפרוצדורה `lzl-map` למיפוי על רשימות עצלות:

```
(define start-reactive-system
    ____________________________________________________________
    ____________________________________________________________
    ____________________________________________________________
    ____________________________________________________________
    ____________________________________________________________
    ____________________________________________________________
    ____________________________________________________________
    ____________________________________________________________
)
```

ג. **[6 נק']** רשימה עצלה יכולה לכלול מספר לא מוגבל (אינסופי) של נתונים.
רשימה היא **מוגדרת היטב** כאשר כל איבר של הרשימה ניתן לשליפה על ידי הפעלת מספר סופי של פעולות: לכל איבר **e** קיים **n** כך ש: **(nth lzl n) ➔ e**

מעוניינים לזהות, או, למצוא את ההופעה הראשונה של ערך כאירוע בערוץ כלשהו. לצורך זה, הוצע לאסוף את האירועים מכל הערוצים ולהפעיל פרוצדורת סינון `lzl-filter` על התוצאה. למשל:

```
> (lzl-filter (lambda (x) (= (cdr x) 2))
              (lzl-flatten (start-reactive-system 3)))
; Channel 1: 1, 3…
; Channel 2: 1, 2…
; Channel 3: 100, …
'((2 . 2) . #<procedure:...>)
```

לצורך זה, נכתבה פרוצדורה `bad-lzl-flatten` ל"שיטוח" רשימה של רשימות עצלות:

```
; Type: [List(Lzl) -> Lzl]
(define bad-lzl-flatten
  (lambda (lst)
    (if (empty? lst)
        empty-lzl
        (let ((lz1 (first lst))
              (rest-lzs (cdr lst)))
          (cond ((empty? rest-lzs) lz1)
                ((empty-lzl? lz1) (bad-lzl-flatten rest-lzs))
                (else
                  (cons-lzl (head lz1)
                    (lambda () (bad-lzl-flatten
                                 (cons (tail lz1) rest-lzs))))))))))
```

אבל הפעלתה על הנתונים שלעיל גרמה לולאה אינסופית:

```
> (lzl-filter (lambda (x) (= (cdr x) 2))
              (bad-lzl-flatten (start-reactive-system 3)))
; Channel 1: 1, 3, 4, 5, 6… (continue with increasing numbers)
; Channel 2: 1, 2, 3, 4, 5… (continue with increasing numbers)
; Channel 3: 100, 101, 102… (continue with increasing numbers)
ERROR: out-of-stack
```

הסבר למה הקריאה ל-**lzl-filter** לא מסתיימת:

_____

_____

_____

ציין האם הרשימה העצלה הנוצרת על ידי פרוצדורת השיטוח היא מוגדרת היטב.

_____

_____

_____

ד. **[10 נק']** תקן את הפרוצדורה כך שתיצור רשימה עצלה מוגדרת היטב:

```
; Type: [List(Lzl) -> Lzl]
(define lzl-flatten
  (lambda (lst)

  ____________________________________________________________
  ____________________________________________________________
  ____________________________________________________________
  ____________________________________________________________
  ____________________________________________________________
  ____________________________________________________________
  ____________________________________________________________
  ____________________________________________________________
  ____________________________________________________________
  ____________________________________________________________
  ____________________________________________________________
  ____________________________________________________________
)
```

שאלה 3 [20 נק'] דחיית חישוב ואופרטורים מיוחדים

א. **[3 נק']** דחיית חישוב היא טכניקה חישובית מקובלת. ציין שתי מטרות (הקשרים), שבהם משתמשים בדחיית חישוב.

_____

_____

ב. **[3 נק']** ציין והדגם אופני דחיית חישוב בכל אחת מהשפות: **Scheme, JavaScript**

_____

_____

_____

ג. **[2 נק']** מהו ההבדל בין אופרטור מיוחד **(special form)** לפרוצדורה פרימיטיבית?
הסבר והדגם בשפת **Scheme**.

_____

_____

ד. **[2 נק']** תארו את חוק ה חישוב של אופרטור המיוחד **OR** בשפת **Scheme**.

מדוע האופרטור איננו יכול להיות מוגדר כפרוצדורת משתמש ב-**applicative-eval**? הסבר והדגם.

_____

_____

ה. **[2 נק']** אחד הסטודנטים בקורס טען כי אין צורך באופרטור מיוחד, עם חוק חישוב משלו, בשביל פעולה כמו **OR**, והציע את פרוצדורת המשתמש הבאה:

```
; Signature: or(val1, val2)
(define or
  (lambda (val1 val2)
    (if val1
        val1
        val2)))
```

האם הגדת ה-**or** (עם פרוצדורת המשתמש) תואמת את סמנטיקת חוק החישוב שתואר בסעיף ד'? נמק והדגם

_____

_____

ו. [4 נק']

הגדירו מחדש את פרוצדורת המשתמש **or**, כך שתתאים לסמנטיקת חוק החישוב בסעיף ג', תוך שימוש בטכניקה של דחיית חישוב. (אין לשנות את החתימה, אך ניתן לאלץ עם תנאי התחלה את אופי הפרמטרים):

```
; Signature: or(arg1,arg2)
; Pre-condition: ____________________________________
(define or
  (lambda (arg1 arg2)
    ____________________________________________
    ____________________________________________
    ____________________________________________
    ____________________________________________
))
```

ז. [4 נק'] מהם החסרונות והיתרונות של הגדרת **OR** כאופרטור מיוחד, לעומת פרוצדורת משתמש:

1. אופרטור מיוחד:

יתרון: _____

חיסרון: _____

2. פרוצדורת משתמש:

יתרון: _____

חיסרון: _____

שאלה 4 [12 נק'] מודל הסביבות

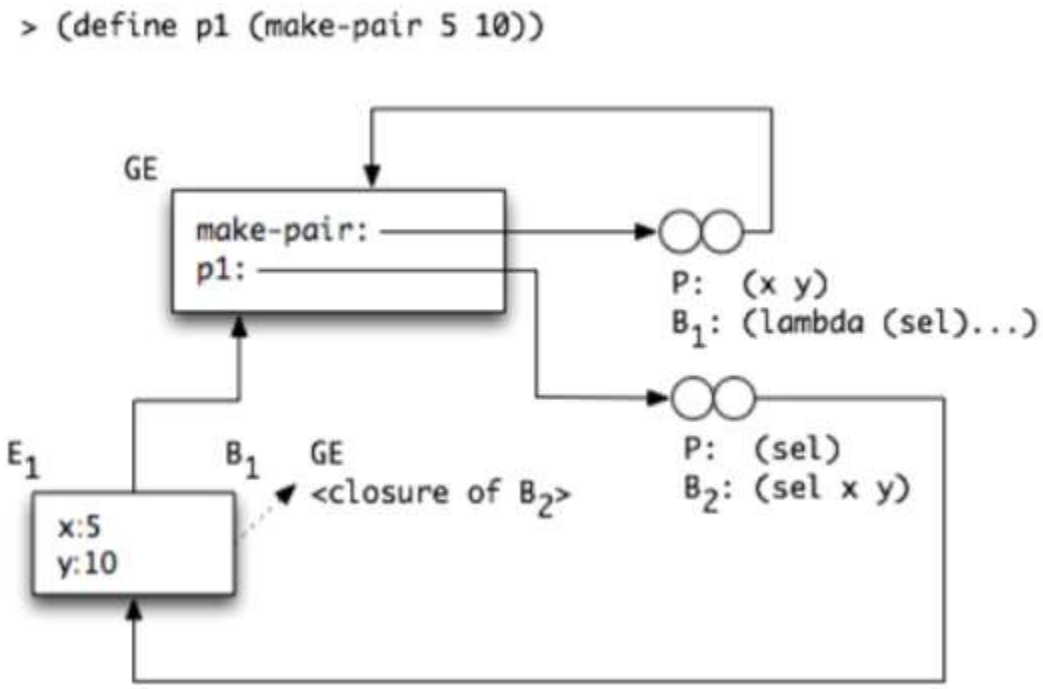
ניתנות ההגדרות למימוש מבוסס closures של מבנה נתונים pair:

```
(define make-pair
  (lambda (x y)
    (lambda (sel
      (sel x y))))

(define p1 (make-pair 5 10))
```

השלם את דיגרמת הסביבות המתקבל בהרצה של הביטוי:

```
(p1 (lambda (a b) (+ a b)))
```



שאלה 5 [נק'] תיכנות לוגי

א. [6 נק'] לשני המקרים הבאים - חשב את הunifier של שתי הנוסחאות.

פרט את צעדי החישוב ואת התוצאה:

unify[t(X,f(a),X), t(g(U),U,W)]

_____

_____

_____

_____

_____

unify[t(X,f(X),X), t(g(U),U,W)]

_____

_____

_____

_____

_____

ב. [8 נק']

ניתנות הפרוצדורות הבאות המגדירות את קידוד מספרים טיבעיים בתיכנות לוגית:

```
% Signature: natural_number(N)/1
% Purpose: N is a natural number.
natural_number(0).                                  % N1
natural_number(s(X)) :- natural_number(X).          % N2

% Signature: Plus(X,Y,Z)/3
% Purpose: Z is the sum of X and Y.
plus(X, 0, X) :- natural_number(X).                 % P1
plus(X, s(Y), s(Z)) :- plus(X, Y, Z).               % P2

% Signature: times(X,Y,Z)/3
% Purpose: Z = X*Y
times(0, X, 0) :- natural_number(X).                % T1
times(s(X), Y, Z) :- times(X, Y, XY), plus(XY, Y, Z). % T2
```

הגדר את הפרוצדורות הבאות:

```
% Signature: exp(X,N,Z)/2
% Purpose: Z = X^N (power)
```

_____

_____

```
% Signature: fact(N,F)/2
% Purpose: F = N! (factorial)
```

_____

_____

ג. [8 נק'] השלם את עץ ההוכחה החלקי שלהלן לחישוב כל התשובות. הקפד על שינוי שמות משתנים. על כל קשת בעץ, ציין מהי הצבת המשתנים עבורה, ומיהו הכלל שהופעל. ציין את התשובה בעלי הצלחה.