מבחן 06.07.14 סמ׳ ב׳ מועד ב׳:

שאלה 1: BNF

א. נכתוב את הדקדוק LE

①   ⟨LE⟩ ::= ⟨ATOM⟩
② | ⟨List⟩
③   ⟨List⟩ ::= Null
④ | (list ⟨LE⟩ ...)
⑤ | (append ⟨List⟩ ...)
⑥ | (cons ⟨LE⟩ ⟨List⟩)
⑦   ⟨ATOM⟩ ::= ⟨Num⟩
⑧ | ⟨Sym⟩

ב. הראה כיצד נוצרת שלוש המילים הבאות:

1) {23}

⟨LE⟩ ⟹① ⟨ATOM⟩
⟹⑦ ⟨Num⟩
⟹ סם הנתונה 23

2) (list 1 (cons 'two null) (append (append) (list)))

⟨LE⟩ ⟹② (⟨list⟩)
⟹④ (⟨list⟩ ⟨LE⟩ ⟨LE⟩ ⟨LE⟩)
⟹⑦ (list ⟨Num⟩ ⟨LE⟩ ⟨LE⟩)
⟹⑥ (list ⟨Num⟩ (cons ⟨LE⟩ ⟨List⟩) ⟨LE⟩)
⟹① (list ⟨Num⟩ (cons ⟨ATOM⟩ ⟨List⟩) ⟨LE⟩)
⟹⑧ (list ⟨Num⟩ (cons '⟨Sym⟩ ⟨List⟩) ⟨LE⟩)
⟹③ (list ⟨Num⟩ (cons '⟨Sym⟩ null) ⟨LE⟩)
⟹② (list ⟨Num⟩ (cons '⟨Sym⟩ null) ⟨List⟩)
⟹⑤ (list ⟨Num⟩ (cons '⟨Sym⟩ null) (append ⟨List⟩ ⟨List⟩))
⟹⑤ (list ⟨Num⟩ (cons '⟨Sym⟩ null) (append (append) ⟨List⟩))
⟹④ (list ⟨Num⟩ (cons '⟨Sym⟩ null) (append (append) (list)))
⟹ סם הנתונה (list 1 (cons 'two null)(append(append)(list)))

3) (cons 1 (cons (append (cons 'x null) (list 'y 'z)) null))

$\langle LE \rangle \overset{\textcircled{6}}{\Longrightarrow} \langle List \rangle$

$\overset{\textcircled{6}}{\Longrightarrow}$ (cons $\langle LE \rangle$ $\langle List \rangle$)

$\overset{\textcircled{1}}{\Longrightarrow}$ (cons $\langle ATOM \rangle$ $\langle List \rangle$)

$\overset{\textcircled{7}}{\Longrightarrow}$ (cons $\langle Num \rangle$ $\langle List \rangle$)

$\overset{\textcircled{6}}{\Longrightarrow}$ (cons $\langle Num \rangle$ (cons $\langle LE \rangle$ $\langle List \rangle$))

$\overset{\textcircled{2}}{\Longrightarrow}$ (cons $\langle Num \rangle$ (cons $\langle List \rangle$ $\langle List \rangle$))

$\overset{\textcircled{5}}{\Longrightarrow}$ (cons $\langle Num \rangle$ (cons (append $\langle List \rangle^{\langle List \rangle}$) $\langle List \rangle$))

$\overset{\textcircled{6}}{\Longrightarrow}$ (cons $\langle Num \rangle$ (cons (append (cons $\langle LE \rangle^{\langle List \rangle}\langle List \rangle$)) $\langle List \rangle$))

$\overset{\textcircled{1}}{\Longrightarrow}$ (cons $\langle Num \rangle$ (cons (append

(cons $\langle ATOM \rangle^{\langle List \rangle}\langle List \rangle$)) $\langle List \rangle$)

$\overset{\textcircled{8}}{\Longrightarrow}$ (cons $\langle Num \rangle$ (cons (append

(cons $\langle sym \rangle$ $\langle List \rangle^{\langle List \rangle}$)) $\langle List \rangle$))

$\overset{\textcircled{3}}{\Longrightarrow}$ (cons $\langle Num \rangle$ (cons (append

(cons '$\langle sym \rangle$ null) $\langle List \rangle$) $\langle List \rangle$))

$\overset{\textcircled{4}}{\Longrightarrow}$ (cons $\langle Num \rangle$ (cons (append

(cons '$\langle sym \rangle$ null) (list $\langle LE \rangle$ $\langle LE \rangle$)' $\langle List \rangle$))

$\overset{\textcircled{1}}{\Longrightarrow}$ (cons $\langle Num$ (cons (append

(cons '$\langle sym \rangle$ null) (list $\langle ATOM \rangle$ $\langle ATOM \rangle$) $\langle List \rangle$))

$\overset{\textcircled{8}}{\Longrightarrow}$ (cons $\langle Num \rangle$ (cons (append

(cons '$\langle sym \rangle$ null) (list '$\langle sym \rangle$ '$\langle sym \rangle$) $\langle List \rangle$))

$\overset{\textcircled{3}}{\Longrightarrow}$ (cons $\langle Num \rangle$ (cons (append (cons '$\langle sym \rangle$ null)

(list '$\langle sym \rangle$ '$\langle sym \rangle$) null))

$\overset{\text{עם כללים}}{\Longrightarrow}$ (cons 1 (cons (append (cons 'x null)

(list 'y 'z)) null))

dynamic :1) הגיﬞﬞ שהוא זﬞ של פונקﬥﬞﬞ כﬥﬢﬞﬞ - more - שﬞﬞﬞ ﬦוּﬨﬢﬞﬞﬞ ﬷﬷﬷ﬦﬞﬞﬞ

סﬨוּﬨﬞﬞﬞﬞﬞ ﬢﬞﬞﬞﬞﬞ ש﬷ﬤ﬷ זﬤ פונﬥﬞﬞﬞﬞ ﬥﬥﬢﬞﬞﬞ - static - שﬞﬞﬞﬞ ﬦוּﬨﬢﬞﬞﬞﬞ חוּוּ﬷ﬞﬞﬞ

ﬠ ﬠﬦוּﬢﬥﬥ ﬠﬠﬤוּוּﬦ הּﬠ﬷﬷﬩ ﬠﬞﬞ ﬥﬥﬥ ﬷ﬦוּ﬩﬷ הּﬨ ﬠ﬷וּ﬷וּוּ ﬢ-with ﬥﬞ ﬷ﬞהּוּוּוּ
ﬢ﬩﬷ﬞוּוּ all (ﬞ ﬞ) fun ﬞ) ﬢוּﬞ ﬠﬠ ﬢהּוּﬦﬥ ﬠוּﬢ﬩ﬠ﬩

ﬠ ﬢהּוּﬢﬥﬥ ﬠﬠﬨﬠﬤﬠﬠ ﬠוּהּ﬩ ﬷ﬞﬤﬤ ﬷ﬦהּ﬩ ﬦ﬷﬩ﬢﬞﬞ ﬠוּﬦ﬷﬷ﬞﬢ ﬢﬥﬥﬤ (ﬤﬦ﬷ﬞﬞהּוּ﬩ ﬤﬠהּוּﬞוּ הּﬤﬠוּﬞוּ

ﬠ ﬢהּוּﬢﬥﬥ ﬠוּﬠﬞﬞהּוּ ﬠﬞוּﬞﬤﬠﬠﬞוּﬞוּﬦﬤﬨﬦ הּ﬷﬷וּﬤﬠ הּﬦוּﬢﬥﬥ ﬠﬠﬨﬠﬤ﬷וּ

```
(Call (Call (fun x
             (Fun y (call (Id y) (Id x))))
             (Num 5))
      (with z
            (fun y (Add (Id y) (Num 1)))
            (Id z)))
```

א. כתבו את הקוד כאשר שמו של אף אחד מהמזהים אינו משתנה עד התחתית (במופעים),
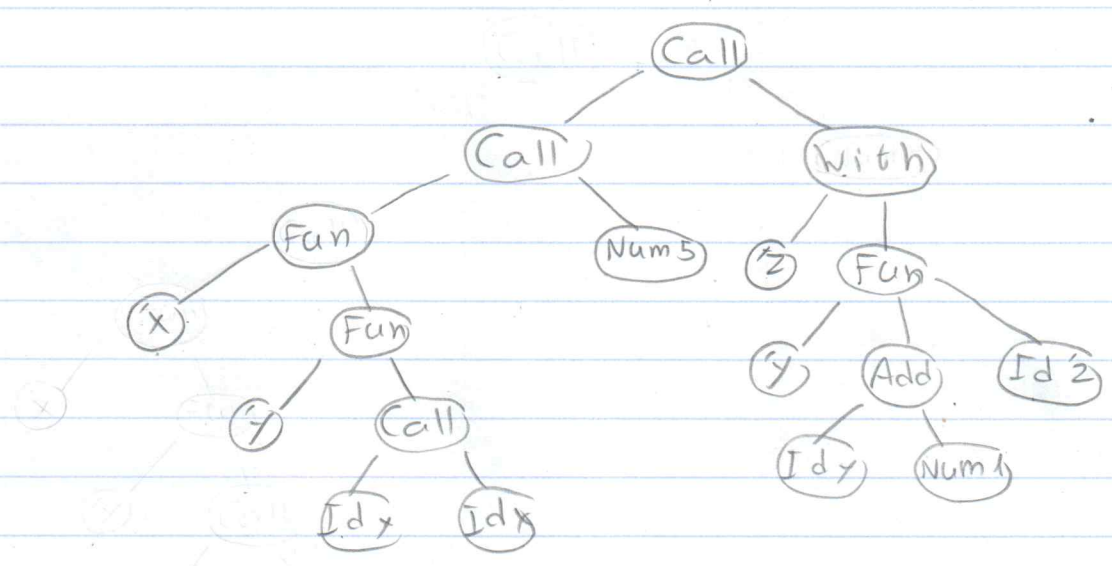
(run "{call {call {fun {x} {fun {y} {call y x }} } 5}
     {with {z} {fun {y} {+ y 1} } z} }")

```
((Call (Call (Fun x
                (Fun y (Call (Id y)(Id x))))  Num 5))
          (With z
              (Fun y (Add (Id y) (Num 1)))
              (Id z)))
```

1.   AST₁: (Call (Call (Fun x (Fun y (Call (Id y) (Id x)))) (Num 5))
                                    fun
                  (With z (Fun y (Add (Id y) (Num 1))) (Id z)))
                                    arg

ENV₁: (EmptyEnv)
RET₁: **NumV 6**

a.   a. AST₂: (Call (Fun x (Fun y (Call (Id y) (Id x)))) (Num 5))
                              fun   Num arg
ENV₂: (EmptEnv)
→ RET₂:

i. AST₃: (Fun x (Fun y (Call (Id y)(Id x))))
ENV₃: (EmptEnv)
RET₃: (FunV x (Fun y (Call (Id y) (Id x))) (EmptyEnv))

ii AST₄: Num 5
ENV₄: (EmptEnv)
RET₄: NumV 5

iii AST₅: Fun y (Call (Id y)(Id x)
ENV₅: (Extend x (NumV 5)(EmptyEnv))
                                    body
RET₅: (FunV y (Call (Id y)(Id x)) (Extend x (NumV 5)(EmptyEnv))

                                    name   body
b.   AST₆: (With z (Fun y (Add (Id y)(Num 1))) (Id z))
                    is
ENV₆: (Extend x (NumV 5)(EmptyEnv))
RET₆: ~~NumV 6~~
   ↗
Num 8
```

i   AST7: Fun y (Add (Id y) (Num 1))

   ENV7: (Extend x (NumV 5) (EmptyEnv))

   RET7: (FunV y (Add (Id y) (Num 1)) (Extend x (NumV 5) (EmptyEnv)

ii  AST8: Id z

   ENV8: (Extend z (FunV y (Add (Id y) (Num 1)

                  (Extend x (NumV 5) (EmptyEnv)))

                  (Extend x (NumV 5) (EmptyEnv))

   RET8: (FunV y (Add (Id y) (Num 1)) (Extend x (NumV 5) (EmptyEnv)

C.  ⟶  AST9: (Call (Id y) (Id x))
                    fun      arg

ENV9: (Extend y (FunV y (Add (Id y) (Num 1))) (Extend x (NumV 5) (EmotEnv)

                  (Extend x (NumV 5) (EmptyEnv))

   RET9: Num 6

i   AST10: Id y

   ENV10: (Extend y (FunV y (Add (Id y) (Num 1))) (Extend x (NumV 5) (EmptyEnv

                  (Extend x (NumV 5) (Empty Env)

   RET10: (FunV y (Add (Id y) (Num 1)) (Extend x (NumV 5) (Empty Env

ii  AST11: Id x

   ENV11: ENV10

   RET11: NumV 5

iii AST12: (Add (Id y) (Num 1)

   ENV12: (Extend y (NumV 5) Extend x (NumV 5) (EmptyEnv)

   RET12: NumV 6

1) AST13: Id y                  2) AST14: Num 1

   ENV13: ENV12                     ENV14: ENV12

   RET12: NumV 5                     RET14: NumV 1

שאלה 4: הרחבת השפה WAE

א. הוסיפו את הקוד הנדרש

```
(define-type WAE
  ⋮
  [Add (Listof WAE)]
  [Sub  WAE (Listof WAE)]
  [Mul (Listof WAE)]
  [Div WAE (Listof WAE)]
```

I. הוסיפו את הקוד הנדרש לפונקציית parse-sexpr

```
(: parse-sexpr : Sexpr -> WAE)
  ⋮
  [(list '+ (sexpr: args)...)
      (Add (parse-sexpr* args))]
  [(list '- fst (sexpr: args)...) (Sub (parse-sexpr fst)
                                      (parse-sexpr* args))]
  [(list '* (sexpr: args)...) (Mul (parse-sexpr* args))]
  [(list '/ fst (sexpr: args)...) (Div (parse-sexpr fst)
                                      (parse-sexpr* args))]
```

ג. הוסיפו פונקציית subst   (: subst : WAE Symbol WAE -> WAE)

```
  ⋮
  [(Add args) (Add (subst* args from to))]
  [(Sub fst args) (Sub (subst fst from to)
                       (subst* args from to)]
  [(Mul args) (Mul (subst* args from to)]
  [(Div fst args) (Div (subst fst from to)
                       (subst* args from to)
```

AST8: Id z

ENV8: (Extend z (FunV y (Add (Id y) (Num 1) (Extend x
            (call (Id y) (Id x)) (Extend x (NumV 5)) (EmptyEnv

RET8: (FunV $\overset{id}{y}$ (Add (Id y) (Num 1)), ENV7

AST9: (Call (Id y) (Id x))

ENV9: Extend y

        $\angle$

```
(:eval : WAE -> Number)
    :

[ (Add args) (foldl + 0 (eval args)) ]
[ (Sub fst args) (foldl - (eval fst) (eval args))
[ (Mul args) (foldl * 1 (eval args))
[ (Div fst args) (arith-op / (eval fst) (eval (mul args)) ]
```