בס"ד

מבחן 08/07/2015 סמ ב' מועד ב':

שאלה 1: BNF

א- כתיבת דקדוק לשפת ROL

```
<ROL>::= {reg-len = <Num> <RegE>}                              ①

<RegE>::= <List-Bits>                                          ②
        |<id>                                                  ③
        | {and <RegE> <RegE>}                                  ④
        | {or <RegE> <RegE>}                                   ⑤
        | {shl <RegE> }                                        ⑥
        | {with{<id> <RegE>}<RegE>}                            ⑦
        | {Fun{<id>} <RegE>}                                   ⑧
        | {call <RegE> <RegE>}                                 ⑨
<List-Bits>::={' {list <Bits> <Bits>} }                        ⑩
<Bits> ::= 0|1                                                 ⑪
```

ב. עבור כל אחת. מהמילים הבאות בדוק אם ניתן לייצר מילה ~~בחוק~~ מהדקדוק

1) {reg-len = 1 {+ 1 22}}

לא ניתן מכיוון שבחלק {+ 1 22} אין לנו בדקדוק + וגם אין ביט של 22

2) {reg-len = 3 {or {1 10} {01}}}

ניתן לייצר מילה כז-

```
<ROL> =①=> {reg-len = <Num> <RegE>}
      =⑤=> {reg-len = 3 {or <RegE> <RegE>}}
פעמיים =②=> {reg-len = 3 {or <List-Bits> <List-Bits>}
      =10=> {reg-len = 3 {or {<Bits> <Bits> <Bits>}
                                {<Bits> <Bits>}}}
      =⑪=> {reg-len =3 {or {1 10} {01}}}
```

←

3) {reg-len = 3   {with {f1 {fun {x} {shl {and x y}}}}
{call {call f1 {00 1}} {010}}}}

ניתן לגזור

4) {reg-len = 3 {call x z}}

<ROL> $\xrightarrow{1}$ {reg-len = 3 <RegE>}   ניתן לגזור

$\xrightarrow{2}$ {reg-len = 3 {call <RegE> <RegE>}}

$\xrightarrow{3}$ {reg-len = 3 {call <Id> <Id>}}

ע"פ הנתוני => {reg-len = 3 {call x z}}

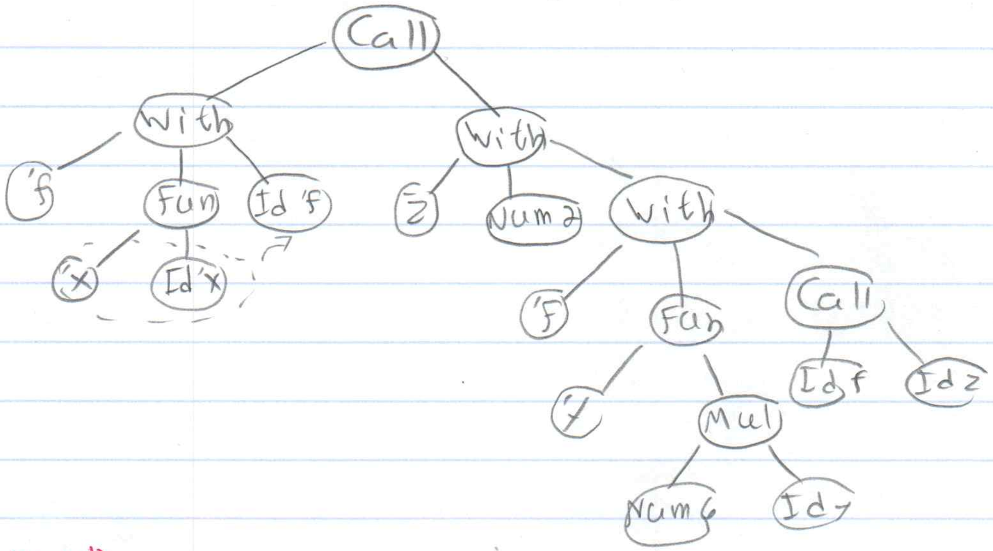
שאלה 2:

נתון הקוד הבא:

```
(run '{ call {with { f { fun {x} x} } f}
          {with { z 2}
               {with {f {fun {y} { * 6y}}}
                    { call  f z} }} }" }
```

א. eval לפי מודל ההחלפות

```
AST1: (Call (with f (Fun x (Id x)) Id f)
          (with z (Num2) (with f (Fun y (Mul (Num6) (Id y)))
                    ((Call (Id f) (Id z)
```



RET1: Num12

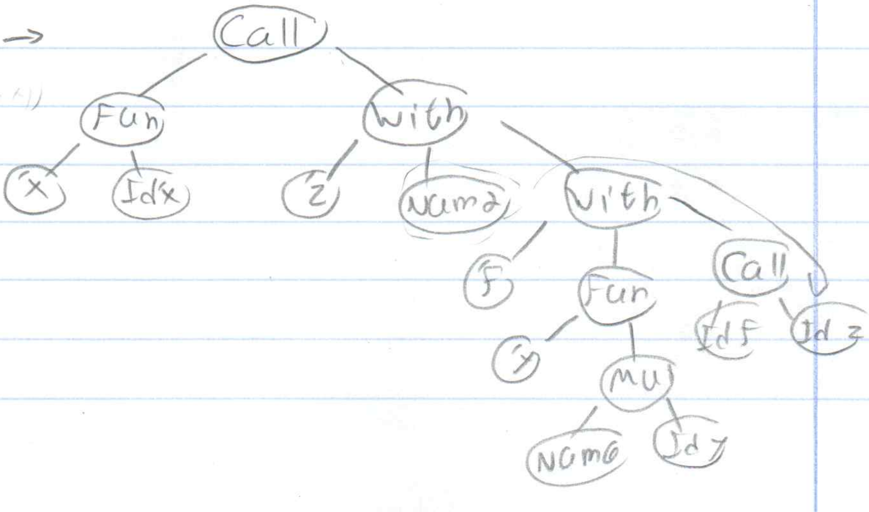
$\text{AST}_2\text{: (with }\overset{id}{\text{f}}\overbrace{\text{(fun x (Id x))}}^{name}\text{ If)}$

RET2: Num 12

AST3: (Fun x (Idx))

RET3: (Fun x (Idx)) →

החלפה →



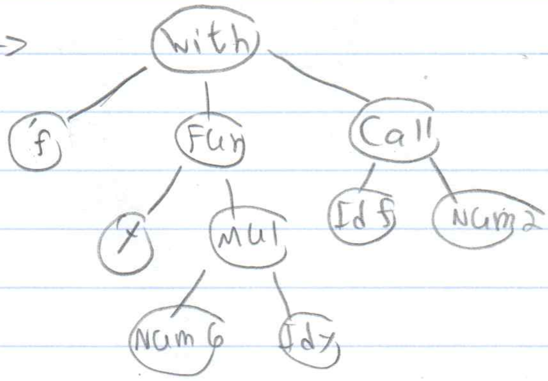
AST$_4$:(with z (Num2) (with f (Fun y (Mul (Num6)(Idy)))
(call (Id f) (Id z))

RET$_4$ Num 12

AST$_5$ Num 2

RE$_5$: Num 2

החלפה →



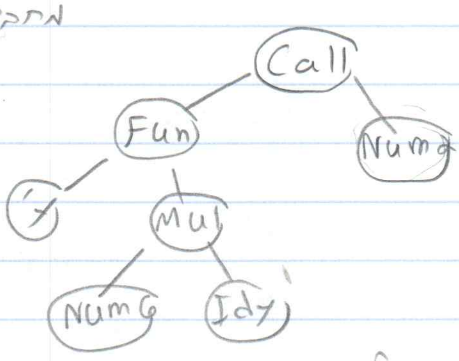
AST$_6$:(with f $\overbrace{\text{(Fun y (Mul(Num6)(Idy)))}}^{\text{name}}$ $\overbrace{\text{(call (Id f Num2))}}^{\text{body}}$

RET$_6$ Num 12

AST$_7$:(Fun y (Mul (Num6)(Idy)))

RET$_7$:(Fun y (Mul (Num6)(Idy)))

מתבצעת החלפה



AST$_8$:(Call $\overbrace{\text{(Fun y (Mul (Num 6) (Idy)))}}^{\text{fun}}$ $\overbrace{\text{Num 2}}^{\text{arg}}$)

RET$_8$: Num 12

AST$_9$:(Fun y (Mul) (Num 6) (Idy)))

RET$_9$:(Fun y (Mul (Num6) (Id y)))

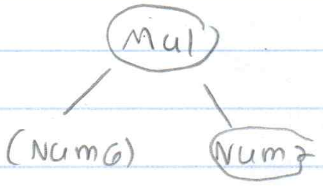
AST$_{10}$: Num 2

RET$_{10}$: Num 2

←

כבג

מתבצעת החלפה



AST_{11} : (Mul (Num 6) (Num 2)

RET_{11} : Num 12

AST_{12} : Num 6

RET_{12} : Num 6

AST_{13} : Num 2

RET_{13} : Num 2

ד. הי"ט מתקבל אותו דבר מכיוון ש-כי הכפל הוא שה-with התנאי
מצק יותר ולכן

א. (1)

```
(: lookup Symbol ENV -> Boolean)
 :
 [(EmptyEnv) #t]
 [(Extend id val rest env)
     (if eq? id name) #f (lookup name rest-env))]))
```

(2)

```
(: containsFreeInstanse? FLANG ENV -> Boolean)
  :
  [(Num n) #f]
  [(Add l r) (or (lookup l
[`list 'fun (list (symbol: name)) body)
  (Fun name (parse-sexpr-Regl body reg-len))]

[`list 'call fun arg (Call (pars-sexp-Regl fun reg-len
                           (parse "     "     arg reg-len)
```

```
(define (bit-and a b)                                    .X
(if (and (= 1 a) (= 1 b)) 1 0)

(define (bit-or a b)
if (or (= 1 a) (= 1 b)) 1 0

(define (bit-arith-op bl1 bl2)                  [illegible] .T
    (if (null? bl1) null
       ( op (first


(cons (op (first bl1) (first bl2) )
          (bit-arith-op op (rest bl1) (rest bl2)
   (Reg V
```