

18.03.2021

## מבחן סוף סמסטר – מועד ב'

ד"ר שחר יצחקי

מתן פلد, איתן סיינגר, עומר בלחשין

מרצה אחראי:

מתרגלים:

### הוראות:

- א. בטופס המבחן 13 עמודים מהם 6 דפי נוסחאות. בדקו שכל העמודים ברשותכם.
- ב. משך המבחן שעתיים וחצי (150 דקות).
- ג. אסור כל חומר עזר חיצוני.
- ד. במבחן 4 שאלות. כל השאלות הין חובה. משקל כל שאלה מופיע בראשיתה. (חלוקת המשקל בין הסעיפים בכל שאלה אינה בהכרח אחידה).
- ה. ניתן לצין לגבי סעיף או שאלה "לא יודעת". תשובה זו תזכה ב- 20% מהניקוד של הסעיף או השאלה.
- ו. חובה לנמק כל תשובה. לא ניתן ניקוד על תשובות ללא נימוק.
- ז. קראו את כל המבחן לפני שאתם מתחילה לענות על השאלות.
- ח. אין צורך להגיש את הטופס בתום הבחינה.
- ט. את התשובות לשאלות יש לרשום במחברת הבחינה בלבד.

**בהצלחה!**

שאלה 1: שלבי הקומPILEציה ואופטימיזציות (25 נק')חלק א': סיווג מאורעות (10 נק')

נתונה התוכנית הבאה בשפת C :

```

1 int foo(int n) {
2     int c = 1;
3     while (c != 0) {
4         c = c - 1;
5         if (n > 100) {
6             n = n - 10;
7         } else {
8             n = n + 11;
9             c = c + 2;
10        }
11    }
12    return n;
13 }
14 int bar(int x) {
15     x = x * 2 + 2;
16     if (x > 50) {
17         return x - 50;
18     } else {
19         return x + 50;
20     }
21 }
22 void main() {
23     int acc = 0;
24     int i = 0;
25     while (i < 500) {
26         acc = acc + foo(bar(16 + 32));
27     }
28 }
```

בסעיפים הבאים מוצגים שינויים (בלתי תלויים) לקוד של התוכנית, או מאורע שתרחש בזמן השימוש בקוד.  
 שלבי הקומPILEציה הם : ניתוח לקסיקלי, ניתוח תחבירי, ניתוח סמנטי, ייצור קוד, זמן ריצה.  
 הניתו כי התוכנית לפני השינויים מותקפלת ורצת ללא שגיאות. עברו כל שינוי כתבו האם הוא גורם לשגיאה,  
 ואם כן, צינו את השלב המוקדם ביותר שבו נגלה אותה ונמקו בקצרה :

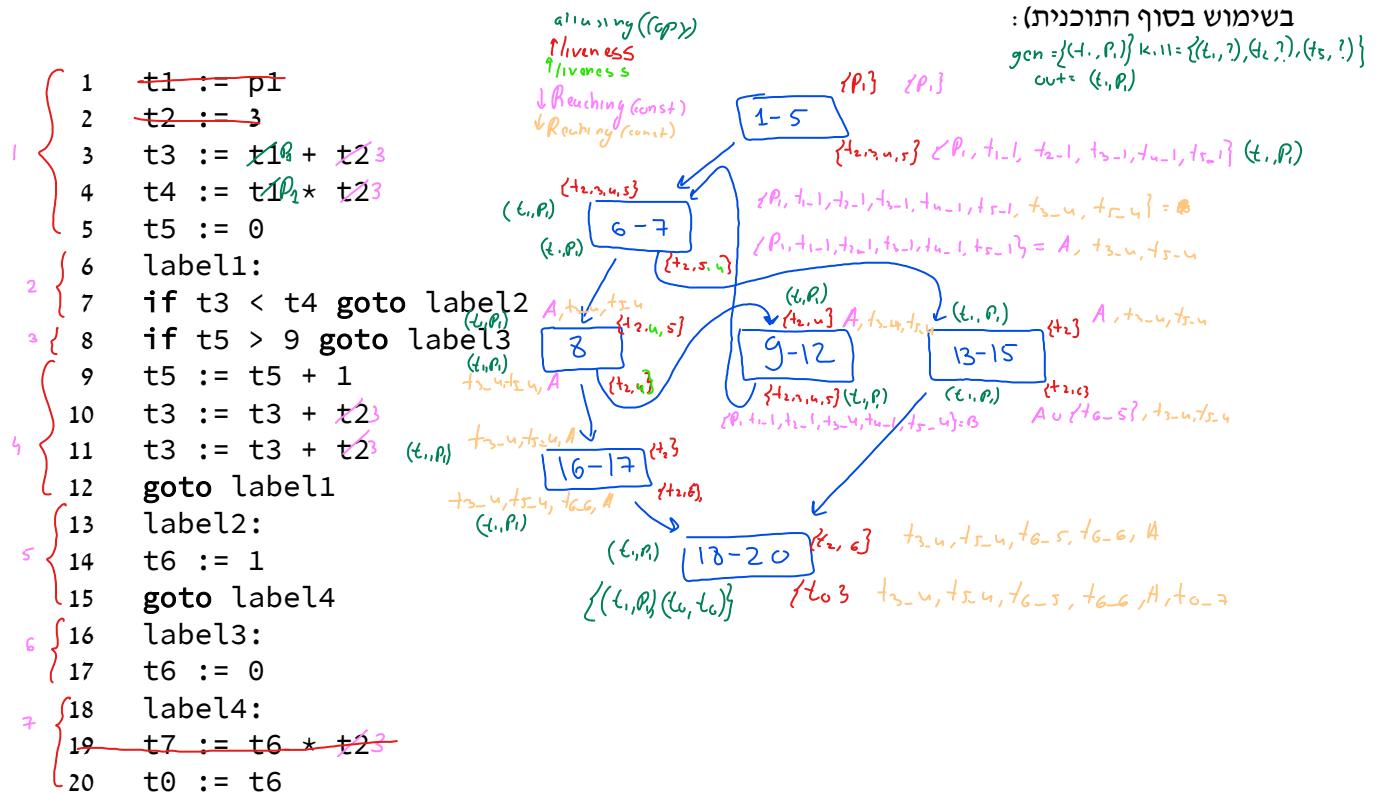
- implcit int to byte cast      ( Semantic )      byte i = 0;      b: ;  
 a. (2 נק') מחליפים את שורה 24 ב: ;  
 lex error  
 b. (2 נק') מחליפים את שורה 19 ב: ;  
 return x ^ 50;  
 g. (2 נק') מוחקדים את שורה 24.      ? do what semantic error, what

עבור כל מאורע, צינו את **כל** שלבי הקומPILEציה בהם הוא יכול להתרחש ונמקו בקצרה.

- ד. (2 נק') השם acc מקשר לטיפוס כלשהו (מזכיר בשורות 23 ו-26).      ? do what  
 ה. (2 נק') מחושב הסכום 16+32 (שורה 26).      ? do what.      ? do what.      ? do what.

חלק ב': אופטימיזציות (15 נק')

נתונה התוכנית הבאה בשפת הבינרים שנלמדה בכיתה, כאשר t1 הוא פרמטר ו-t0 הוא ערך חוזר (כלומר



- א. (3 נק') בצעו את אלגוריתם החלוקה ל-basic blocks שנלמד בכיתה, וציירו את ה-HCFG של התוכנית הנתונה. התשובה צריכה להיות בצורת תרשימים.

ב. (12 נק') בצעו על התוכנית הנתונה את האופטימיזציות הבאות:

Constant propagation/folding (reaching definitions) .1

Copy propagation (aliasing analysis) .2

Useless code elimination (liveness analysis) .3

הראו את התוצאה של הריצת האנליזה הרלוונטיות לכל אופטימיזציה, ואז את המצב הסופי של הקוד אחרי הפעלת כל האופטימיזציות. סמן בכל שורה שבתבצעה אופטימיזציה, סמן או איזו אופטימיזציה פעלת.

**שאלה 2: אنجזה סטטיות (30 נק')**

ריך הוא גאון, ולכן הוא המציא שפת תכנות וסביבת זמן-ריצה שבה יש ניהול זיכרון אוטומטי, בהתבסס על שיטת **Mark & Sweep**. בשפה שלו יש משתנים לוקליים וגלובליים, טיפוסי מצביעים בסגנון Java – כולל – מצביעים שיכולים להצביע אך ורך לאובייקטים, ותמיד לתחילה אובייקט (אין pointer arithmetic) ומפטים מהצורה הבאה:

```
A x;           // declare `x` as pointer to `A`
x := new A    // allocate object of class `A`
x := y        // copy pointer
x := y.f      // field read
x.f := y      // field write
```

ובנוסף ביטויים אריתמטיים ובוליאניים, מפטים תנאי וЛОואות כמו בכל שפת תכנות סבירה והגיונית.

ריך מעוניין לדעת לגבי כל אובייקט שהוקצה דינמית בשורה כלשהי או בקוד שלו, האם הוא **יכול להיות נגיש** (reachable) לאחר שהפונקציה הוכחה — זו שמכילה את השורה או מסתימת. אובייקט שנשאר נגיש לאחר שהפונקציה שהקצתה אותו הסתיימה נקרא אובייקט בורה (escaped). לדוגמה:

1 void wubba() { 2     Dub x; 3     x := new Dub; 4     x.dub(); 5     return; 6 }	1 Dub lubba() { 2     Dub y; 3     y := new Dub; 4     y.dub(); 5     return y; 6 }
`new Dub` at line 2: <i>not escaped</i>	`new Dub` at line 2: <i>escaped</i>

א. (15 נק') הניחו שבתכניות של ריך אין **שדות מティפוס מצביע** בהגדירות של טיפוסים מורכבים וכן אין **מערכיים**. הציעו לריך אنجזה **נאורה** – כזו שאם היא אומרת שהקצתה בשורה מסוימת אינה בורה, אז האובייקט המוקצת בודאות אינו בורה, באף ריצה אפשרית. מותר לאנגזה להיות שמרנית (כלומר, להגיד שאובייקט עלול לבורח ממחזקתו גם כאשר בכל הריצות הוא אינו נגיש מעשה לאחר היציאה מהמחזקה); אבל ריך דורש שתכתבו אנגזה מדויקת ככל האפשר, ולא כדי להכעיס את ריך.

הגדרו את טווח הערכים (הdomין), יחס הסדר, פעולה join, ופונקציות המעבירים המתאימות עבור המשפטים השונים בשפה.

ב. (10 נק') הסבירו כיצד הייתה מושנים את האנגזה אם בתכניות **יתכנו שדות מティפוס מצביע**, למשל

```
1 class Wubba { Lubba dub; }
:
101 x = new Wubba;
102 x.y = new Lubba;
```

escaped

```
201 x = new Wubba;
202 x.y = new Lubba;
203 return z; // does line 202 escape? 😊
```

בסעיף זה יש לשמור את הנאותות, אך אין דרישת שהאנליזה מדויקת (למעשה זה קשה מאד לדiyik במרקירים אלה, כמו בשורות 202, 203 בדוגמה). עם זאת, אין לפגוע בדיקות של האנליזה במרקירים המתווארים בסעיף א.

**הנחייה:** נסו להשאיר את הדומיינו כמות שהוא ולשנות רק את פונקציות המעברים (זה יכול לחסוך הרבה עבודה ותסכול).

ג. (5 נקי) מורותי בוהה באנליזה שתכתבם, ותמה: "אבל למה זה טוב בכלל לדעת אם אובייקט יכול או לא יכול לבРОוח מפונקציה?"

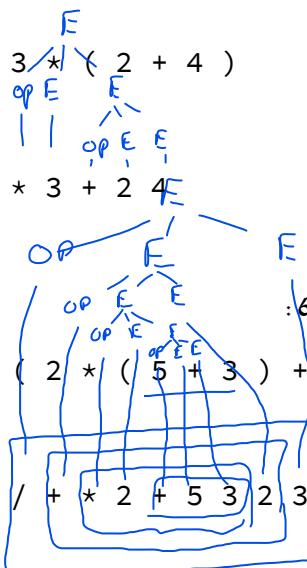
תארו למורותי **אופטימיזציה** שהקומפילר של ריק יכול לבצע בעזרה האנליזה, ובאיזה מקרים היא מתאפשרת. התיחסו בתשובתכם לניהול זיכרונות מחסנית ובערמה.

שאלה 3: ניתוח תחבורי וסמנטי (5 נק')

בתרגיל זה נבנה ניתוח תחבורי וסמנטי על מנת לחשב ביטויים חישובניים בייצוג פרפיקסי (prefix).  
בייצוג זה, ביטויים חישובניים בניוים תחילת מהאופרטור ולאחר מכן מושני האופרנדים (**מספריים בלבד**), למשל הביטוי הבא שערכו 8:

+ 3 5

נשים לב שייצוג פרפיקסי מיתר את הצורך בלסמן את סדר הפעולות בביטוי ע"י סוגרים. למשל הביטוי הבא בצורתו הרגילה, שערכו 18:



שקלל בביטוי הבא בייצוג פרפיקסי:

בנוסף, נוכל לכתוב ביטויים מורכבים יותר, כגון הביטוי הבא בצורה הרגילה שערכו 6:

+ 2 4  
x 3

השקלל בביטוי הבא בייצוג פרפיקסי:

א. (5 נק') השלימו את כללי הגזירה על מנת להגדיר דקדוק חסר-הקשר המקבל את שפת הביטויים החישובניים בייצוג פרפיקסי שנוכל למשו באמצעות המנתח Bison שנלמד בקורס. הטרמינלים בדקדוק מסוימים בכו תחתיו, ניתן להגדיר טרמינלים נוספים במידת הצורך.

- S       $\rightarrow E$
- E       $\rightarrow \frac{OP}{num}$
- E       $\rightarrow \frac{num}{OP}$
- OP      $\rightarrow \pm \cdot \cdot \cdot \cdot \cdot$

num – אינטגר, סיכון גודל – מינימום או מקסימום  
OP – אופרטור, סיכון גודל – מינימום או מקסימום  
S – סמלים רגולריים, סיכון גודל – מינימום או מקסימום

But sum-up, we can do it

$OP \rightarrow \frac{+}{\ast} : \quad \{ \$\$.op = \$1.to\_str() \}$

$E \rightarrow num : \quad \{ \$\$.val = \$1.val \}$

$E \rightarrow OP E E : \quad \{ switch (\$1.op) {$   
     $\text{case } +: E1.val = E1.val + E2.val; break;$   
     $\text{case } /: if (E2.val == 0) output::divByZero();$   
         $E1.val = E1.val / E2.val;$   
         $break; \}$

$S \rightarrow E : \quad \{ \$\$.val = \$1.val; \}$

בסעיפים הבאים נבנה הגדרה מונחת תחביר (תכונות וכללים סמנטיים) שתדפיס את ערכו של ביטויי חשבוני ביצוג פרטיקסי הנוצר מהדקוד שהגדרתם בסעיף א'.

ב. (5 נק') הגדרו את **התכונות הסמנטיות** הדרשות על מנת להדפיס את ערכו של ביטויי חשבוני ביצוג פרטיקסי בעת הניתוח הסמנטי. הסבירו את תשובתכם. ✓

ג. (15 נק') פרטו את **הכללים הסמנטיים** הדרושים באמצעות פסאודו-קוד בשימוש התכונות שהגדרתם בסעיף ב' על מנת להדפיס את ערכו של ביטויי חשבוני ביצוג פרטיקסי בעת הניתוח הסמנטי. ✓

הנחיות:

- השתמשו בתכונות נורשות בלבד או בתכונות נוצרות בלבד.
- יש לבצע את הניתוח הסמנטי בזמן בניית עץ הגירה.

שאלה 4: Backpatching (20 נק')

הוסיפו לשפת FanC מבנה בקרה עבור לולאת `for` מעל טווח עם פרדיקט. לולאה זו מכילה שתי רצפי פקודות, הרץף הראשון `Statements1` יירוץ לכל איבר במידה והפרדיקט מבחן עבורי `true` ו-`false`. יירוץ במידה והפרדיקט יחזיר `false`. הטווח נתון ע"י `NUM1` ו-`NUM2` והינו כוללני `[num1, num2]`. `ID1` מכיל מזהה שמשמש כשם עבר האינדקס של הלולאה. `ID2` מכיל מזהה של פונקציה שתשמש כפרדיקט. נתון כלל גזירה חדש לשפה:

$$\begin{aligned} Statement \rightarrow & \underline{\text{for } NUM_1 \text{ to } NUM_2 \text{ as } ID_1 \text{ with } ID_2 \text{ LBRACKET}} \underline{Statements_1 \text{ RBRACKET}} \\ & \underline{\text{LBRACKET}} \underline{Statements_2} \underline{\text{RBRACKET}} \end{aligned}$$

דוגמא :

```
bool p(int x) {
    return x < 2;
}
for 0 to 3 as x with p {
    printi(x);
    print("p(x) returned true");
} {
    printi(x);
    print("p(x) returned false");
}
```

תדפיס :

```
0
p(x) returned true
1
p(x) returned true
2
p(x) returned false
3
p(x) returned false
```

לצורך המימוש ניתן להניח שהתוכנה תמיד נכונה תחבירית וסמנטית, זאת אומרת שאין בעיות טיפוסים או דרישת של משתנים.

בנוסף, בדומה להרצאות, נוסך לשפת נתונים מבנה של קריאה לפונקציה. תחילת מצהירים על פרמטרים ולאחר מכן מפעילים את הפונקציה ומציינים את מספר הפרמטרים המועברים. לדוגמה קריאה לפונקציה `f` עם פרמטר אחד (`t1`):

```
param t1
```

a = call f, 1

א. (8 נק') הציעו פריסת קוד המתאים לשיטת backpatching עבור מבנה הבדיקה הניל. על הקוד הנוצר להיות יעיל ככל האפשר. הסבירו מהן התוכנות שאתם משתמשים בהן עבור כל משתנה.

ב. (12 נק') כתבו סכימת תרגום בשיטת backpatching המיצרת את פריסת הקוד שהוצעם בסעיף הקודם. על הסכימה להיות יעילה ככל האפשר, הן מבחינת זמן הריצה שלה והן מבחינת המקום בזיכרון שנדרש עבור התוכנות הסמנטיות.

שימוש לב:

- אין לשנות את הדקדוק, למעט הוספת מקרים N,M שנלמדו בכיתה בלבד.
- אין להשתמש בכללים סמנטיים באמצעות כל גזירה.
- אין להשתמש במשתנים גלובליים בזמן קומpileציה.
- למשתנה Statement יש כללי גזירה פרט לאלו המוצגים בשאלת.

## סוף המבחן



## נוסחאות ואלגוריתמים

כל ההגדרות מתייחסות לדקזוק  $G = (V, T, P, S)$ .

### Top Down

$$\text{first}(\alpha) = \{ t \in T \mid \alpha \Rightarrow^* t\beta \wedge \beta \in (V \cup T)^* \}$$

$$\text{follow}(A) = \{ t \in T \cup \{\$\} \mid S\$ \Rightarrow^* \alpha At \beta \wedge \alpha \in (V \cup T)^* \wedge \beta \in (V \cup T)^*(\epsilon | \$) \}$$

$$\text{select}(A \rightarrow \alpha) = \begin{cases} \text{first}(\alpha) \cup \text{follow}(A) & \alpha \Rightarrow^* \epsilon \\ \text{first}(\alpha) & \text{otherwise} \end{cases}$$

**הגדרה:** דקזוק  $G$  הוא  $\text{LL}(1)$  אם ורק אם לכל שני כלליים ב-  $G$  השبيיכים לאותו משתנה  $A$  מתקאים:  
 $\text{select}(A \rightarrow \alpha) \cap \text{select}(A \rightarrow \beta) = \emptyset$

הגדרת טבלת המעברים  $M : V \times (T \cup \{\$\}) \rightarrow P \cup \{\text{error}\}$  עבור דקזוק  $\text{LL}(1)$ :

$$M[A, t] = \begin{cases} A \rightarrow \alpha & t \in \text{select}(A \rightarrow \alpha) \\ \text{error} & t \notin \text{select}(A \rightarrow \alpha) \text{ for all } A \rightarrow \alpha \in P \end{cases}$$

:  $\text{LL}(1)$

```

Q.push(S)
while !Q.empty() do
    X = Q.pop()
    t = next token
    if X ∈ T then
        if X = t then MATCH
        else ERROR
    else // X ∈ V
        if M[X , t] = error then ERROR
        else PREDICT(X , t)
    end if
end while
t = next token
if t = $ then ACCEPT
else ERROR
  
```

## Bottom Up

<p><b>פריט LR(0)</b> הוא <math>A \rightarrow \alpha\beta \in P</math> כאשר <math>(A \rightarrow \alpha\bullet\beta) \in closure(I)</math> על קבוצת פריטים <math>I</math> מוגדר באופן אינדוקטיבי:</p> <ul style="list-style-type: none"> <li>○ <math>.closure(I) = I</math> : בסיס.</li> <li>○ <math>\text{צעד : אם } (B \rightarrow \bullet\gamma) \in closure(I), \text{ אז לכל } \gamma \in P, \text{ גם } B \rightarrow \gamma \in closure(I), \text{ אז } A \rightarrow \alpha\bullet B\beta \in closure(I)</math>.</li> </ul> <p><b>פונקציית המעברים של האוטומט :</b></p> $\delta(I, X) = \bigcup \left\{ closure(A \rightarrow \alpha X \bullet \beta) \mid (A \rightarrow \alpha \bullet X \beta) \in I \right\}$
--

<p><b>פריט LR(1)</b> הוא <math>t \in T \cup \{\\$\}</math>, <math>A \rightarrow \alpha\beta \in P</math> כאשר <math>(A \rightarrow \alpha\bullet\beta, t) \in closure(I)</math> על קבוצת פריטים <math>I</math> מוגדר באופן אינדוקטיבי:</p> <ul style="list-style-type: none"> <li>○ <math>.closure(I) = I</math> : בסיס.</li> <li>○ <math>\text{צעד : אם } (B \rightarrow \bullet\gamma, x) \in closure(I), \text{ אז לכל } \gamma \in P \text{ ולכל } t \in first(\beta t), (A \rightarrow \alpha\bullet B\beta, t) \in closure(I)</math>.</li> </ul> <p><b>פונקציית המעברים של האוטומט :</b></p> $\delta(I, X) = \bigcup \left\{ closure(A \rightarrow \alpha X \bullet \beta, t) \mid (A \rightarrow \alpha \bullet X \beta, t) \in I \right\}$
--

: הגדרת טבלת action למנתח SLR

$action[i, t] = \begin{cases} SHIFT_j & \delta(I_i, t) = I_j \\ REDUCE_k & \text{rule } k \text{ is } A \rightarrow \alpha, (A \rightarrow \alpha\bullet) \in I_i \text{ and } t \in follow(A) \\ ACCEPT & (S' \rightarrow S\bullet) \in I_i \text{ and } t = \$ \\ ERROR & \text{otherwise} \end{cases}$
---

: הגדרת טבלת action למנתח LR(1)

$action[i, t] = \begin{cases} SHIFT_j & \delta(I_i, t) = I_j \\ REDUCE_k & \text{rule } k \text{ is } A \rightarrow \alpha \text{ and } (A \rightarrow \alpha\bullet, t) \in I_i \\ ACCEPT & (S' \rightarrow S\bullet, \$) \in I_i \text{ and } t = \$ \\ ERROR & \text{otherwise} \end{cases}$
---

: הגדרת טבלת goto ו-SLR action למנתח LR(1)

$goto[i, X] = \begin{cases} j & \delta(I_i, X) = I_j \\ error & \text{otherwise} \end{cases}$
---

אלגוריתם מנתח : shift/reduce

```

Q.push(0)           // where 0 is the initial state of the prefix automaton
while true do
    k = Q.top().state
    t = next token
    do action[k , t]
end while

```

## קוד בינויים

$x := y \text{ op } z$   
 $x := \text{op } y$   
 $x := y$   
**goto L**  
**if**  $x \text{ relop } y$  **goto L**  
**print**  $x$

- סוגי פקודות בשפת הבינויים :
1. משפטי השמה עם פעולה ביןארית
  2. משפטי השמה עם פעולה אונריה
  3. משפטי העתקה
  4. קפיצה בלתי מותנה
  5. קפיצה מותנה
  6. הדפסה

## Data-Flow Analysis

הגדרות מתאימות ל-  $G=(V,E)$ : CFG

הצורה הכללית של המשוואות בחישוב סריקה קדמית :

$$\begin{aligned} \text{in}(B) &= \bigcap_{(S,B) \in E} \text{out}(S) \quad \text{או} \quad \text{in}(B) = \bigcup_{(S,B) \in E} \text{out}(S) \\ \text{out}(B) &= f_B(\text{in}(B)) \end{aligned}$$

הצורה הכללית של המשוואות בחישוב סריקה אחוריית :

$$\begin{aligned} \text{out}(B) &= \bigcap_{(B,S) \in E} \text{in}(S) \quad \text{או} \quad \text{out}(B) = \bigcup_{(B,S) \in E} \text{in}(S) \\ \text{in}(B) &= f_B(\text{out}(B)) \end{aligned}$$

# FanC שפת

איסימוניים:

תבנית	איסימון
void	VOID
int	INT
byte	BYTE
b	B
bool	BOOL
set	SET
and	AND
or	OR
not	NOT
true	TRUE
false	FALSE
return	RETURN
if	IF
else	ELSE
while	WHILE
break	BREAK
continue	CONTINUE
;	SC
,	COMMA
(	LPAREN
)	RPAREN
{	LBRACE
}	RBRACE
[	LBRACKET
]	RBRACKET
=	ASSIGN
==   !=   <   >   <=   >=   in	RELOP
+   -   *   /	BINOP
..	DOTS
[a-zA-Z][a-zA-Z0-9]*	ID
0   [1-9][0-9]*	NUM
"([^\n\r\"\\] \\[rnt\"\\\])+"	STRING

**דקדוק:**

1.  $Program \rightarrow Funcs$
2.  $Funcs \rightarrow \epsilon$
3.  $Funcs \rightarrow FuncDecl\ Funcs$
4.  $FuncDecl \rightarrow RetType\ ID\ LPAREN\ Formals\ RPAREN\ LBRACE\ Statements\ RBRACE$
5.  $RetType \rightarrow Type$
6.  $RetType \rightarrow VOID$
7.  $Formals \rightarrow \epsilon$
8.  $Formals \rightarrow FormalsList$
9.  $FormalsList \rightarrow FormalDecl$
10.  $FormalsList \rightarrow FormalDecl\ COMMA\ FormalsList$
11.  $FormalDecl \rightarrow Type\ ID$
12.  $FormalDecl \rightarrow EnumType\ ID$
13.  $Statements \rightarrow Statement$
14.  $Statements \rightarrow Statements\ Statement$
15.  $Statement \rightarrow LBRACE\ Statements\ RBRACE$
16.  $Statement \rightarrow Type\ ID\ SC$
17.  $Statement \rightarrow Type\ ID\ ASSIGN\ Exp\ SC$
18.  $Statement \rightarrow EnumType\ ID\ ASSIGN\ Exp\ SC$
19.  $Statement \rightarrow ID\ ASSIGN\ Exp\ SC$
20.  $Statement \rightarrow Call\ SC$
21.  $Statement \rightarrow RETURN\ SC$
22.  $Statement \rightarrow RETURN\ Exp\ SC$
23.  $Statement \rightarrow IF\ LPAREN\ Exp\ RPAREN\ Statement$
24.  $Statement \rightarrow IF\ LPAREN\ Exp\ RPAREN\ Statement\ ELSE\ Statement$
25.  $Statement \rightarrow WHILE\ LPAREN\ Exp\ RPAREN\ Statement$
26.  $Statement \rightarrow BREAK\ SC$
27.  $Statement \rightarrow CONTINUE\ SC$
28.  $Call \rightarrow ID\ LPAREN\ ExpList\ RPAREN$
29.  $Call \rightarrow ID\ LPAREN\ RPAREN$
30.  $ExpList \rightarrow Exp$
31.  $ExpList \rightarrow Exp\ COMMA\ ExpList$
32.  $Type \rightarrow INT$
33.  $Type \rightarrow BYTE$
34.  $Type \rightarrow BOOL$
35.  $Type \rightarrow SET\ LBRACKET\ NUM\ DOTS\ NUM\ RBRACKET$
36.  $Exp \rightarrow LPAREN\ Exp\ RPAREN$
37.  $Exp \rightarrow Exp\ BINOP\ Exp$

38.  $Exp \rightarrow ID$
39.  $Exp \rightarrow Call$
40.  $Exp \rightarrow NUM$
41.  $Exp \rightarrow NUM\ B$
42.  $Exp \rightarrow STRING$
43.  $Exp \rightarrow TRUE$
44.  $Exp \rightarrow FALSE$
45.  $Exp \rightarrow NOT\ Exp$
46.  $Exp \rightarrow Exp\ AND\ Exp$
47.  $Exp \rightarrow Exp\ OR\ Exp$
48.  $Exp \rightarrow Exp\ RELOP\ Exp$
49.  $Exp \rightarrow LPAREN\ Type\ RPAREN\ Exp$