23.08.2020

מבחן סוף סמסטר – מועד ב׳

חלק א׳

מרצה אחראי: ד״ר שחר יצחקי

מתרגלים: אדם בוטח, שקד ברודי, רפאל כהן, יעקב סוקוליק

הוראות:

א. בחלק זה של המבחן 11 עמודים מהם 7 דפי נוסחאות. בדקו שכל העמודים ברשותכם.

ב. משך חלק זה של המבחן הוא שעה (60 דקות).

ג. אסור כל חומר עזר חיצוני.

ד. בחלק זה של המבחן 2 שאלות. כל השאלות הינן חובה. משקל כל שאלה מופיע בראשיתה. (חלוקת המשקל בין הסעיפים בכל שאלה אינה בהכרח אחידה.)

ה. ניתן לציין לגבי סעיף או שאלה ״לא יודע/ת״. תשובה זו תזכה ב- 20% מהניקוד של הסעיף או השאלה. תשובות שגויות לא יזכו בניקוד.

ו. חובה לנמק כל תשובה. לא יינתן ניקוד על תשובות ללא נימוק.

ז. קראו את כל המבחן לפני שאתם מתחילים לענות על השאלות.

ח. **את התשובות לשאלות יש לרשום על דפים. בתום המבחן יש לסרוק את הפתרון, להעלות אותו כקובץ PDF לאתר הקורס ולשלוח אותו לתיבת המייל של הקורס.**

בהצלחה!

שאלה 1 (25 נק'): שלבי הקומפילציה

שני חלקי השאלה מתייחסים לשפת FanC שהופיעה בתרגילי הבית.

חלק א - סיווג מאורעות (10 נק')

נתון קטע הקוד הבא בשפת FanC:

```
enum days {Sunday, Monday, Tacoday, Wednesday, Thursday,
           Friday, Saturday};

int bar(enum days d2, int p, int y) {
    enum days z = Thursday;
    if ((int) d2 == (int) Tacoday) {
        print("It's Taco Tueeeesday!");
        d2 = Wednesday;
    }
    while (p < 42) {
        p = p + 3;
    }
    printi((int) z);
    return p + y;
}

void main() {
    print("What day is it?");
    enum days d1 = Tacoday;
    printi(bar(d1, 3, 3));
    enum days w = Sunday;
    printi(bar(w, 2, 3));
    int t; int m; int k;
    k = 7;
    printi(bar(Monday, 2, m));
    m = 42;
    t = bar(Friday, k, 1);
    k = bar(w, k, t);
    printi((int) w);
    return;
}
```

בסעיפים הבאים מוצגים שינויים (בלתי תלויים) לקוד של התוכנית. עבור כל שינוי כתבו האם הוא גורם לשגיאה. אם כן, ציינו את השלב המוקדם ביותר שבה נגלה אותה (ניתוח לקסיקלי, ניתוח תחבירי, ניתוח סמנטי, ייצור קוד, זמן ריצה) ונמקו בקצרה:

א. מחליפים את שורה 10 בשורה הבאה:

```
10.     while(42 < p < 1337) {
```

ב. מחליפים את שורה 14 בשורה הבאה:

```
14.     return bar(d2, p+1, y-1);
```

ג. מחליפים את שורה 23 בשורה הבאה:

```
23.     int t, m, k;
```

ד. מחליפים את שורה 11 בשורה הבאה:

```
11.         p = p + 3.0;
```

ה. מחליפים את שורה 10 בשורה הבאה:

```
10.     while (not p < 42);
```

<u>חלק ב – הרחבת השפה (15 נק')</u>

א. (5 נק') הסבירו מה מיוחד בתוכנית הבאה, ומדוע איננה תקינה בשפת FanC:

```
bool isEven(byte n) {
    if (n == 0)
        return true;
    else
        return isOdd(n-1);
}

bool isOdd(byte n) {
    if (n == 0)
        return false;
    else
        return isEven(n-1);
}

void main() {
    bool x;
    x = isEven(42b);
    return;
}
```

ב. (10 נק׳) הציעו הרחבה לשפת FanC, שבאמצעותה ניתן יהיה להפוך את התוכנית מסעיף א׳ לתקינה על ידי הוספה של **שורה אחת בלבד**.

i. פרטו בקצרה איזה שינוי צריך להתבצע בכל שלב בקומפילציית השפה. **התייחסו לשלבים לקסיקלי, תחבירי, סמנטי, ייצור קוד אסמבלי (שפת ביניים)**. הקפידו על ההפרדה בין השלבים. יש להקפיד על פתרון **יעיל**.

ii. בהינתן ההרחבה שהצעתם לשפת FanC בסעיף ב׳, מהו השינוי לתוכנית מסעיף א׳ אשר יהפוך אותה לתקינה? זכרו כי על השינוי להיות הוספה של שורה אחת בלבד.

שאלה 2 (25 נק׳): ניתוח תחבירי וסמנטי

נתון הדקדוק הבא:

1. $S \rightarrow L \ \underline{.} \ L$
2. $S \rightarrow L$
3. $L \rightarrow L \ B$
4. $L \rightarrow B$
5. $B \rightarrow \underline{0}$
6. $B \rightarrow \underline{1}$

דקדוק זה מגדיר מספרים בינאריים שלמים ולא שלמים.

אין קשר בין שני סעיפי השאלה.

א. (12 נק׳) האם הדקדוק שייך למחלקה SLR?

ב. (13 נק׳) נרצה שלכל מילה בשפה תהיה התכונה הסמנטית val, המחזיקה את הערך העשרוני של המילה. לדוגמה, עבור המילה 11 התכונה val תכיל את המספר העשרוני 3 ועבור המילה 101.101 התכונה val תכיל את המספר העשרוני 5.625.
נדגים את אופן חישוב הערך העשרוני של המספר הבינארי 101.101:

לקסמה	1	0	1	.	1	0	1
משמעות הלקסמה	$1 * 2^2$	$0 * 2^1$	$1 * 2^0$		$1 * 2^{-1}$	$0 * 2^{-2}$	$1 * 2^{-3}$
ערך עשרוני מתאים ללקסמה	4	0	1		0.5	0	0.125

כתבו כללים סמנטיים לתמיכה בתכונה הסמנטית לעיל כך שהדקדוק הנוצר יהיה L-attributed.
הדרכה: תוכלו להגדיר תכונה סמנטית המתארת את המיקום ביחס לנקודה העשרונית.

הנחיות:

- אין לשנות את הדקדוק.
- ניתן להוסיף למשתנים תכונות סמנטיות כרצונכם. עבור כל תכונה סמנטית שתוסיפו **יש לציין את משמעותה והאם היא נוצרת או נורשת.**
- אין להשתמש במשתנים גלובליים.
- יש לכתוב את הכללים הסמנטיים במלואם.

בהצלחה!

נוסחאות ואלגוריתמים

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

Top Down

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

$$\text{follow}(A) = \{t \in T \cup \{\$\} \mid S\$ \Longrightarrow^* \alpha A t \beta \ \wedge \ \alpha \in (V \cup T)^* \ \wedge \ \beta \in (V \cup T)^*(\varepsilon \mid \$)\}$$

$$\text{select}(A \to \alpha) = \begin{cases} \text{first}(\alpha) \cup \text{follow}(A) & \alpha \Longrightarrow^* \varepsilon \\ \text{first}(\alpha) & \text{otherwise} \end{cases}$$

הגדרה: דקדוק G הוא LL(1) אם ורק אם לכל שני כללים ב- G השייכים לאותו משתנה A מתקיים:

$$\text{select}(A \to \alpha) \cap \text{select}(A \to \beta) = \emptyset$$

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

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

אלגוריתם מנתח 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

פריט LR(0) הוא $(A\to\alpha\bullet\beta)$ כאשר $A\to\alpha\beta \in P$
סגור (closure) על קבוצת פריטים I מוגדר באופן אינדוקטיבי:
- בסיס: $closure(I) = I$.
- צעד: אם $(A\to\alpha\bullet B\beta) \in closure(I)$, אז לכל $B\to\gamma \in P$, גם $(B\to\bullet\gamma) \in closure(I)$

פונקציית המעברים של האוטומט:
$$\delta(I, X) = \cup \{closure(A\to\alpha X\bullet\beta) \mid (A\to\alpha\bullet X\beta) \in I\}$$

פריט LR(1) הוא $(A\to\alpha\bullet\beta\ ,\ t)$ כאשר $A\to\alpha\beta\in P$, $t\in T\cup\{\$\}$
סגור (closure) על קבוצת פריטים I מוגדר באופן אינדוקטיבי:
- בסיס: $closure(I) = I$.
- צעד: אם $(A\to\alpha\bullet B\beta\ ,\ t) \in closure(I)$, אז לכל $B\to\gamma \in P$ ולכל $x \in first(\beta t)$, גם $(B\to\bullet\gamma\ ,\ x) \in closure(I)$

פונקציית המעברים של האוטומט:
$$\delta(I, X) = \cup \{closure(A\to\alpha X\bullet\beta, t) \mid (A\to\alpha\bullet X\beta, t) \in I\}$$

הגדרת טבלת action למנתח SLR:

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

הגדרת טבלת action למנתח LR(1):

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

הגדרת טבלת goto למנתח SLR ו- 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 ביניים

סוגי פקודות בשפת הביניים:

1. משפטי השמה עם פעולה בינארית
2. משפטי השמה עם פעולה אונרית
3. משפטי העתקה
4. קפיצה בלתי מותנה
5. קפיצה מותנה
6. הדפסה

```
x := y op z
x := op y
x := y
goto L
if x relop y goto L
print x
```

Data-Flow Analysis

ההגדרות מתייחסות ל-CFG: $G = (V, E)$

הצורה הכללית של המשוואות לסריקה קדמית:

$$\text{in}(B) = \bigcap_{(S,B)\in E} \text{out}(S) \quad \text{או} \quad \text{in}(B) = \bigcup_{(S,B)\in E} \text{out}(S)$$
$$\text{out}(B) = f_B\big(\text{in}(B)\big)$$

הצורה הכללית של המשוואות לסריקה אחורית:

$$\text{in}(B) = \bigcap_{(B,D)\in E} \text{out}(D) \quad \text{או} \quad \text{in}(B) = \bigcup_{(B,D)\in E} \text{out}(D)$$
$$\text{out}(B) = f_B\big(\text{in}(B)\big)$$

ייצור קוד בשיטת Backpatching

פונקציות:

`makelist(quad)`	יוצרת רשימה ריקה עם איבר אחד (ה"חור" quad).
`merge(list1, list2)`	מחזירה רשימה ממוזגת של הרשימות list1, list2
`emit(code_string)`	מדפיסה קוד בשפת הביניים ומאפשרת להדפיס פקודות קפיצה עם "חורים".
`nextquad()`	מחזירה את כתובת הרביעיה (הפקודה) הבאה שתצא לפלט.
`backpatch(list, quad)`	מקבלת רשימת "חורים" list וכתובת quad ו"מטליאה" את הרשימה כך שבכל החורים תופיע הכתובת quad.
`newtemp()`	מחזירה שם של משתנה זמני חדש שאינו נמצא בשימוש בתכנית.

משתנים סטנדרטיים:

- S: גוזר פקודות (statements) בשפה. תכונות:
  - nextlist: רשימת כתובות של פקודות המכילות חור שיש להטליא בכתובת הפקודה הבאה לביצוע אחרי הפקודה הנגזרת מ- S.
- B: גוזר ביטויים בוליאניים. תכונות:
  - truelist: רשימת כתובות של פקודות המכילות חור שיש להטליא בכתובת אליה יש לקפוץ אם הביטוי הבוליאני מתקיים.
  - falselist: רשימת כתובות של פקודות המכילות חור שיש להטליא בכתובת אליה יש לקפוץ אם הביטוי הבוליאני אינו מתקיים.
- E: גוזר ביטויים אריתמטיים. תכונות:
  - place: שם המשתנה הזמני לתוכו מחושב הביטוי האריתמטי.
- M: מרקר שמטרתו שמירת כתובת פקודה. תכונות:
  - quad: כתובת הפקודה הבאה מיד לאחר המרקר.
- N: מרקר שמטרתו דילוג על קטע קוד המוסיף קוד קפיצה שיעדה עדיין לא ידוע. תכונות:
  - nextlist: רשימת כתובות המכילה את מיקום פקודת הקפיצה שנכתבה ע"י N.

שפת FanC

אסימונים:

אסימון	תבנית
VOID	void
INT	int
BYTE	byte
B	b
BOOL	bool
ENUM	enum
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	}
ASSIGN	=
RELOP	== | != | < | > | <= | >=
BINOP	+ | - | * | /
ID	[a-zA-Z][a-zA-Z0-9]*
NUM	0 | [1-9][0-9]*
STRING	”([^\n\r\”\\]|\\[rnt”\\])+”

דקדוק:

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

39. $ExpList \rightarrow Exp\ COMMA\ ExpList$
40. $Type \rightarrow INT$
41. $Type \rightarrow BYTE$
42. $Type \rightarrow BOOL$
43. $EnumType \rightarrow ENUM\ ID$
44. $Exp \rightarrow LPAREN\ Exp\ RPAREN$
45. $Exp \rightarrow Exp\ BINOP\ Exp$
46. $Exp \rightarrow ID$
47. $Exp \rightarrow Call$
48. $Exp \rightarrow NUM$
49. $Exp \rightarrow NUM\ B$
50. $Exp \rightarrow STRING$
51. $Exp \rightarrow TRUE$
52. $Exp \rightarrow FALSE$
53. $Exp \rightarrow NOT\ Exp$
54. $Exp \rightarrow Exp\ AND\ Exp$
55. $Exp \rightarrow Exp\ OR\ Exp$
56. $Exp \rightarrow Exp\ RELOP\ Exp$
57. $Exp \rightarrow LPAREN\ Type\ RPAREN\ Exp$