03.07.2019

מבחן סוף סמסטר – מועד א'

מרצה אחראי: ד"ר בלהה מנדלסון

מתרגלים: אנטוניו אבו-נאסר, שקד ברודי, יעקב סוקוליק, יואב צוריאל

הוראות:

א. בטופס המבחן 16 עמודים מהם עמוד אחד ריק ו6 דפי נוסחאות. בדקו שכל העמודים ברשותכם.

ב. משך המבחן שלוש שעות (180 דקות).

ג. אסור כל חומר עזר חיצוני.

ד. במבחן 5 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. אין צורך להגיש את הטופס בתום הבחינה.

ט. את התשובות לשאלות יש לרשום במחברת הבחינה בלבד.

בהצלחה!

שאלה 1 (20 נק'): שלבי הקומפילציה

שני חלקי השאלה מתייחסים לשפת FanC שהופיעה בתרגילי הבית.

חלק א - סיווג מאורעות (10 נק')

נתון קטע הקוד הבא בשפת FanC:

```
1.  bool foo(int x, int y)
2.  {
3.        return y * (x / y) == x;
4.  }
5.
6.  bool bar(int i, int x, int y)
7.  @pre(i > 0)
8.  {
9.      while(i >= 0)
10.     {
11.         if (not (foo(i, x) and foo(i, y)))
12.             continue;
13.         printi(i);
14.         i = i - 1;
15.     }
16. }
17.
18. void main()
19. {
20.     bar(100, 4, 5);
21. }
```

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 20 בשורה הבאה:

```
20. bar(100, 4, 5, 6);
```

ב. מחליפים את שורה 7 בשורה הבאה:

```
7.  @pre(n < 256b)
```

ג. מחליפים את שורה 12 בשורה הבאה:

```
12. break;
```

ד. מחליפים את שורה 3 בשורה הבאה:

```
3.  return y * ((x / y) == x);
```

ה. מוחקים את שורה 12.

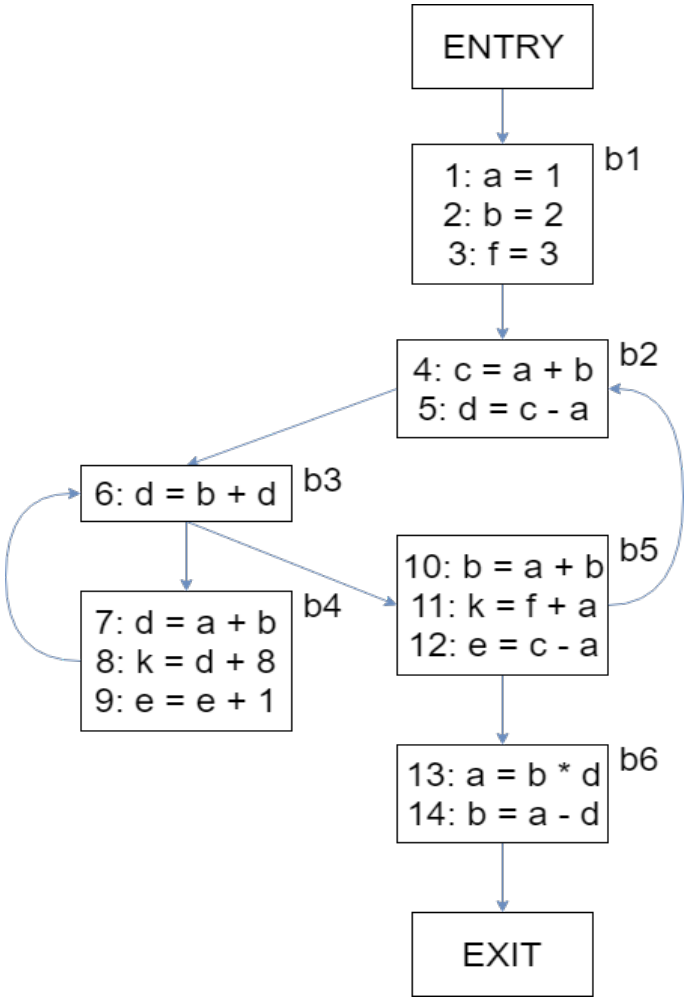
חלק ב – הרחבת השפה (10 נק')

הנכם מתבקשים להוסיף **לשפת FanC** יכולת חדשה. קראו את תיאור היכולת, ופרטו 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**.

נרצה להוסיף לשפת FanC תמיכה במצביעים. בדומה לשפת C, המתכנת יוכל להגדיר משתנה כמצביע למשתנים מטיפוס כלשהו. הערך של משתנה מסוג מצביע יהיה הכתובת בזכרון בה שמור המשתנה מהטיפוס אליו הוא מצביע. הגדרת משתנה כמצביע לטיפוס X תתבצע באמצעות הגדרת המשתנה באמצעות הטיפוס X*.
ניתן לבצע dereferencing למצביע, כלומר לקרוא את הערך הנמצא בכתובת השמורה במצביע באמצעות האופרטור האונרי * שפועל על טיפוסים מסוג מצביע בלבד. הפעלת האופרטור * על משתנה מטיפוס X* תחזיר ערך מטיפוס X.
ניתן לקבל כתובת של ערך שנמצא בזיכרון באמצעות האופרטור האונרי &, שפועל על משתנים בלבד.
הפעלת האופרטור & על משתנה מטיפוס X תחזיר ערך מטיפוס X*.
מצביע מטיפוס X* יכול לקבל ערך אחר מאותו טיפוס או כתובת של משתנה מטיפוס X.
לדוגמה:

```
int x = 42;
int* y = &x;   // y stores x's address (points to x)
*y = 0;        // assigns to what y points at the value 0
printi(x);     // prints 0
```

שאלה 2 (20 נקודות): אנליזה סטטית



עבור הגרף הזרימה (control flow graph) הנ"ל

א. (8 נק') חשבו את הקבוצות gen ו- kill לכל אחד מהבלוקים b1-b6 עבור אנליזת חיות.

ב. (8 נק') חשבו את הקבוצות IN ו- OUT לכל אחד מהבלוקים b1-b6. **הניחו** שכל המשתנים **חיים** בסוף התכנית.

ג. (4 נק') השתמשו בקבוצות שחישבתם בסעיף 2 ובצעו את אלגוריתם להסרת קוד מיותר (useless code elimination) להסרת פקודות שאין בהן שימוש.

שאלה 3 (20 נקודות): אופטימיזציות

אופטימיזציה חשובה שלא דיברנו עליה בקורס היא אופטימיזציה מסוג ***method inlining***, בה קריאה לפונקציה מוחלפת בגוף הפונקציה.

למשל עבור התוכנית הבאה:

```
1. int power(int x) {
2.     return x * x;
3. }
4.
5. int main(){
6.     int y = 3;
7.     int yy = power(y);
8.     return 0;
9. }
```

אם נבצע ***inlining*** למתודה **power**, לאחר האופטימיזציה נקבל את הקוד הבא:

```
1. int main(){
2.     int y = 3;
3.     int yy = y * y;
4.     return 0;
5. }
```

א. (3 נק') כיצד אופטימיזציה מסוג זה גורמת לתוכנית לרוץ מהר יותר?
ב. (3 נק') לאופטימיזציה זו מספר חסרונות. ציינו לפחות חסרון אחד.
ג. (5 נק') אילו פונקציות מתאימות לביצוע ***inlining***? ציינו מאפיין אחד לפחות (של הפונקציה או של התוכנית).
ד. (5 נק') אילו פונקציות אינן מתאימות לביצוע ***inlining***? ציינו מאפיין אחד נוסף שלא הזכרתם בסעיפים הקודמים.
ה. (4 נק') נניח כעת כי לא חשובים לנו היתרונות והחסרונות של האופטימיזציה.
אנו מעוניינים לדעת עבור כל פונקציה האם ניתן לבצע לה ***inlining*** (רמז: הדבר לא תמיד אפשרי) הציעו בקצרה אלגוריתם שמטרתו לקבוע עבור כל פונקציה האם ניתן לבצע לה ***inlining*** או לא.

שאלה 4 (20 נק'): ניתוח תחבירי וסמנטי

נתון הדקדוק $G = (V = \{ S, L \},\ T = \{\underline{item}, \underline{comma}, \underline{and}\},\ P,\ S)$ עם כללי הגזירה הבאים:

$$P:\quad S \rightarrow \underline{\text{item}} \mid L\ \underline{\text{and}}\ \underline{\text{item}}$$
$$L \rightarrow \underline{\text{item}}\ \underline{\text{comma}}\ L \mid \underline{\text{item}}$$

הדקדוק הנ"ל גוזר לדוגמא את המחרוזות הבאות:

item
item and item
item comma item comma item and item

א. (8 נק') האם הדקדוק G שייך למחלקה $LALR(1)$? הוכיחו את תשובתכם.

נרצה לכתוב מנתח לקסיקלי בעזרת התוכנה flex עבור האסימונים item, comma, and, כך שהאסימון comma יותאם ללקסמה שמכילה רק פסיק, האסימון and יותאם ללקסמה and, והאסימון item יותאם לכל מחרוזת אחרת המורכבת מאותיות קטנות באנגלית בלבד. נקרא ללקסמה של item "פריט".

ב. (4 נק') האוורד, סטודנט בהארוורד, הציע לעזור לנו בבניית המנתח, אך לצערנו כשהרצנו את flex על הקוד שהוא סיפק, קיבלנו תוצאות לא נכונות. הסבירו בקצרה מה הבעייתיות בקוד של האוורד:

```
%%
[a-z]+    return item;
,         return comma;
and       return and;
[\t\r\n ] /* ignore whitespace */
.         error();
```

לאחר שתי כוסות קפה, התעורר האוורד וסיפק לנו קוד שעובד. לכן, דרשנו ממנו לבנות מנתח תחבירי וסמנטי בתוכנת bison. אנו רוצים שבסיום הרצת המנתח על קלט תקין תודפס רשימת הפריטים, כלומר הלקסמות המתאימות לitem, בסדר הפוך מסדר הופעתם בקלט, ללא הדפסת פסיקים והמילה and. לדוגמא, עבור הקלט הבא:

marshmallow, caramel and chocolate

יודפס הפלט הבא:

chocolate
caramel
marshmallow

האוורד עוד לא הגיב, לכן אנו זקוקים לעזרתכם.

ג. (8 נק') השלימו את הכללים הסמנטיים המתאימים לכל כלל גזירה בדקדוק כך שבעת ביצוע reduce למשתנה S יודפס הפלט הנדרש. על הכללים הסמנטיים <u>לא לחרוג מ- 5 שורות</u> לכל כלל

גזירה. אין להוסיף מרקרים או כללים חדשים לדקדוק. השתמשו רק בתכונות נוצרות, ולכל תכונה הסבירו את משמעותה (למה צריכים אותה, מה היא מחזיקה ואיך משתמשים בה) וציינו את טיפוס התכונה.

```
S : item    { … }
   |  L and item { … }
;
L : item    { … }
   | item comma L2 { … }
;
```

שאלה 5 (20 נקודות): ייצור קוד

א. (6 נק') ציינו שתי סיבות מדוע חשוב שמתכנן הקומפיילר יפעל לפי מוסכמות קריאה (ABI, calling conventions) ידועות מראש.

ב. (14 נק') לפניכם הקוד הבא שכתוב בשפת FanC:

```
1. int baz(int x, bool y){
2.     if(y)
3.         return 256 * x;
4.     else
5.         return x;
6. }
7.
8. int bar(int x){
9.     return 256 - x;
10. }
11.
12. int foo(int x, int y){
13.     int r;
14.     if(x >= bar(y))
15.         r = baz(x, true);
16.     else
17.         r = baz(y, false);
18.     return r + x;
19. }
20.
21. void main(){
22.     int x = 5;
23.     int y = 8;
24.     printi(foo(x, x + y));
25. }
```

הקוד עבר קומפילציה עם קומפיילר שמוסכמות הקריאה (ABI, calling conventions) שלו לא ידועות (לאו דווקא אלה שלמדתם בכיתה).

לפניכם מחסנית עם חלקים חסרים בה. בתוך המחסנית ניתן לכתוב כתובות אחרות למחסנית (על ידי 0d ואז הכתובת), ערכים מספרים/בוליאניים של משתנים, או כתובות חזרה של פונקציה (את זה יש לכתוב על ידי "חזרה לשורה..." ואז מספר השורה אליה חוזרים). שימו לב שאין צורך לשמור רגיסטרים כלל.

הסיקו מהמחסנית את מוסכמות הקריאה של הקומפיילר והסבירו עליהן. בפרט התייחסו איפה עוברים הפרמטרים לפונקציה ובאיזה סדר, איך נראית רשומת ההפעלה של פונקציה ומה סדר המשתנים הלוקליים.

בנוסף, מלאו את החלקים החסרים במחסנית עבור תוכנית שנעצרה בשורה 3 או 5 (הניחו כי הפונקציה הקוראת מנקה את הפרמטרים).

שימו לב: main היא לא באמת הפונקציה הראשונה שרצה, ולכן יש מי שקורא לה.

כתובת המחסנית	ערך
0d2000	
0d1996	כתובת החזרה של main
0d1992	8
0d1988	
0d1984	0d2000
0d1980	
0d1976	13
0d1972	חזרה לשורה 24
0d1968	
0d1964	
0d1960	
0d1956	false
0d1952	
0d1948	
0d1944	
0d1940	
0d1936	

בהצלחה!

נוסחאות ואלגוריתמים

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

Top Down

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

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

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

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

$$select(A \to \alpha) \cap select(A \to \beta) = \varnothing$$

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

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

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:

$$\delta(I, X) = \bigcup \left\{ closure(A \to \alpha X \bullet \beta) \mid (A \to \alpha \bullet X\beta) \in I \right\}$$

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:

$$\delta(I, X) = \bigcup \left\{ closure(A \to \alpha X \bullet \beta, t) \mid (A \to \alpha \bullet X\beta, t) \in I \right\}$$

הגדרת טבלת action למנתח SLR:

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

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	**x := y op z**
2. משפטי השמה עם פעולה אונרית	**x := op y**
3. משפטי העתקה	**x := y**
4. קפיצה בלתי מותנה	**goto L**
5. קפיצה מותנה	**if x relop y goto L**
6. הדפסה	**print x**

Data-Flow Analysis

ההגדרות מתייחסות ל – CFG: G=(V,E).

הצורה הכללית של המשוואות בחישוב סריקה קדמית:

$$in(B) = \bigcap out(S) \text{ או } in(B) = \bigcup out(S)$$
$$out(B) = f_B(in(B))$$

הצורה הכללית של המשוואות בחישוב סריקה אחורית:

$$out(B) = \bigcap in(S) \text{ או } out(B) = \bigcup in(S)$$
$$in(B) = f_B(out(B))$$

שפת FanC

אסימונים:

אסימון	תבנית
VOID	void
INT	int
BYTE	byte
B	b
BOOL	bool
AND	and
OR	or
NOT	not
TRUE	true
FALSE	false
RETURN	return
IF	if
ELSE	else
WHILE	while
BREAK	break
CONTINUE	continue
PRECOND	@pre
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. $Program \rightarrow Funcs$
2. $Funcs \rightarrow \epsilon$
3. $Funcs \rightarrow FuncDecl\ Funcs$
4. $FuncDecl \rightarrow$
   $RetType\ ID\ LPAREN\ Formals\ RPAREN\ PreConditions\ LBRACE\ Statements\ RBRACE$
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. $PreConditions \rightarrow \epsilon$
13. $PreConditions \rightarrow PreConditions\ PreCondition$
14. $PreCondition \rightarrow PRECOND\ LPAREN\ Exp\ RPRAEN$
15. $Statements \rightarrow Statement$
16. $Statements \rightarrow Statements\ Statement$
17. $Statement \rightarrow LBRACE\ Statements\ RBRACE$
18. $Statement \rightarrow Type\ ID\ SC$
19. $Statement \rightarrow Type\ ID\ ASSIGN\ Exp\ SC$
20. $Statement \rightarrow ID\ ASSIGN\ Exp\ SC$
21. $Statement \rightarrow Call\ SC$
22. $Statement \rightarrow RETURN\ SC$
23. $Statement \rightarrow RETURN\ Exp\ SC$
24. $Statement \rightarrow IF\ LPAREN\ Exp\ RPAREN\ Statement$
25. $Statement \rightarrow IF\ LPAREN\ Exp\ RPAREN\ Statement\ ELSE\ Statement$
26. $Statement \rightarrow WHILE\ LPAREN\ Exp\ RPAREN\ Statement$
27. $Statement \rightarrow BREAK\ SC$
28. $Statement \rightarrow CONTINUE\ SC$
29. $Call \rightarrow ID\ LPAREN\ ExpList\ RPAREN$
30. $Call \rightarrow ID\ LPAREN\ RPAREN$
31. $ExpList \rightarrow Exp$
32. $ExpList \rightarrow Exp\ COMMA\ ExpList$
33. $Type \rightarrow INT$
34. $Type \rightarrow BYTE$
35. $Type \rightarrow BOOL$
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. $Exp \rightarrow Exp\ RELOP\ Exp$