27.2.2015ô

מבחן סוף סמסטר – מועד ב'

מרצה אחראי: פרופ' ארז פטרנק

מתרגלים: עדי סוסנוביץ יורי משמן

הוראות:

א. בטופס המבחן 9 עמודים מהם 4 דפי נוסחאות. בדקו שכל העמודים ברשותכם.

ב. משך המבחן שלוש שעות (180 דקות).

ג. אסור כל חומר עזר פרט לדף הנוסחאות המצורף לבחינה.

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 (16 נקודות): סיווג מאורעות

האם המאורעות הבאים קורים בזמן קומפילציה, בזמן ריצה או בזמן בניית הקומפיילר? אם מדובר על זמן קומפילציה כתבו באיזה שלב של הקומפילציה המאורע קורה. נמקו בקצרה.

1. התוכנית מגלה שביטוי לוגי מקבל ערך FALSE.
2. משתנה מוחלף בערך קבוע שתמיד מוצב אליו.
3. מתגלה ששורה מסויימת היא הערה ואין צורך להריץ שורה זו.
4. נוצר קוד ביניים עבור פקודת השמה.
5. משתמשים בסילוק רקורסיה שמאלית כדי שהדקדוק יהפוך לנוח יותר לניתוח סינטקטי.
6. קוד שלעולם לא מתבצע מושמט מהתוכנית.
7. מתגלה השמה של קבוע מספרי אל משתנה מסוג מחרוזת.
8. מתגלה ששכחו לסגור הגדרת פונקציה (למשל עם סוגר מסולסל כמו ב-C).

שאלה 2 (16 נקודות + 4 נקודות בונוס): Copying Garbage Collection

כזכור, אלגוריתם זה שומר על חצי מה-heap ריק, בעת האיסוף הוא מעתיק את האובייקטים החיים לאיזור הריק וממשיך משם לעבוד עם האיזור אליו הועתקו האובייקטים, בעוד האיזור שבו היו האובייקטים קודם הופך לאיזור שנשמר ריק. וחוזר חלילה.

א. (3 נק') אם נצליח לשנות את האלגוריתם כך שהחלק השמור (to-space) יתפוס פחות מחצי heap. כיצד יועיל השינוי?

ב. (3 נק') מדוע באלגוריתם המקורי גודל ה- to-space חייב להיות לפחות חצי מגודל ה-heap?

ג. (10 נק') נרצה לחסוך מקום ע"י חלוקת ה- heap לשלושה חלקים שווים. חלק אחד יישמר ריק (to-space). בעת האיסוף, נרצה להשתמש באלגוריתם ההעתקה הרגיל לעבור על שליש אקטיבי אחד של ה-heap (שייקרא from-space-1) ולהעביר את האובייקטים החיים ממנו אל השליש הריק (אל to-space). הבעיה היא שלא ניתן לדעת מי חי ב- from-space-1 כי חלק מהמצביעים אל אובייקטים אליו נמצאים באזור האקטיבי השני: from-space-2.

a. (4 נק) הציעו דרך לדעת מי האובייקטים בכל איזור המצביעים אל איזור אחר. נסו להיות יעילים.

b. (3 נק) מה המחיר במקום (כלומר כמות זיכרון שנתפס) ובזמן (כלומר בביצועים) של הפיתרון שהצעתם?

c. (4 נק) תארו תוכנית (מספיק לתאר פסאודו קוד, או אפילו רק את התצורה של ה-heap שהתוכנית יוצרת) שעבורה השיטה שהצעתם תגבה מחיר יקר של מקום או מחיר יקר של זמן (או שניהם). נסו לתכנן תוכנית "גרועה" ככל האפשר עבור השיטה שהצעתם.

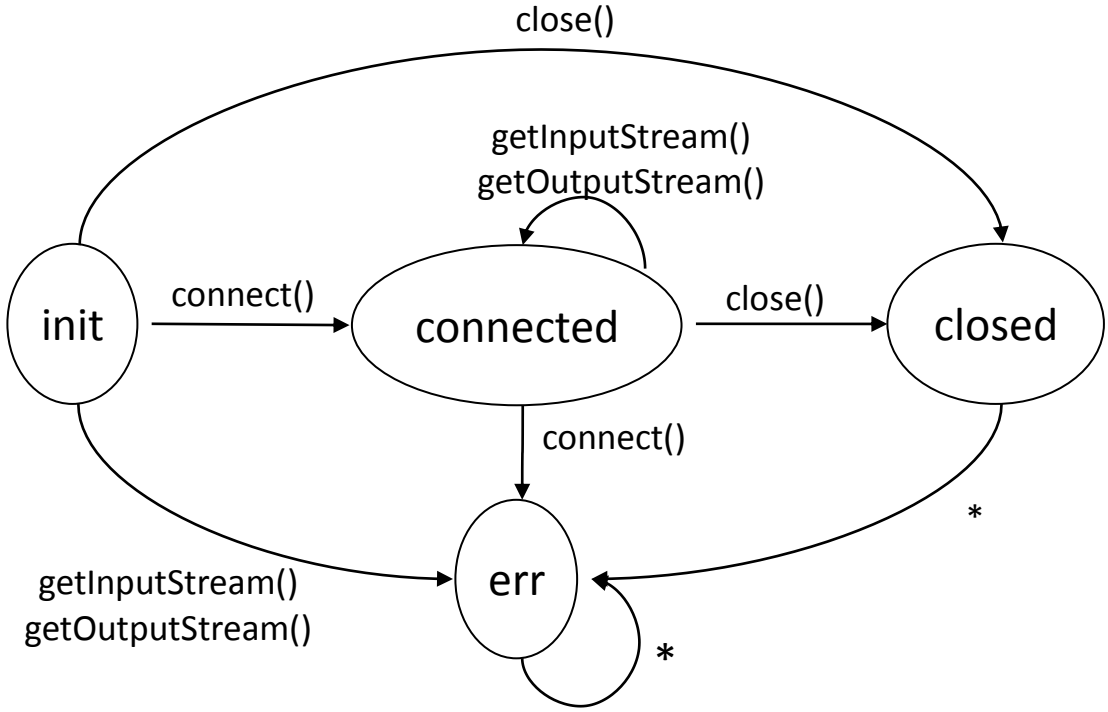
d. (4 נק' בונוס): האם בשיטה שהצעתם תיתכן ריצה שבה יהיה אובייקט לא נגיש שלא נוכל לאסוף לעולם? או שבסופו של דבר נוכל לאסוף כל אובייקט? נמקו. (בשאלת בונוס זו ניחוש ללא נימוק לא ייזכה בנקודות, וגם רישום "לא יודע" לא יזכה בנקודות הבונוס.)

שאלה 3 (24 נקודות): שאלת DFA (אנליזה סטטית)

בתרגיל זה נרצה לממש גרסה של אנליזה הנקראת **type-state-analysis–המנסה להבין לכל אובייקט מה ההתנהגות שלו בתוכנית.**
תוכנית לדוגמה:

```
11. s1 = new Socket();
12. s2 = new Socket();
13. s1.connect()
14. s2.connect()
15. while(s1.getInputStream(&a) ){
16.         s2.getOutputStream().write(a.first());
17. }
18. s1.close()
19. s2.close();
20. s1.getInputStream(&a)
```

בתוכנית זו (ובאופן כללי באנליזה הנדרשת) אנחנו מתעניינים רק באובייקטים מסוג Socket().
אנחנו מעוניינים לדעת בכל נקודה מה המצב של האוביקט.
כל אוביקט מתחיל ממצבinit ובהתאם למתודות שמופעלות עליו עובר למצבים אחרים.
סט המצבים של האוביקט והמעברים בין המצבים כתוצאה ממתודות מוגדר באמצעות הגרף הבא:



קשת המסומנת בכוכבית (*) משמעה כי לקריאת כל מתודה על האוביקט המעבר ממצב זה יהיה למצב המטרה של הקשת. (לדוגמא ממצב closed עבור הפעלת כל מתודה שהיא, מצב האוביקט עובר ל err)

<u>הנחות</u>

1. ניתן להניח כי קיימים רק אוביקטים מטיפוס Socket().
2. ניתן להניח כי לכל משתנה בתוכנית מושם אוביקט יחיד בתחילת התוכנית והוא לא מקבל אוביקטים אחרים במהלך התוכנית.
3. ניתן להניח כי בכל שורה בתוכנית ישנם רק משתנים שמקבלים טיפוס Socket (ניתן להתעלם ממשתנים דוגמת a בדוגמה הנתונה)
4. ניתן להניח שלא מופעלות על Socket מתודות אחרות מאלו המופיעות בגרף.
5. הנקודות המעניינות אותנו הן הנקודות שבתחילת וסוף בלוק בסיסי.

שאלות לדוגמה עליהן נרצה שהאנליזה תאפשר לנו לענות:

(*) אילו אוביקטים בנקודה מסויימת עלולים להיות במצב שגיאה (err)

(**) אילו אוביקטים בהכרח מחוברים בנקודה מסויימת בקוד (במצב connected)

א. (3 נק׳)עבור הדוגמא הנתונה, חלקו את הקוד לבלוקים ורישמו את ה-control flow graph.

ב. (3 נק׳) רשמו בכל נקודה בתוכנית (ולא רק בתחילת ובסוף בלוק) מה המצב של כל אחד מהאוביקטים. ניתן להתייחס אל האוביקטים בעזרת שמות המשתנים שמצביעים אליהם (בדוגמא: s1 ו s2).

ג. (10 נק׳) הציעו אנליזת DFA שתוכל לענות על (*).יש לציין מהם פריטי המידע, האם האנליזה היא must/may, האם היא קדמית או אחורית, מהם GENו-KILLומהי פונקציה העידכון באיטרציה. בנוסף יש להגיד במפורש איך מתוצאות האנליזה אפשר לקבל את התשובה ל (*) בתחילת ובסוף בלוק.

ד. (3 נק׳) הסבירו מדוע מהאנליזה של סעיף 3 לא ניתן לקבל תשובה ל(**) על ידי דוגמה קצרה (יחד עם תוצאות אנליזת סעיף ב׳ עליה) שבה תתקבל תשובה שגויה ל(**). יש לתת הסבר קצר ללמה לא ניתן לקבל את התשובה הנכונה מהפלט.

ה. (5 נק׳) תארו אנליזת DFAשתוכל לענות על (**).

<u>שאלה 4: ניתוח סמנטי (24 נקודות)</u>

בשאלה זו נדון במעבר בין צורות infix ו postfix של ביטויים אריתמטיים.
ביטוי אריתמטי בצורת **infix** נכתב בצורה הבאה: **A OP B**
ביטוי אריתמטי בצורת **postfix** נכתב בצורה הבאה: **A B OP**
בביטויים **בצורת postfix אין צורך בסוגריים.**
אלגוריתם החישוב של ביטויpostfix הוא באופן הבא: מתחזקים מחסנית ועוברים על הביטוי משמאל לימין. אם האלמנט הבא הוא מספר דוחפים אותו למחסנית. אם האלמנט הוא אופרטור מוציאים שני אלמנטים מהמחסנית, מחשבים את תוצאת הפעלת האופרטור עליהם, ודוחפים את התוצאה למחסנית. כשמסיימים לעבור על ביטוי הקלט המספר שנשאר במחסנית הוא תוצאת החישוב.

א) (10 נק׳) הציעו סכימת תרגום בשיטת תרגום מונחה תחביר שמתרגמת ביטויים אריתמטיים בצורת infix לביטויים בצורת postfix. יש להשתמש אך ורק בתכונות נוצרות.
לדוגמה: הקלט ״3*4+5*2״ יתורגם ל: ״ 3 4 * 5 2 * + ״, והקלט ״(3*4)״ יתורגם ל: ״3 4 *״.

על הפתרון לכלול דקדוק חסר הקשר מתאים, תכונות סמנטיות, וחוקים סמנטיים.
לכל תכונה סמנטית יש לפרט מאיזה טיפוס היא ומה משמעותה.
הראו כיצד מופעלת סכימת התרגום שהצעתם על הקלט "3*4+5*2".

הערות:

- הפלט צריך להיות מטיפוס מחרוזת שמייצגת את הבטוי האריתמטי השקול בצורת postfix.
- כתבו דקדוק פשוט ואינטואיטיבי ככל שניתן, עם מעט משתני גזירה ככל שניתן. מותר להשתמש בדקדוק דו משמעי, כך שניתן להתמודד עם הקונפליקטים באמצעות הגדרת עדיפויות ואסוציאטיביות, אך אין צורך להסביר כיצד בשאלה זו.
- מספיק לתמוך בפעולות כפל וחיבור בביטויים אריתמטיים.

ב) (14 נק') הציעו סכימת תרגום עבור ביצוע התרגום ההפוך מצורת postfix לצורת infix.
יש להוסיף סוגריים **רק** במקומות בהם הם נחוצים בצורת ה infix
<u>לדוגמה:</u>
הקלט " * 5 – 4 3" יתורגם ל: "(3-4)*5"
הקלט " * - 4 3 5 " יתורגם ל: "(3-4)*5"
הקלט " + 4 * 5 3" יתורגם ל: "3*5+4"
הקלט "++ 4 3 5 " יתורגם ל: "5+(3+4)"
הקלט " + 5 + 4 3" יתורגם ל: "3+4+5"

בסעיף זה מותר להשתמש גם בתכונות נורשות.
על הפתרון לכלול דקדוק חסר הקשר מתאים, תכונות סמנטיות, וחוקים סמנטיים.
עבור כל תכונה סמנטית ציינו האם היא נורשת או נוצרת והסבירו את תפקידה והטיפוס שלה.
הסבירו את נכונות הפתרון, והראו כיצד מופעלת סכימת התרגום שהצעתם על הקלט " * 5 – 4 3" .

<u>שאלה5: תרגום לשפת ביניים בשיטת תוויות נורשות (20 נקודות)</u>

בשאלה זו נדון בהרחבה לדקדוק הבסיסי שראינו בתרגולים.
הוחלט להוסיף לביטוי בוליאני כלל גזירה עם האופרטור XOR, מהצורה הבאה:

$B \rightarrow B_1 \; xor \; B_2$

ע"פ הסמנטיקה של האופרטור XOR, ערך האמת של B הוא TRUE אם ורק אם ערך האמת של B1 שונה מזה של B2.

א.(5 נק') הציעו פריסת קוד המתאימה לכלל הגזירה החדש, בשיטת **התוויות הנורשות**.
ב.(10 נק') כתבו סכימת תרגום בשיטת **התוויות הנורשות** המייצרת את פריסת הקוד שהצעתם בסעיף הקודם.
ג. (5 נק') הדגימו את קוד הביניים שייווצר ע"פ הסכימה שהצעתם על הבטוי $(a < b) \; xor \; (c > d)$. הניחו שכתובת הפקודה הראשונה בקוד שהינכם מייצרים היא 0, וכי הכתובות גדלות ב 1 בכל פקודה.

הנחיות:

- הקוד הנוצר יישמר בתכונה *code* ולא יתבצע *emit* ל *buffer*
- יש להשתמש רק בתכונות שנלמדו בשיטת התוויות הנורשות: *B.true,B.false,B.code*
- יש לבצע מספר מינימלי של קריאותלפונקציה *newtemp()*.

בהצלחה!!

נוסחאות ואלגוריתמים

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

Top Down

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

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

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

<u>הגדרה:</u> דקדוק G הוא LL(1) <u>אם ורק אם</u> 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 SHIFT
                else ERROR
        else    // X ∈ V
                if M[X , t] = error then ERROR
                else REPLACE(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
```

ניתוח סמנטי

אלגוריתם dfvisit לניתוח סמנטי עבור הגדרות L-attributed:

```
procedure dfvisit(node n) :
        foreach child m of n in left-to-right order do
                evaluate the inherited attributes of m
                dfvisit(m)
        end
        evaluate the synthesized attributes of n
```

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:
  - E.place: שם המשתנה הזמני לתוכו מחושב הביטוי האריתמטי.

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. פרמטרים וקריאה לפרוצדורות	`param x`
	`call p, n`
	`return y`

7. indexed assignments	`x := y [ i ]` `x [ i ] := y`

8. השמה של כתובות ומצביעים	`x := addr y`
	`x := * y`
	`* x := y`

Data-Flow Analysis

ההגדרות מתייחסות ל – CFG: G=(V,E).

הצורה הכללית של המשוואות בחישוב סריקה קדמית:

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

$$out(B) = f_B\big(in(B)\big)$$

הצורה הכללית של המשוואות בחישוב סריקה אחורית:

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

$$in(B) = f_B\big(out(B)\big)$$