27.06.2011

מבחן סוף סמסטר – מועד א'

מרצה אחראי: דר' ערן יהב

מתרגלים: רן צמח עדי סוסנוביץ' רן בן-בסט

הוראות:

א. **המבחן אנונימי! נא לרשום רק מספר זהות ולא את השם.**

ב. בטופס המבחן 6 עמודים. בדקו שכל העמודים ברשותכם.

ג. משך המבחן שלוש שעות (180 דקות).

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

ז. אין צורך להגיש את הטופס בתום הבחינה.

ח. את התשובות לשאלות יש לרשום במחברת הבחינה בלבד.

בהצלחה!

<u>שאלה 1 (10 נקודות): סיווג מאורעות</u>

סווגי את המאורעות הבאים לפי זמן ריצה, זמן קומפילציה, או זמן בניית הקומפיילר. פרטי בקצרה מתי בדיוק מתרחש כל מאורע ואילו נתונים נדרשים עליו על-ידי הקומפיילר או כותבת הקומפיילר, כלומר באילו מבני נתונים ואלגוריתמים משתמשים לצורך המאורע. **במידה ויש מספר תשובות נכונות, הסבירי בקיצור נמרץ (שתי שורות) את כולן**:

א. (2 נק') בקריאה לפונקציה x = foo() נמצא שהטיפוס של x אינו תואם את טיפוס ערך החזרה של הפונקציה.
ב. (2 נק') נמצא שהדקדוק עבור שפת התכנות הוא רב-משמעי.
ג. (2 נק') ניתן להקצות את המשתנה המקומי x והמשתנה המקומי y לאותו רגיסטר.
ד. (2 נק') השם $aloha אינו שם משתנה חוקי.
ה. (2 נק') פעולת החלוקה z = x / y גורמת לחילוק באפס.

<u>שאלה 2 (20 נקודות): אנליזה</u>

בשאלה זו נעסוק בניתוח תכנית ובה פעולות על מנעולים.
קטעים קריטיים הינם חלק מהותי בתכנות מקבילי. תהליך חייב לבצע פעולת LOCK לפני כניסה לקטע קריטי, ופעולת UNLOCK ביציאה מהקטע הקריטי, על מנת לקיים מניעה הדדית.
לשם פשטות, נניח כי:

- יש רק מנעול אחד במערכת, ולכן לפעולות LOCK, UNLOCK אין ארגומנטים.
- בבלוק בסיסי יש פקודה אחת בדיוק.
- כל התהליכים בתכנית מבצעים את אותו קטע קוד וניתן לנתח את הקוד **באופן סדרתי**. כלומר – בניתוח קטע הקוד **אין חשיבות** לעובדה שהוא מתבצע במקביל ו**לא נדרשת** אנליזה שמתחשבת בתהליכים אחרים.

על מנת שהתכנית תתבצע באופן נכון, נדרש שכל תהליך יקיים:

- פעולת LOCK מתבצעת רק אם המנעול לא ננעל על ידי התהליך קודם לכן או שהוא כבר שוחרר באמצעות פעולת UNLOCK (כלומר המנעול פנוי לפני נעילתו).
- לכל פעולת UNLOCK קדמה פעולת LOCK (המנעול היה נעול לפני ניסיון לשחררו).

עליך להציע ניתוח DFA אשר יאפשר מתן התרעות כדלהלן:

א. (4 נק') אזהרת LOCK→ LOCK: מתן אזהרה על פעולת LOCK אם היא עשויה לבוא לאחר פעולת LOCK אחרת (ללא UNLOCK ביניהן).
ב. (4 נק') אזהרת UNLOCK→ UNLOCK: מתן אזהרה על פעולת UNLOCK אם היא עשויה לבוא לאחר פעולת UNLOCK אחרת (ללא LOCK ביניהן).
ג. (7 נק') אזהרת EXIT→ LOCK: מתן אזהרה על פעולת LOCK אשר לאחריה עשויה התכנית להסתיים ללא ביצוע פעולת UNLOCK.
ד. (5 נק') אזהרת UNLOCK→ ENTRY: מתן אזהרה על פעולת UNLOCK אם היא עשויה להתבצע לפני שבוצעה פעולת LOCK כלשהי.

<u>הערות:</u>

- ניתן להגדיר כמה אלגוריתמי DFA נפרדים לצורך הפתרון.

- בכל אלגוריתם DFA יש לציין את פריטי המידע בקבוצות in(B), out(B) , את כיוון זרימת המידע, האם הבעיה היא מסוג may או must, מהן משוואות הזרימה, מהי הפונקציה f_B , וכיצד יאותחלו הקבוצות.
- יש להסביר כיצד יתבצע מתן האזהרות בהינתן הניתוח שהצעת.
- <u>להלן דוגמא:</u>

עבור קטע הקוד הבא:
(נניח כי " input x" היא פקודה לקבלת קלט במשתנה x):

```
0     Lock
1     input x
2     if(x>5) goto  4
3     goto 7
4     y=3
5     Lock
6     goto 10
7     y=1
8     Unlock
9     goto 10
10    y=2
11    Unlock
```

נדרש כי תהיה אזהרה מסוג *LOCK ® LOCK* בשורה 5, ואזהרה מסוג *UNLOCK ® UNLOCK* בשורה 11.

<u>שאלה 3 (30 נקודות): הרחבה לרשומות הפעלה וניהול זמן ריצה</u>

בשאלה זו הנך מתבקשת להסביר את השינויים הדרושים בקומפיילר על מנת לספק הגנת **זמן ריצה** מפני דריסת זכרון על ידי **buffer overflow**.
בשלב ראשון את נדרשת לתאר הגנה מפני דריסה של כתובת חזרה ברשומת ההפעלה על ידי גלישה של מערך.
לדוגמא, אם הקלט שהמשתמש מספק לתכנית הבאה ארוך מ 11 תווים, המערך buf יגלוש וידרוס את כתובת החזרה ברשומת ההפעלה.

```
void foo (char *x) {
  char buf[10];
  strcpy(buf, x);
}
int main (int argc, char *argv[]) {
  foo(argv[1]);
}
```

ברצוננו לממש מנגנון אשר יבדוק דריסה של כתובת החזרה **בזמן ריצה**. כלומר, כאשר מריצים את התכנית לעיל עם קלט ארוך מ 11 תווים, תתקבל הודעת שגיאה בזמן ריצה. **על הקומפיילר לייצר את הקוד המתאים לצורך כך**.
עני על הסעיפים הבאים ושימי לב:

- אנו מניחים ש**כל הטיפוסים** בשפה הם בגודל אחיד של מילה אחת בת 64 ביט.
- עליך לפרט במפורש כל הנחות נוספות שאת מבצעת.

א. (5 נק') הסבירי את מבנה רשומת ההפעלה בקריאה לפונקציה foo בדוגמא ומדוע קלט באורך של יותר מ 11 תווים ידרוס את כתובת החזרה. ניתן, ואף רצוי, להדגים את מבנה רשומת ההפעלה בתרשים.

ב. (10 נק') הסבירי את השינויים הנדרשים בקומפיילר ותארי את הפרטים הבאים:

- השינויים הדרושים ברשומת ההפעלה. יש לתאר במפורט את המבנה החדש של רשומת ההפעלה. ניתן להדגים זאת בתרשים.
- השינויים הנדרשים בקוד שמנהל את רשומות ההפעלה. יש לתאר במפורט את המבנה החדש של הקוד ברמה של פעולות מכונה. **אין צורך** להשתמש בסינטקס של שפת מכונה מסוימת.

הדגימי כיצד השינויים יאפשרו את גילוי השגיאה בדוגמא שלנו.

ג. (10 נק') ברצוננו להרחיב את המנגנון כך שיזהה **דריסה של מערך** כתוצאה מכתיבה למערך אחר. בשאלה זו הניחי שכתיבות גולשות הן תמיד רציפות, כלומר מתחילות במערך אחד וגולשות באופן רציף אל תוך מערך אחר. לשם פשטות אפשר להניח שכל הכתיבות מבוצעות באמצעות strcpy אך **אין לשנות** את הקוד של strcpy.

לדוגמא, בתכנית הבאה היינו רוצים לזהות **בזמן ריצה** שפעולת ה strcpy לתוך המערך buf2 גולשת ודורסת את המערך buf1.

```
void bar() {
  char buf1[10];
  char buf2[10];
  char str[22];
  strcpy(str,"Lorem ipsum dolor sit");
  strcpy(buf2, str);
  char x = buf1[0];
}
```

הסבירי את השינויים הנדרשים בקומפיילר ותארי את הפרטים הבאים:

- אם מניחים שמערכים מוקצים אך ורק על המחסנית - מתי ניתן להתריע על כך שכתיבה למערך אחד גולשת לתוך מערך אחר? כיצד נגלה זאת? באיזה חלקים של הקומפיילר נדרש שינוי?

- אם מניחים שמערכים יכולים להיות מוקצים גם באופן דינמי בזיכרון הערימה - מתי ניתן להתריע על כך שכתיבה למערך אחד גולשת לתוך מערך אחר? כיצד נגלה זאת? באיזה חלקים של הקומפיילר נדרש שינוי? **שימי לב שבמקרה זה נדרשים שינויים שאינם קשורים ברשומת ההפעלה ובקריאה/חזרה משגרה**.

ד. (5 נק') מדוע לדעתך לא מממשים מנגנון דומה לזה של סעיף ג' בשפות תכנות סטנדרטיות?

<u>שאלה 4 (20 נקודות): ניתוח תחבירי</u>

כזכור, מנתח SLR(1) הינו מנתח המשתמש באוטומט LR(0) וב-Lookahead של אות אחת, כאשר הרעיון הוא לעדן את טבלת המעברים (לתמוך ביותר דקדוקים) בעוד שלא נדרשים להרבה זיכרון נוסף (אין צורך לבנות אוטומט עם פריטי LR(1)).
בשאלה זו נעסוק בהכללה של רעיון הSLR ונבחן בניה של מנתח **SLR(2)** (כלומר מנתח המשתמש בLookahead של שני תווים, ובאוטומט הזהה לאוטומט LR(1).
הסיבה לבניה של מנתח כזה יכולה להיות תמיכה בדקדוק של שפה שלא ניתן לתת במנתח LALR/LR(1).

א. (5 נק') האם ניתן להשתמש באותו מבנה של טבלת הactions/goto (כלומר שורה לכל מצב ועמודה לכל טרמינל או סימן ה-$) עבור אוטומט SLR(2)?
אם כן, נמקי, אם לא, הסבירי מה צריך להיות השינוי בטבלאות.

ב. (10 נק') כזכור, במנתח SLR(1), שמנו פעולת reduce בתא המתאים למצב מסויים ואות lookahead אם במצב הופיע פריט מהצורה $\left(A\to\alpha\cdot\right)$ וגם התקיים כי אות הLookahead הייתה חלק מFollow(A).
בסעיף נעסוק בשאלה כיצד למקם את כללי הreduce בטבלה החדשה.
הערה: ניתן להשתמש בפונקציות first/follow/select לצורך החישובים בסעיף זה.
בסעיף זה **<u>אין</u>** צורך להוכיח נכונות.

i. נגדיר: $SingleTerminalWords\left(\alpha\right)=\left\{t\in T\mid\alpha\to^{*}t\right\}$
כלומר, קבוצת המילים באורך 1 הנגזרות מתבנית פסוקית α.
הסבירי כיצד ניתן לחשב את $SingleTerminalWords\left(\alpha\right)$ לכל תבנית פסוקית α.

ii. נגדיר: $2-first\left(\alpha\right)=\left\{t_1t_2\mid t_1,t_2\in T\ ,\ \alpha\to^{*}t_1t_2\beta\ ,\ \beta\in\left(V\cup T\right)^{*}\right\}$
כלומר, כל הרצפים של שני תווים היכולים להוות רישא של תבנית פסוקית הנגזרת מ α.
הסבירי כיצד ניתן לחשב את $2-first\left(\alpha\right)$ לכל תבנית פסוקית α.

iii. נגדיר:
$$2-follow\left(A\right)=\left\{t_1t_2\mid t_1,t_2\in\left(T\cup\{\$\}\right)\ ,\ S\$\to^{*}\alpha At_1t_2\beta\ ,\ \alpha\in\left(V\cup T\right)^{*}\ ,\ \beta\in\left(\left(V\cup T\right)^{*}\cdot\{\$,\varepsilon\}\right)\right\}$$
כלומר כל הרצפים של שני תווים היכולים להופיע **אחרי** משתנה A בגזירה כלשהי המתחילה מהמשתנה ההתחלתי, S.
הסבירי כיצד ניתן לחשב את $2-follow\left(A\right)$ לכל משתנה A.

iv. הסבירי היכן צריך למקם את כללי הreduce בטבלה של מנתח SLR(2).

ג. (5 נק') מהם יחסי ההכלה בין $LL\left(2\right),SLR\left(2\right)$ (כלומר האם אחד מוכל בשני, האם זרים או האם הם חופפים חלקית).

הסעיפים הבאים הם בונוס (לא יכולים לפגוע בציון הבחינה, **מומלץ לנסות לפתור רק אחרי שאר השאלות**).

ד. (3 נק') **בונוס**. תנו דוגמא לדקדוק G המקיים $G\in SLR\left(2\right)\wedge G\notin LR\left(1\right)$. יש לנמק **בקצרה**.

ה. (2 נק') **בונוס**. תנו דוגמא לדקדוק G המקיים $G\notin SLR\left(2\right)\wedge G\in LR\left(2\right)$. יש לנמק **בקצרה**.

ו. (5 נק') **בונוס**. תני דוגמא לשימוש אפשרי לSLR(2) (כלומר הרחיבו שפת תכנות מוכרת עם פקודה (או מבנה בקרה) חדשה אשר מתאפשרת לניתוח בSLR(2) ולא התאפשרה בLR(1).

שאלה 5 backpatching (20 נק׳)

נתון המבנה החדשני הבא לגזירת לולאות דו כיווניות:

Statement → bidirectionalSequence (Exp) { StatementsList }
StatementsList → StatementsList Statement ;
StatementsList → Statement ;
Statement → …

כאשר Statement משתנה דקדוק שגוזר כל אחת מהפקודות בשפה (את חלקי הדקדוק המתאימים לקריאות אילו לא נפרט כאן).

המבנה מתחיל בטרמינל bidirectionalSequence שלאחריו ביטוי אריתמטי מוקף בסוגריים, ולאחר מכן בלוק מוקף בסוגריים מסולסלות, ובו רצף של פקודות המופרדות ב ';' .
הקוד שאת מתבקשת ליצור בעת קומפילציה, יבצע בעת ריצה את הלוגיקה הבאה:

בעת תחילת ביצוע הקוד של המבנה הנ"ל, משוערך הביטוי האריתמטי אל משתנה זמני, נניח t. אם ערכו 0 אזי נצא מהמבנה אל הפקודה העוקבת. אם $t > 0$, נבצע ברצף את t הפקודות הראשונות בבלוק. אם לעומת זאת $t < 0$, אזי נבצע ברצף את (t-) הפקודות האחרונות בבלוק בכיוון הפוך – החל באחרונה.
ברגע שמבוצעות ברצף | t | הפקודות האמורות, שוב משוערך הביטוי האריתמטי, והכל חוזר חלילה.

א. הוסיפי לדקדוק את המרקרים M ו-N (המוכרים לך מהתרגולים) בכל מקום שדרוש כדי שניתן יהיה להעזר בתכונותיהם הידועות על מנת לפרוש את הקוד בשיטת ה- backpatching. תארי לכל הוספה מה מיקומה ומה ההצדקה לה.

ב. עבור משתנה הדקדוק StatementsList פרטי את כל התכונות (properties) הדרושות לו על מנת לפרוש את הקוד בשיטת ה- backpatching.

ג. הציעי פריסת קוד המתאימה לשיטת backpatching עבור מבנה הבקרה הנ"ל. על הפתרון לכלול שרטוט סכימטי ברור של המבנה על כל מרכיביו כולל קשתות הבקרה, וכן לכלול את הקוד שאמור להפרש בשפת הביניים שלמדנו בתרגולים.
קוד בשפות אחרות לא יתקבל. על הקוד הנוצר להיות יעיל ככל האפשר הן מבחינת זמן הריצה שלו והן מבחינת המקום בזיכרון שנדרש עבור התכונות הסמנטיות.

שימי לב:
1. אין להשתמש בכללים סמנטיים באמצע כלל גזירה.
2. אין להשתמש במשתנים גלובליים בזמן קומפילציה.
3. למשתנה Statement קיימים כללי גזירה פרט לכללים המוצגים בשאלה.
4. למשתנה Statement קיימת התכונה **nextlist** כמו שהוגדרה בכיתה.
5. ניתן ורצוי להעזר ב - M ו- N, המרקרים שהיכרנו בתרגולים, **ואך ורק בהם**.

בהצלחה!!