מבחן סוף סמסטר – מועד א'

מרצה אחראי: ד"ר איל זקס

מתרגלים: מתן פלד, אורן בניטה בן שמחון, שי גנדלמן, נדב רובינשטיין

הוראות:

א. בטופס המבחן 11 עמודים כולל עמוד זה, וכן 6 דפי נוסחאות. בדקו שכל העמודים ברשותכם.
ב. משך המבחן **שלוש שעות (180 דקות).**
ג. אסור כל חומר עזר חיצוני.
ד. במבחן 20 שאלות. כל השאלות הינן חובה. כל שאלה בת 5 נקודות.
ה. קראו את כל המבחן לפני שאתם מתחילים לענות על השאלות.
ו. אין צורך להגיש את טופס מבחן זה בתום הבחינה.
ז. את התשובות לשאלות יש לסמן בטופס התשובות הנפרד בלבד.

טור א

בהצלחה!

Backpatching

התיאור הבא מתייחס לשאלות 1-5

נרצה להוסיף לדקדוק מבנה בקרה של לולאה עם בדיקת יציאה באמצע, במקום בהתחלה או בסוף. לדוגמא:

```
midloop
   read(&buf);
midtest (iseof())
   print(buf);
midrepeat;
```

בדוגמא הנ"ל בתחילת הלולאה נקרא מקובץ, אח"כ נבדוק אם הגענו לסוף הקלט ואם כן נצא מהלולאה, אחרת נדפיס את מה שקראנו, ואז נקפוץ לתחילת הלולאה. כמו שניתן לראות, לולאה מסוג זה שימושית מאוד עבור מקרים בהם אנו נדרשים לקרוא מידע מהקלט עד שמגיעים ל-sentinel כלשהו, כי היא חוסכת שכפול של read והבדיקה לפני תחילת הלולאה.
נוסיף לדקדוק את הכלל הבא עבור מבנה בקרה זה:

S → <u>midloop</u> S <u>midtest</u> <u>LPAREN</u> B <u>RPAREN</u> S <u>midrepeat</u> <u>;</u>

שימו לב:

- בדקדוק אותו אנו מרחיבים יש למשתנים S ו-B כללי גזירה נוספים, ואין break או continue.
- למשתנה S יש תכונת nextlist ולמשתנה B תכונות truelist ו-falselist במסגרת הטלאה לאחור של חישוב מקוצר - short-circuit evaluation, backpatching.
- ניתן להניח שהקוד של משתנה S תמיד מסתיים ב-goto, כולל עבור משתנים S הנגזרים לפעולות read ו-print וכן midloop כבדוגמא לעיל.
- אין לשנות את הדקדוק, למעט הוספת מרקרים M, N שנלמדו בכיתה בלבד.
- אין להשתמש בכללים סמנטיים באמצע כלל גזירה.
- אין להשתמש במשתנים גלובליים בזמן קומפילציה.

נמספר מופעים של S ונציין מספר מקומות [P1] – [P8] בכלל הגזירה:

S → <u>midloop</u> [P1][P2] S_0 [P3] <u>midtest</u> <u>LPAREN</u> B [P4][P5] <u>RPAREN</u> [P6][P7] S_1 [P8] <u>midrepeat</u> <u>;</u>

בבואנו להוסיף כלל סמנטי בכדי לייצר קוד תוך שימוש בחישוב מקוצר והטלאה לאחור:

<u>שאלה מספר 1:</u>
לאילו כללים עלינו להוסיף תכונות סמנטיות, בנוסף לאלו שראינו בכיתה ומפורטים לעיל?

א. לכלל B

ב. לכללים S ו-B

ג. לכלל S

ד. אין צורך להוסיף תכונות סמנטיות

שאלה מספר 2:

היכן יש למקם מרקרי N (מרקר N ימוקם לפני מרקר M באותה נקודה)?

א. ניתן לייצר קוד ללא שימוש במרקרי N כלל

ב. לפני S_0, כלומר ב-[P1]

ג. אחרי B, כלומר ב-[P4]

ד. שני מרקרים: אחד לפני כל S, כלומר ב-[P1] וב-[P6]

שאלה מספר 3:

היכן יש למקם מרקרי M (מרקר M ימוקם לאחר מרקר N באותה נקודה)?

א. שלושה מרקרים: אחד לפני S_0, אחד אחרי S_0, ואחד לפני S_1, כלומר ב-[P2], [P3] ו-[P7]

ב. אחרי S_0, כלומר ב-[P3]

ג. לפני S_0, כלומר ב-[P2]

ד. אחרי B, כלומר ב-[P4]

שאלה מספר 4:

מה יכיל ה-nextlist של משתנה הכלל של מבנה הלולאה, כלומר של ה-S בצד שמאל של כלל הגזירה?

א. את ה-nextlist של S_0

ב. אף תשובה אינה נכונה

ג. איחוד של ה-nextlist-ים של S_0 ושל S_1

ד. את ה-nextlist של S_1

שאלה מספר 5:

היכן ימוקם הקוד לו נעשה emit בתוך הפעולה הסמנטית של הכלל?

א. חובה לעשות emit, הקוד ימוקם לאחר הקוד שמחשב את תוצאת הביטוי הבוליאני B

ב. ניתן לייצר קוד ללא כל emit בתוך הפעולה הסמנטית של הכלל

ג. חובה לעשות emit, הקוד ימוקם לפני הקוד של S_0, כלומר בהתחלה

ד. חובה לעשות emit, הקוד ימוקם לאחר הקוד של S_1

אופטימיזציה

קומפיילרים רבים מממשים אופטימיזציה שמגוללת לולאות במלואן (complete loop unrolling) ומחליפה לולאה בשכפול של הפעולות בתוכה כמספר האיטרציות של הלולאה. נניח כי בשפה אין הפעולות break או continue.

לדוגמא קטע הקוד הבא:

```
int i = 0;
do {
   sum = sum + 5;
   i = i + 1;
} while (i < 4);
```

יוחלף ע"י הקוד הבא:

```
int i = 0;

sum = sum + 5;
i = i + 1;
sum = sum + 5;
i = i + 1;
sum = sum + 5;
i = i + 1;
sum = sum + 5;
i = i + 1;
```

שאלה מספר 6:

מתי **לא** ניתן לבצע את האופטימיזציה complete loop unrolling?

א. טענות ד ו- ה נכונות

ב. טענות ג ו-ד נכונות.

ג. כאשר מספר האיטרציות של הלולאה איננו ניתן לחישוב בזמן קומפילציה.

ד. כאשר ישנן בלולאה פעולות המקצות זכרון באופן דינאמי.

ה. כאשר ישנן בלולאה פעולות printf.

שאלה מספר 7:

במהלך התרגול, שלושה סטודנטים התווכחו ביניהם:

- אופיר טוען שישנם מצבים בהם לקומפיילר מסוג JIT יש יתרון בהפעלת אופטימיזציה מסוג complete loop unrolling
- אורנה טוענת שעדיף להריץ ניתוח חיות משתנים ואופטימיזציית dead code elimination לאחר הפעלת אופטימיזציית complete loop unrolling, מאשר לפניה
- מוחמד טוען שאת האופטימיזציה ניתן לבצע רק בשלב ה backend של הקומפיילר

מי מהם צודק?

א. רק מוחמד צודק

ב. רק אופיר צודק

ג. רק אורנה צודקת

ד. אופיר ומוחמד צודקים

ה. אורנה ומוחמד צודקים

ו. אורנה ואופיר צודקים

נתון קטע קוד הביניים הבא, **המתייחס לשאלות 8-10**:

```
        X = 4
        N = X + 1
        M = Y + N
Label0: Y = Y * X
        X = X - 1
        if X > 0 goto Label0
        Z = Y + 5
```

על קוד הביניים הכולל שמכיל קטע זה הופעלה תחילה אנליזת חיות משתנים, ודיווחה כי **לאחר** קטע קוד זה חיים המשתנים M ו-Z, ורק הם.

כעת עליך להריץ את האופטימיזציות השונות שנלמדו בקורס כולל אופטימיזציית complete loop unrolling עד לקבלת קוד אופטימלי.

<u>שאלה מספר 8:</u>

איזו אופטימיזציה **אינה** מתבצעת?

א. Common sub-expression elimination
ב. Algebraic simplification
ג. Constant propagation
ד. Constant folding
ה. Useless code elimination

<u>שאלה מספר 9:</u>

כמה בלוקים בסיסיים היו לפני הרצת האופטימיזציות, וכמה לאחר הרצת האופטימיזציות?

א. לפני 3 ואחרי 1
ב. לפני 2 ואחרי 1
ג. לפני 4 ואחרי 2
ד. לפני 3 ואחרי 2

<u>שאלה מספר 10:</u>

על קוד הביניים הכולל שמכיל קטע זה הופעלה תחילה גם אנליזת הגדרות מגיעות (reaching definitions), ודיווחה כי לבלוק הבסיסי הראשון המתחיל בפקודה X=4 מגיעה רק הגדרה יחידה של Y והיא Y=3.

כעת הריצו שוב את האופטימיזציות השונות שנלמדו בקורס כולל complete loop unrolling עד לקבלת קוד אופטימלי. מהו מספר פקודות קוד הביניים המתקבל?

א. 1
ב. 7
ג. 0
ד. 3
ה. 2

Data Flow Analysis

שאלה מספר 11:

תזכורת: בלוק בסיסי A **שולט על** בלוק בסיסי B, אם כל מסלול מבלוק ההתחלה ל-B חייב לעבור דרך A.
לאחר חישוב הגדרות מגיעות reaching definitions וחישוב בלוקים שולטים dominating blocks
איזו מבין הטענות הבאות נכונה?

א. אם לתחילת בלוק בסיסי B מגיעה הגדרה יחידה d עבור משתנה נתון x, אזי בלוק בסיסי Bd המכיל את d חייב לשלוט על בלוק B?

ב. אם לתחילת בלוק בסיסי B מגיעות שתי הגדרות d1, d2 עבור אותו משתנה נתון x, כאשר d1 ו-d2 נמצאים בבלוקים שונים B1 ו-B2 בהתאמה, אזי יתכן שגם B1 שולט על B וגם B2 שולט על B?

ג. אף טענה איננה נכונה

ד. אם לתחילת בלוק בסיסי B מגיעות שתי הגדרות d1, d2 עבור אותו משתנה נתון x, אזי d1 ו-d2 יכולות להימצא באותו בלוק בסיסי Bd?

שאלה מספר 12:

בהמשך, נניח שזיהינו את כל הלולאות בתכנית; איזו מבין הטענות הבאות נכונה?

א. אם אף הגדרה שנמצאת בבלוק בסיסי B1 **לא** מגיעה לתחילת B1, אזי B1 בהכרח **לא** נמצא בלולאה?

ב. אף טענה איננה נכונה

ג. טענות א ו-ד נכונות

ד. אם הגדרה d1 שנמצאת בבלוק בסיסי B1 מגיעה לתחילת B1, אזי בלוק B1 בהכרח נמצא בלולאה?

שאלה מספר 13:

נאמר שבלוק בסיסי A "**קודם**" לבלוק בסיסי B אם ישנה קשת מ-A ל-B (A is predecessor block of B).
עבור כל הגדרה d1 שמגיעה לתחילת בלוק בסיסי כשלהו B, נרצה לדעת **מהיכן** מגיעה d1 ל-B; כלומר, **דרך** אילו בלוקים A ה"קודמים" ל-B מגיעה d1. איזו מבין הטענות הבאות נכונה?

א. אף טענה איננה נכונה

ב. טענות ג ו-ד נכונות

ג. הגדרה d1 שמגיעה לתחילת B, מגיעה אליו **דרך** בלוק "קודם" A אם ורק אם הגדרה d1 מגיעה לסוף בלוק A.

ד. ניתן לחשב ישירות את הבלוקים הקודמים דרכם מגיעה כל הגדרה, יחד עם פתרון בעיית ההגדרות המגיעות, באמצעות פתרון בעיית DFA אחת.

שאלה מספר 14:

נתונה פעולה d:i=i+1 אשר מגדירה ומשתמשת במשתנה i, ונמצאת בבלוק בסיסי B אשר נמצא בתוך לולאה. איזו מבין הטענות הבאות נכונה?

א. אם d היא ההגדרה היחידה של i בתוך הלולאה, אנליזת חיות משתנים בהכרח תדווח שהמשתנה i חי בסוף בלוק B.

ב. אם קיימת הגדרה נוספת של i בתוך הלולאה, אנליזת הגדרות מגיעות בהכרח תדווח שהגדרה d **איננה** מגיעה לתחילת בלוק B.

ג. אף טענה איננה נכונה.

ד. טענות א ו-ב נכונות

שאלה מספר 15:

איזו מבין הטענות הבאות נכונה?

א. טענות ג ו-ד נכונות

ב. אף טענה איננה נכונה.

ג. אם נדע שהגדרה d:i=i+1 של משתנה i, מגיעה לשימוש יחיד והוא d עצמו, ניתן לבטל את הפקודה d במסגרת הרחבה של useless code elimination.

ד. באנליזת DFA המחשבת עבור כל הגדרה d של משתנה i, אם ההגדרה d מגיעה לשימוש **יחיד** של משתנה i, ומדווחת שימוש יחיד זה, הסריקה היא **קדמית**.

בשאלות הבאות, אותיות גדולות A,B,... הם משתנים, S הוא המשתנה ההתחלתי, ואותיות קטנות a,b,... הם טרמינלים.

שאלה מספר 16:

נתונים הדקדוקים הבאים:

G1:	G2:
$S \to AB$	$S \to A$
$A \to aA \mid \epsilon$	$A \to AB \mid ab$
$B \to ba \mid bBa$	$B \to b$

איזו מבין הטענות הבאות לגבי **השפות** המוגדרות ע"י דקדוקים אלה, נכונה?

א. אף אחת מהשפות אינה ב-LL(1)

ב. רק השפה המוגדרת ע"י G1 נמצאת ב-LL(1)

ג. גם השפה המוגדרת ע"י G1 וגם השפה המוגדרת ע"י G2 נמצאות ב-LL(1)

ד. רק השפה המוגדרת ע"י G2 נמצאת ב-LL(1)

עבור שאלות 17-18 נתונים הדקדוקים הבאים:

G3:	G4:	G5:
$S \to aAb \mid aBc$	$S \to aAb \mid aBc$	$S \to AB \mid A$
$A \to a$	$A \to ac$	$A \to cAa \mid b$
$B \to a$	$B \to ab$	$B \to b$

<u>שאלה מספר 17:</u>

אילו **מהדקדוקים** הנ"ל נמצאים ב-$LR(0)$?

א. G4

ב. G3, G4

ג. G3, G4, G5

ד. G3

<u>שאלה מספר 18:</u>

אילו **מהדקדוקים** הנ"ל נמצאים ב-$LR(1)$?

א. G3, G4, G5

ב. G3, G4

ג. G3, G5

ד. G4, G5

שאלה מספר 19:

נתונות פעולות אריתמטיות בינאריות - @,#.

- בעל אסוציאטיביות שמאלית, ו-@ בעל אסוציאטיביות ימנית. כמוכן @ בעל קדימות גבוהה יותר משל #.

נתון הביטוי:

1 @ 2 @ 3 # 4 @ 5 # 6 # 7

אילו מהסוגריים הבאות מתאימות לסדר הפעולות בו יש לחשב את הביטוי הנתון?

א. ((1 @ 2) @ 3) # ((4 @ 5) # (6 # 7))

ב. ((1 @ 2) @ ((3 # 4) @ (5 # (6 # 7))

ג. (((1 @ (2 @ 3)) # ((5 @ 4) # (6 # 7

ד. 1 @ (2 @ ((3 # 4) @ ((5 # 6) # (7))

שאלה מספר 20:

נתון קוד bison הבא:

```
%{
#include <stdio.h>
int yylex();
void yyerror(char const *);
%}

%%

S:  A           {printf("1");}
    | B         {printf("2");}
;
A:  'a'         {printf("3");}
    | 'a' M A   {printf("4");}
;
B:  'b'         {printf("5");}
    | B M 'b'   {printf("6");}
;
M:  %empty      {printf("$");}
;
%%

int yylex() {return getchar();}
void yyerror(char const * s) {printf("ERR\n"); exit(1);}
int main() {return yyparse();}
```

כאשר הסימון %empty מסמן גזירה של המילה הריקה, והפונקציה yylex() מחזירה את התו הבא בקלט.

מה יהיה פלט המנתח המתקבל בריצה על הקלטים "aaaaa" ו-"bbbbb" בהתאמה?

א. 5$6$6$6$62 ,3$4$4$4$41

ב. 5$6$6$6$62 ,$$$$344441

ג. $$$$566662 ,3$4$4$4$41

ד. $$$$566662 ,$$$$344441