30.1.2017

מבחן סוף סמסטר – מועד א׳

מרצה אחראי: ד״ר אוהד שחם

מתרגלים: אבנר אליזרוב מיכל בדיאן הילה פלג עידן שורץ

הוראות:

א. בטופס המבחן X 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 (12 נק'):סיווג מאורעות

לפניכם מס' שורות המכילות נתוני סטודנטים ממוסד אקדמי מוכר. השורות יכילו את שם הפקולטה ולאחר מכן את שמות התלמידים מאותה פקולטה.
העמודה הראשונה מציינת את שם הסטודנט, העמודה השנייה את תאריך הבחינה, העמודה השלישית את המקצוע והרביעית את הציון בבחינה. כל עמודה מופרדת בסימן ',' ולאחריו רווח.

```
CS Students
David Cohen, 29.01.17, Compilation, 80
David Cohen, 29.02.16, Theory of computation, 95
EE Students
Shani Sitbon, 29.01.17, Compilation, 80
Shani Sitbon, 11.03.15, Human-Computer Interfaces, 60
Shani Sitbon, 29.02.16, Compilation, 85
```

ברצוננו להשתמש בידע שרכשנו בקורס קומפילציה כדיי לבצע מספר פעולות עיבוד על הקלט. להזכירכם למדנו על מספר שלבים בתהליך הקומפילציה : השלב הסמנטי, השלב התחבירי השלב הסמנטי ושלב האופטימיזציה.
נמקו בקצרה מהו השלב הכי מוקדם אשר יתאים לכל פעולה ובעזרת איזה כלי שלמדנו(flex, bison) תבצעו זאת.

1. ברצוננו להדפיס ממוצע של כל סטודנט (עם הציון האחרון במקצוע), הפלט הרצוי

```
CS Students
David Cohen, 87.5
EE Students
Shani Sitbon, 80
```

2. ברצוננו לבדוק שהתאריך בעמודה השנייה תקין.
3. ברצוננו לבדוק שהקובץ מכיל לפחות סטודנט אחד לכל פקולטה המצוינת בקובץ. בקובץ הנ"ל' מצוינות הפקולטות CS וEE.
4. ברצוננו לבדוק שהשמות מסודרים בסדר לקסיקוגרפי עולה.
5. ברצוננו לבדוק שכל פקולטה מכילה מספר זוגי של סטודנטים.
6. ברצוננו להדפיס קובץ נקי המכיל רק את הציון האחרון של כל סטודנט במקצוע. הפלט הרצוי

```
CS Students
David Cohen, 29.01.17, Compilation, 80
David Cohen, 29.02.16, Theory of computation, 95
EE Students
Shani Sitbon, 29.01.17, Compilation, 80
Shani Sitbon, 11.03.15, Human-Computer Interfaces, 60
```

שאלה 2 (30 נק'): רשומות הפעלה ופעולות וקטוריות

בשאלה זו נגדיר מנגנון חדש המאפשר הרצת פרוצדורות במקביל. המקבילות תתבצע על ידי חוט יחיד אבל סדר פקודות הפרוצדורות שקול להרצת הפרוצדורות במקביל ותוחלט על ידי הקומפיילר.

נתונה הדוגמא הבאה:

```
void sum (int *x, int y, int z) {
    *x = y + z;
}

void sumArray(int *x, int *y, int *z) {

     for (int i=0; i < N; i+=4) {
             par4( sum(x+i, y[i]], z[i]),
                   sum(x+i+1, y[i+1]], z[i+1]),
                   sum(x+i+2, y[i+2]], z[i+2]),
                   sum(x+i+3, y[i+3]], z[i+3])
             );
     }
}
```

1) הפקודה par4 מריצה 4 פרוצדורות sum במקביל. כיצד היית תומכת בפקודה זו? התמיכה חייבת להתבצע על ידי קריאה אחת לפרוצדורה שתבצע את כל הרצות sum. נמקי מה היית משנה במנגנון רשומות ההפעלה.
2) מה היתרון בשימוש ב par4 על פני הרצת הפרוצדורות אחת אחרי השניה.
3) נניח שהארכיטקטורה כוללת רגיסטרים וקטוריים של 128 ביט. כיצד היית משנה את מימוש sum שתתמוך בפעולות וקטוריות? מה היית משנה בתשובה שנתת בסעיף 1 בשביל להשתמש בפונקציה הוקטורית ביעילות? במידה ולא היית משנה כלום, נמקי מדוע.
4) הסבירי איזה פעולה נדרשת עבור ויקטור התוכנית הבאה.

```
void sum (int *x, int y, int z) {
    if (y > 0 && z > 0) {
        *x = y + z;
    }
}
```

שאלה 3 (21 נקודות) אופטימיזציה ו DFA

א) (6 נקודות) בהינתן קטע הקוד הבא, הציעו 3 אופטימיזציות אשר אפשריות בשלב הקומפילציה.
ניתן להניח שקוד הביניים ייכתב בפקודות שאתם מכירים בלבד אך אין צורך לכתוב אותו.
כתבו את מספר השורה, שם האופטימיזציה והסבירו את השינוי.

```
1.  void print (int N, int M)
2.  {
3.      if(N<3 || M<3) return;
4.      int i;
5.      int x,y;
6.      int s=0;
7.      char * first_name = “Michal”;
8.      char * last_name = “Hila”;
9.      for(i=2; i<M; i+=2)
10.     {
11.         s=5*i;
12.         printf(“%d”,s);
13.     }
14.     y = 6*2 + M;
15.     if (false)
16.        printf(%s,first_name);
17.     else
18.        printf(%s,last_name);
19. }
```

ב) (15 נקודות) נתונה פקודה "סימון" חדשה מהצורה tag(p,k).
עבור פקודה זו, p הוא שם של משתנה ו k הוא הגודל שלו אשר קבוע וידוע בזמן קומפילציה.
כמו כן נתון ערך M קבוע וידוע מראש.
נאמר שמשתנה x עדיין מסומן בשורה מסוימת בתוכנית, אם קיים מסלול מפקודת הסימון לשורה הזו כך ש:

a. כל פקודות הסימון עבור המשתנה שהיו לאורך המסלול $\mathrm{tag(x, k\_1)} \ldots, \mathrm{tag(x, k\_n)}$ מקיימות $\mathrm{k\_1} + \cdots + \mathrm{k\_n} \leq \mathrm{M}$.
b. השמה מחדש למשתנה מוחקת את הסימון.

למשל בקטע הקוד הבא:

נתון $M = 32$.

```
1.  b:=0
2.  p=3
3.  tag(p,8)
4.  tag(r,4)
5.  if(b>0) {
6.          tag(p,30)
7.          tag(p,30)
8.          r=2
9.          print(a)
10. }
```

בשורה 5, המשתנים p ו r מסומנים.
בשורה 6 המשתנה r מסומן.
בשורה 7 המשתנה r ו p מסומנים.
בשורה 8 המשתנה p מסומן.

הראו אנליזת DFA המחשבת לכל בלוק בסיסי את קבוצת המשתנים המסומנים.

הערות:

- ניתן להגדיר שורה בודדת כבלוק בסיסי.
- יורדו נקודות על אי <u>הגדרה מלאה</u> של DFA כפי שנלמד בתרגול. יש לציין מהם פריטי המידע, האם האנליזה היא must/may, האם היא קדמית או אחורית, ושאר פרמטרי DFA כפי שנלמדו בכיתה.

שאלה 4 (22 נק׳): ניתוח תחבירי או סמנטי

נתון הדקדוק (החלקי) הבא עבור שפת תכנות. ניתן להניח כי החלק החסר הוא statements בלבד: מבני בקרה נוספים, השמות למשנים, וכד'.

```
E -> E BINOP E
E -> E RELOP E
E -> E LOGOP E
E -> id | num | true | false
BINOP -> + | -
RELOP -> == | != | < | > |<= | >=
LOGOP -> && | ||
TYPE -> int | bool
S -> TYPE id;
S -> if (E) then S;
S -> print string;
```

האסימונים עבור BINOP ו-RELOP מוגדרים כולם כבעלי אסוציאטיביות שמאלית.

דקדוק זה מאפשר לגזור את הפקודה הבאה:

```
if (4 < x < 6) then print "five";
if (num < id < num) then print string;
```

הביטוי בתנאי מבצע את > על מספר וביטוי בוליאני ועל כן נכשל בבדיקת הטיפוסים של השפה.

א) (8 נקודות) מפתחי השפה מעוניינים לחסוך לעצמם את הצורך לבצע את בדיקת הטיפוסים ולכן רוצים לחסום אפשרות זו בדקדוק עצמו.

האם ניתן לבצע זאת ברמת הדקדוק? אם כן, הראו את הדקדוק המתוקן. אם לא, נמקו מדוע.

ב) (8 נקודות) האם ניתן לבצע זאת על ידי תכונות נורשות? אם כן, הראו את התכונות והסבירו את אופן חישובן ופעולתן. אם לא, נמקו מדוע.

ג) (6 נקודות) (סעיף זה אינו תלוי בסעיפים הקודמים) האם ייתכנו שני המצבים הבאים של מחסנית הריצה כמצבים עוקבים בדקדוק LR(1)? אם כן, הראו דקדוק LR(1) והציגו ריצה זאת. אם לא, נמקו מדוע.

n) (0,)(1,A)(2,b)

n+1) (0,)(1,A)(2,b)(3,B)

שאלה 5 (15 נקודות): Backpatching

נתון מבנה הבקרה הבא:

$SL \rightarrow SL\ ;\ S\ \mid\ S$

$S \rightarrow break;$

$S \text{ -> } for\ (\ S1\ ;B\ ;\ S2\ )\ \{\ SL\ \}$

המבנה זהה ללולאת for המוכרת מרוב שפות התכנות –

- מבצעים את S_1
- כל עוד התנאי B מתקיים –
  - מבצעים את רשימת ה statement.
  - מבצעים את S_2
- אם יש פעולת break יוצאים מהלולאה לגמרי ללא צורך בעוד עדכונים.

לדוגמא,
עבור הקוד הבא –

```
for( int i=0;  i< 5 ;  i++){
        print(i);
}
```

המשתנה i יאותחל ב0 ואז הלולאה תתבצע 5 פעמים שבכל פעם יודפס של i ואז יקודם ערכו ב1 .
באיטרציה הראשונה יודפס 0, אחר כך 1 וכו׳.

א. (5 נקודות) הציעו פריסת קוד המתאימה לשיטת backpatching עבור מבנה הבקרה הנ״ל. על הקוד הנוצר להיות יעיל ככל האפשר. הסבירו מהן התכונות שאתם משתמשים לכל משתנה.

ב. (10 נקודות) כתבו סכימת תרגום בשיטת backpatching המייצרת את פריסת הקוד שהצעתם בסעיף הקודם. על הסכימה להיות יעילה ככל האפשר, הן מבחינת זמן הריצה שלה והן מבחינת המקום בזיכרון שנדרש עבור התכונות הסמנטיות.

שימו לב:

- אין לשנות את הדקדוק
- אין להשתמש בכללים סמנטיים באמצע כלל גזירה
- ניתן להשתמש במרקרים N,M שנלמדו בכיתה בלבד.
- אין להשתמש במשתנים גלובליים בזמן קומפילציה.
- למשתנים B,S ישנן התכונות שהוגדרו בכיתה בלבד.
- למשתנים B,S יש כללי גזירה פרט לאלו המוצגים בשאלה.

בהצלחה!!

נוסחאות ואלגוריתמים

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

Top Down

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

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

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

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 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:

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

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:

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

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
```

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
2. משפטי השמה עם פעולה אונרית
3. משפטי העתקה
4. קפיצה בלתי מותנה
5. קפיצה מותנה
6. פרמטרים וקריאה לפרוצדורות
7. indexed assignments
8. השמה של כתובות ומצביעים

```
x := y op z
x := op y
x := y
goto L
if x relop y goto L
param x
call p, n
return y
x := y [ i ]
x [ i ] := y
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:

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

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