שאלון למטלת מנחה (ממ״ין)11

שם הקורס: **קומפילציה**

שם המטלה: **ממ״ן 11**	**מס׳ הקורס: 20364**
משקל המטלה: **3 נקודות**	**מס׳ המטלה: 11**
מספר השאלות: **3**	**מחזור: א 2014**

מועד משלוח המטלה: 27.10.2013

אנא שים לב:
מלא בדייקנות את הטופס המלווה לממ״ן
בהתאם לדוגמה המצויה בפתח חוברת המטלות
העתק את מספר הקורס ומספר המטלה הרשומים לעיל

שאלה 1 (35%)

נגדיר שפה הנקראת שפת Open University XML –OUXML.

האסימונים בשפה הם:

שם האסימון	תאור מילולי
start_OUXML	הפקודה <ouxml>
end_ OUXML	הפקודה <ouxml/>
date	המשך (תכונה) של הפקודה ouxml <date=”11 may 2014”
new_line	הפקודה <nl>
inner_text	סדרה לא ריקה של תווים. הסדרה לא כוללת סימנים ״>״ או״<״
start_course	הפקודה <course>
course_number	המשך (תכונה) של הפקודה course תכונה מספר הקורס number=20364
semester	המשך (תכונה) של הפקודה course תכונה סמסטר בו סטודנט רוצה ללמוד את הקורס ״semester="2014a
end_course	הפקודה <course/>
start_comment	הפקודה --!>
end_comment	הפקודה <--
OU_student	הפקודה <oustudent
student_name	המשך של הפקודה student שם "name=rina zviel"
number_of_points	המשך (תכונה) של הפקודה student מספר נקודות זכות <number_of_points=85
start_account	הפקודה <acc
bank_data	המשך (תכונה) של הפקודה account שם הבנק ומספר vחשבון <bank="fibi" number=12345
end_account	הפקודה <acc/>

הערה: מנתח לקסיקלי מדלג על כל האסימונים הנמצאים בין start_comment (<!--) ל- end_comment (<!--)

א. הציגו כל האסימונים של שפת OUXML על ידי ביטוים רגולריים.

ב. נניח שנתון לנו מנתח לקסיקלי עבור השפה המתוארת, המבוסס על האוטומט שרשמתם בסעיף ב'. תארו כיצד יפעל המנתח כאשר תינתן לו כקלט המחרוזת הבאה:

```
<ouxml date="11 may 2014">
<oustudent name="rina zviel" number_of_poits=85>
        </oustudent>
<!-- This line is a comment and should be ignored -->
<course number=20364 semester="2014a">Compilation</course>
<course number=20407 semester="2014b">Introduction to Algorithms and Data
Structures
        </course>
<acc bank="leumi" number=12345>3628 SH</acc>
</ouxml>
```

הערה: יש לשים לב שהשפה איננה case_sensitive (גודל האותיות איננו משנה).

שאלה 2 (תוכנית מחשב - 50%)

כתבו מנתח לקסיקלי המקבל כקלט תוכנית בשפת OUXML.
המנתח הלקסיקלי הוא תוכנית, כתובה בשפת C. הוא מקבל קובץ קלט ובו תוכנית בשפת OUML, ומזהה את האסימונים המופיעים בתוכנית.

מנתח לקסיקלי צריך להתעלם מרווחים, שורות חדשות וtabs.
כאשר נתקלים בשגיאה יש להוציא הודעת שגיאה (עם מקום מדויק של שגיאה) ולהמשיך לנתח את האסימונים הבאים.

הממשק
המנתח יקרא lexouxml.exe
המנתח הלקסיקלי יופעל משורת הפקודה של DOS.
קלט – התוכנית מקבלת כפרמטר יחיד שם של קובץ קלט (קובץ טקסט המכיל ,תוכנית בשפת מיני HTML). הסיומת של שם קובץ הקלט חייבת להיות OUML או ouml.
שורת הפקודה היא: lexouxml <file_name>.OUXML
פלט – התוכנית יוצרת קובץ "אסימונים", עם סיומת tok או TOK: בקובץ זה מדפיס המנתח את טבלת האסימונים מהם מורכבת התוכנית, כאשר כל אסימון מופיע בשורה נפרדת.

לדוגמה, שורה בטבלה יכולה להראות כך:

TOKEN	LEXENE	ATTRIBUTES
start_account	<acc	
bank_data	bank="leumi" number =12345>	Bank name leumi Account number 12345

שימו לב:
חשוב שממשק התוכנית שתכתבו יהיה בדיוק כפי שמוגדר במטלה. ייתכן שבדיקת הריצה של תוכניתכם תיעשה בצורה אוטומטית, ובמקרה כזה תוכנית בעלת שם שונה או ממשק שונה תיכשל. וודאו גם שניתן להריץ את תוכניתכם כאשר נמצאים במדריך אחר, ולא המדריך שבו

נמצאת התוכנית עצמה.

<u>הערה</u> :
מותר להשתמש ב-FLEX

שאלה 3 (15%)

נתונים הדקדוקים הבאים :

1. **S→ bbS | bSaa | ε**

2. S→ Sb | aA
A→ aA | a

3. S→ bbZa | bZaa | aba
Z→ bba | bb| a

מהי שפת הדקדוק?
האם מדובר בשפה סופית?
(יש לתת תיאור פורמלי ככל שניתן. אם יש דרך לרשום ביטוי רגולרי אז יש לכתוב את הביטוי.)

שאלון למטלת מנחה (ממ״ין)12

שם הקורס: קומפילציה	**מס׳ הקורס: 20364**
שם המטלה: ממ״ן 12	**מס׳ המטלה: 12**
משקל המטלה: 3 נקודות	**מחזור: א 2014**

מספר השאלות: 5

מועד משלוח המטלה: 7.11.2013

אנא שים לב:
מלא בדייקנות את הטופס המלווה לממ״ן
בהתאם לדוגמה המצויה בפתח חוברת המטלות
העתק את מספר הקורס ומספר המטלה הרשומים לעיל

שאלה 1 (תוכנית מחשב - 40%)

כתבו מנתח לקסיקלי לשפת CPL.

בנו אוטומט סופי משותף לזיהוי אסימוני השפה וממש אותו בקוד.
ראו הגדרות והסברים של אסימונים חוקיים בממ״ן 16.

<u>הבהרות</u>

יש לממש את המנתח הלקסיקלי כפונקציה (שתיקרא בשלב מאוחר יותר על-ידי המנתח התחבירי). הפונקציה תחזיר בכל קריאה את ״האסימון הבא״ בקלט ואת ערכי תכונותיו (אם ישנן כאלה), כפי שנלמד בקורס.
כדי לאפשר את בדיקת המנתח הלקסיקלי בשלב זה, הוסף תכנית ראשית, שתייצג עבור המנתח הלקסיקלי את ״שאר הקומפיילר״.

התכנית הראשית תקרא למנתח הלקסיקלי שוב ושוב עד סוף קובץ הקלט ותייצר קובצי פלט כמפורט להלן, בסעיף ״ממשק״.

<u>טבלת סמלים</u>

אם אתה מעוניין בכך, תוכל להוסיף בשלב זה גם ניהול של טבלת סמלים שבה יישמרו המזהים. את הטבלה ניתן לממש על-ידי מבנה-נתונים מוכר (טבלת hash, עצים למיניהם), או על-ידי וריאציה מתוצרת עצמית. בכל מקרה, <u>אין</u> לממש את טבלת הסמלים בצורה הפשוטה ביותר של מערך או רשימה עם חיפוש לינארי. הבא בחשבון שמבנה הרשומה בטבלת הסמלים עשוי להשתנות בשלב מאוחר יותר של כתיבת הקומפיילר כדי לאפשר שמירה של מידע נוסף על כל מזהה.
שים לב: סעיף 7.6 בספר דן בטבלאות סמלים, ותוכל להיעזר בו כעת.

כמה החלטות תכנון נוספות הנדרשות כאן הן:
- אלו תכונות (אם בכלל) יש לאסימונים השונים.
- האם הכנסת המזהים לטבלת הסמלים נעשית במנתח הלקסיקלי או בחלק אחר.
- האם מלים שמורות מוכנסות לטבלת הסמלים או לא.

בחלק זה של הפרויקט יש גם לנהל את הקלט מבחינת ספירת השורות, איתור סוף קובץ הקלט וכדומה.

<u>הממשק</u>

המנתח יקרא cla (cpl lexical analyzer) (עם סיומת c. לקובץ המקור ו-exe. לקובץ הריצה).
המנתח התחבירי יופעל משורת הפקודה של DOS.
קלט – התוכנית מקבלת כפרמטר יחיד שם של קובץ קלט (קובץ טקסט המכיל משפט בשפת המחשבון). הסיומת של שם קובץ הקלט חייבת להיות cpl או CPL.
שורת הפקודה היא: cla <file_name>.cpl
פלט – התוכנית יוצרת שני קובצי טקסט עם שם זהה לשם קובץ הקלט ועם סיומות כדלקמן:
<u>קובץ "אסימונים", עם סיומת tok או TOK</u>: בקובץ זה מדפיס המנתח את האסימונים (פורמט נוח לבדיקה (!)) מהם מורכבת התוכנית, כאשר כל אסימון מופיע בשורה נפרדת.
<u>קובץ "רישום" (listing), עם סיומת lst או LST</u>: אל הקובץ הזה מעתיק המנתח את קובץ הקלט ובקובץ זה מופיעות גם הערות השגיאה (אם ישנן) ומקום מדויק של השגיאה.
במקרה של שגיאה בפרמטר הקלט, בפתיחת קבצים וכדומה, יש לסיים את הביצוע בצירוף הודעת שגיאה מתאימה למסך (stderr).
יש לכתוב שורת "חותמת" עם שם הסטודנט ל-stderr, וכן בתחילת הקבצים lst ו-tok.

שימו לב:
חשוב שממשק התוכנית שתכתבו יהיה בדיוק כפי שמוגדר במטלה. ייתכן שבדיקת הריצה של תוכניתכם תיעשה בצורה אוטומטית, ובמקרה כזה תוכנית בעלת שם שונה או ממשק שונה תיכשל. וודאו גם שניתן להריץ את תוכניתכם כאשר נמצאים במדריך אחר, ולא המדריך שבו נמצאת התוכנית עצמה.

<u>הגשה</u>

יש להגיש דיסקט/דיסק ועליו:

- קובצי מקור (בפסקל או ב-C) של התכנית.
- קובץ לריצה לאחר קומפילציה של הקבצים הנ"ל (קובץ exe.).
  (הנח שהמטלה תיבדק על מערכת MS-DOS).
- קובץ ושמו readme ובו סקירה של כל הקבצים על-גבי הדיסקט/דיסק, והערות מיוחדות.

שאלה 2 (20%)

נתון דקדוק הבא:

S→S^S | S/S | S-S | id

בדקדוק זה אין חשיבות לסדר פעולות.
א. רשום את הדקדוק מחדש לפי סדר עדיפויות הבא:
^ פעולה בעלת עדיפות הגבוהה ביותר
/ פעולה בעלת עדיפות נמוכה יותר
- פעולה בעלת עדיפות נמוכה ביותר.
יש לרשום דקדוק חד-משמעי.
ב. בנה עץ פיסוק עבור המחרוזת id^id/id-id^id

שאלה 3 (20%)

נגדיר דקדוק G, שבו הטרמינלים הם {**0,1,2**} והמשתנים הם { **S,A,B**} (S הסימן התחילי). כללי הדקדוק הם:

S → 0BA | S1
A → 1B1| ε
B → 1A2

ב. חשבו את first ו-follow לכל אחד ממשתני הדקדוק G.
ג. בנו טבלת פיסוק תחזית (טבלת LL(1)) לדקדוק G.
האם הדקדוק הינו דקדוק מסוג LL(1)?

שאלה 4 (10%)

נגדיר דקדוק G, שבו הטרמינלים הם {**a,b,c,d** } והמשתנים הם { **S, A, B, C**} (S הסימן התחילי). כללי הדקדוק הם:

S → aS | SbBb | BSa
B → abA | dAad | Aac
A → Bba | caCd
C → cBa | cB | ε

הפעילו על הדקדוק את הפעולות הבאות, במידת הצורך:

- סילוק רקורסיה שמאלית (ישירה ולא ישירה)

(בתשובתכם, אין הכרח לסלק מהדקדוק את הכלל $C \rightarrow \varepsilon$: הוא אינו מפריע לביצוע האלגוריתם לסילוק רקורסיה שמאלית בדוגמא זו למרות שהאלגוריתם הזה פועל על דקדוקים ללא כללי אפסילון).

- צמצום גורמים שמאליים (left factoring).

שאלה 5 (10%)

נתון הדקדוק הבא:

S → Aab | cA | b
A → bbA | aSb | aS

השלימו את טבלת הפעולות שמבצע מפסק תחזית טבלאי על מחרוזת הקלט cbbaababb.

האם המפסק מצליח לגזור את המילה?

Parser stack	Remaining input	Parser action
S$	cbbaababb $	Predict S→cA, pop S, push Ac

שאלון למטלת מנחה (ממ״ין)13

שם הקורס: **קומפילציה**

מס׳ הקורס:	**20364**
מס׳ המטלה:	**13**
מחזור:	**א 2014**

שם המטלה: **ממ״ן 13**

משקל המטלה: **3 נקודות**

מספר השאלות: **3**

מועד משלוח המטלה: **1.12.2013**

אנא שים לב:
מלא בדייקנות את הטופס המלווה לממ״ן
בהתאם לדוגמה המצויה בפתח חוברת המטלות
העתק את מספר הקורס ומספר המטלה הרשומים לעיל

שאלה 1 (40 %)

נגדיר דקדוק G, שבו הטרמינלים הם {**a,b,c,d,e**} והמשתנים הם { S, M, P, Q}(S הסימן התחילי). כללי הדקדוק הם:

S → M | P
M → e | bM
P → adM | dMQ | ad
Q → QPc | Qbc | ed | ε

א. הפעילו על הדקדוק את הפעולות הבאות, במידת הצורך:

1. סילוק רקורסיה שמאלית
2. צמצום גורמים שמאליים (left factoring) – יש למספר את הכללים מחדש.

ב. בנו את אוטומט פריטי LR(0) לדקדוק 'G. (כלומר, רשמו את האוסף של קבוצות פריטי LR(0), וציירו את קשתות המעברים בין הקבוצות). אל תשכחו לעבור קודם לדקדוק מורחב.

ג. בנו טבלאות פיסוק SLR לדקדוק 'G (action ו-goto). האם דקדוק הינו SLR.

ד. בנו את אוטומט פריטי LR(1) לדקדוק 'G (כמו בסעיף ב', הפעם עם פריטי LR(1)).

ה. האם 'G הוא דקדוק מסוג LALR? הוכיחו את תשובתכם.

שאלה 2 (40 %)
טבלת הפיסוק והשימוש ב-bison

בפרויקט גמר עליכם לממש מפסק בשיטת הניתוח העולה, כלומר באחת משיטות LR שנלמדות בספר: SLR, LALR, או LR קנוני (LR(1)).

בפרויקט גמר תוכלו לבחור באחת משתי אפשרויות לבניית המפסק:

1. שימוש בתוכנת bison (ראו פירוט בסעיף הבא).
2. בניית מנתח תחבירי לפי אחת מהשיטות המפורטות לעיל, ללא שימוש בכלים אוטומטיים.

כיון שמדובר בדקדוק גדול למדי, קשה לבנות את הטבלה באופן ידני. ניתן להיעזר ב-bison לצורך בניית טבלת LALR.

בשאלה זו אתם צריכים לשתמש ב-bison לבניית המפסק לשפת הפרויקט CPL.

- כדי להפעיל את bison בצורה המתוארת עליכם ליצור עבורו קובץ קלט פשוט, המכיל רק את תיאור הדקדוק. לדוגמה, לדקדוק של שפת CPL יש לבנות קובץ בעל הצורה הכללית הבאה:

```
%{
%}
%token ID
%token PROCEDURE
…
%token INT
%token REAL
…    (Define all terminals in the grammar.
       Symbols such as +, –, (, ) don't need to be defined).
%%
PROGRAM   :    PROCEDURE ID {  DECLARATIONS  STMTLIST }
PERIOD_TOKEN
          ;
PERIOD_TOKEN    :   .
          ;
…

DECLARATIONS    :     VARIABLE  DECLIST  ';'
          ;

DECLARLIST   :     DECLIST , ID : TYPE
                 |    ID : TYPE
          ;
TYPE            :   INT
                 |   REAL
          ;
   … (Continue with all grammar productions).
```

- הקובץ יקרא cla.y (cpl lexical analyzer)
- אתם צריכים להפעיל את bison בעזרת האופציה "v-", ייווצר קובץ נוסף בעל סיומת out, המכיל מידע על מצבי המפסק, כלומר על קבוצות פריטי LR(1) שלו.
- שורת הפקודה היא: bison –v cla.y
- בקובץ cla.out רשומים
  - מצבים כקבוצת פריטי LR(0)
  - לכל מצב נתונות כניסות ה- action וה-goto

o הסימן . מחליף כאן את הנקודה שנהוגה בספר

<u>הערה</u> :
שיטת הפיסוק של bison היא שיטת LALR, ולכן שימוש בהנחיות הללו יוביל אתכם לבניית טבלת LALR לדקדוק של שפת המקור.
תוכנת bison נמצאת במדריך bison\ בתקליטון שנשלח אליכם.

<u>הגשה</u>
יש להגיש דיסקט/דיסק ועליו :

- קובצי מקור cla.y של התכנית.
- קובץ cla.out.
- הדפסה של קובץ cla.out.

שאלה 3 (20 %)

נתון הדקדוק הבא :

S→ 0A | 0B
A→ 1A | 0
B→ 2B | 1C
C→1

א. בנה טבלת first ו-follow עבור הדקדוק.
ב. האם הדקדוק הוא מסוג LL(1)? השלימו בנית טבלה פיסוק מסוג LL(1).

	0	1	$
S			
A		**A→ 1A**	
B			

ג. האם הדקדוק הוא מסוג LL(2)? בנה טבלה מסוג LL(2). (יש לענות על השאלה בלי לבצע שינוים בדקדוק).

שאלון למטלת מנחה (ממ״ין)14

שם הקורס: קומפילציה	**מס׳ הקורס: 20364**
שם המטלה: ממ״ן 14	**מס׳ המטלה: 14**
משקל המטלה: 3 נקודות	**מחזור: א 2014**

מספר השאלות: 4

מועד משלוח המטלה: 22.12.2013

אנא שים לב:
מלא בדייקנות את הטופס המלווה לממ״ן
בהתאם לדוגמה המצויה בפתח חוברת המטלות
העתק את מספר הקורס ומספר המטלה הרשומים לעיל

שאלה 1 (20%)

נתון הדקדוק הבא:

S → aS
S → Sbb
S → aSb
S → aaa
S → abb

א. הוסף לדקדוק זה פעולות סמנטיות המדפיסה את כל ה-"b" במילה, תוך כדי קריאתה.

ב. הוסף לדקדוק זה פעולות סמנטיות שמאפשרות הדפסה של כל ה-"a"-ים לאחר שמודפס מספר המסמן מספר ה-"b"-ים במילה, תוך קריאתה.
דוגמה: עבור הקלט הבא aabb הפלט יהיה 2aa.

שאלה 2 (25%)

נתאר שפה של ״ביטויי רשימות״ – כל ביטוי בשפה מתאר רשימה של מספרים טבעיים - שפת **OULP (Open University List Programming language)**

מרכיבי השפה הם:

- רשימות פשוטות של מספרים - לדוגמה: [12 , 4 , 100 , 57]
המספרים ברשימה הם שלמים אי-שליליים. מספר יכול לחזור יותר מפעם אחת ברשימה.
- פעולת first(L) - מקבלת רשימה L ומחזירה את המספר השני ברשימה. לדוגמה:
second([12 , 4 , 100 , 57]) = 12
שימו לב שפעולה זו מחזירה מספר ולא רשימה.
- פעולת last(L) - מקבלת רשימה L ומחזירה את המספר האחרון ברשימה. לדוגמה:
last([12 , 4 , 100 , 57]) = 57
שימו לב שפעולה זו מחזירה מספר ולא רשימה.

- פעולת add(n, L) - מקבלת מספר n ורשימה L, ומחזירה רשימה שבה n מצורף בתור איבר ראשון ל-L .

לדוגמה: add(5, [12 , 4 , 100 , 57]) = [5 , 12 , 4 , 100 , 57]

- פעולת rconc(n, L) - מקבלת מספר n ורשימה L, ומחזירה רשימה שבה n מצורף בתור איבר אחרון ל-L . לדוגמה: rconc(5, [12 , 4 , 100 , 57]) = [12 , 4 , 100 , 57 , 5]

דוגמה לביטוי בשפה:

rconc(second([2 , 3]) , add(second([4, last([5 , 6]), 7]) , [8 , 9]))

הביטוי מייצג את הרשימה [6, 8 , 9 , 3] .

נתון הדקדוק הבא הגוזר "ביטויי רשימות":

1. $S \rightarrow L$
2. $L \rightarrow$ **[** NUMLIST **]**
3. $L \rightarrow$ **add (** ITEM **,** L **)**
4. $L \rightarrow$ **rconc (** ITEM **,** L **)**
5. NUMLIST $\rightarrow$ NUMLIST **,** ITEM
6. NUMLIST $\rightarrow$ ITEM
7. ITEM $\rightarrow$ **num**
8. ITEM $\rightarrow$ **first (** L **)**
9. ITEM $\rightarrow$ **last (** L **)**

כתבו סכמת תרגום המסתמכת על הדקדוק הנתון, אשר

1. מחליטה אם הרשימה שמיוצגת על-ידי הביטוי היא ממוינת בסדר עולה
2. מדפיסה את רשימת התוצאה (במקרה שרשימה ממוינת)
3. מחשבת ממוצע של איברי הרשימה ומדפיסה אותה על המסך

לדוגמה:

אם הביטוי מייצג את הרשימה [1, 2, 3], סכמת התרגום תדפיס הודעה "הרשימה ממוינת בסדר עולה -[1, 2, 3] . ממוצע - 2.0".

אם הביטוי מייצג את הרשימה [1, 4, 3], סכמת התרגום תדפיס הודעה "הרשימה אינה ממוינת בסדר עולה. ממוצע -2.67 "

<u>הערות:</u>

מותר להגדיר תכונות למשתני הדקדוק לפי הצורך. אין להגדיר משתנים גלובאליים.
הפעולות שמוצמדות לכל כלל צריכות להתייחס רק לתכונות של המשתנים המופיעים באותו כלל, ולא ל"משתנים גלובליים".
האסימון num מייצג מספרים. הניחו שהתכונה num.val מכילה את ערך המספר num, והיא מסופקת על-ידי המנתח הלקסיקלי.

- אין צורך להתייחס למקרה של רשימה ריקה – ניתן להניח שהקלט לא יהיה ביטוי המייצג רשימה ריקה, וגם לא יהיה בו תת-ביטוי המייצג רשימה ריקה.

שאלה 3 (15%)

הראו כיצד ניתן ליישם תרגום של ביטוים הבאים בשפה דמוית C לשפת QUAD.

a. **for(j=0;j<5;j++);**

b. **j=0; do {++j} while(j<5);**

c. **j=0; k=0; while(j<5) {++j; k+=j;}**

בכל מקרה, הראו מהו הקוד שנוצר.
האם התקבל קוד זהה בחלק מן המקרים? אם כן, הסבר מדוע.

שאלה 4 (תוכנית מחשב - 40%)

בנו מנתח תחבירי וסמנטי לשפת OULP.
קלט: תוכנית בשפת OULP.
פלט:

א. הדפסת הקלט.
ב. הדפסת פעולות סמנטיות.
ג. הודעת שגיאה במקרה הצורך או הודעה על תקינות הקלט.
ד. קובץ oulp.par המכיל את כל הכללי גזירה שמנתח תחבירי השתמש בהם.

הממשק
המפסק יקרא ouplpr.exe
המפסק יופעל משורת הפקודה של DOS.
קלט – התוכנית מקבלת כפרמטר יחיד שם של קובץ קלט (קובץ טקסט המכיל משפט בשפת OULP). הסיומת של שם קובץ הקלט חייבת להיות oulp או OULP.
שורת הפקודה היא: ouplpr <file_name>.oupl

<u>הגשה</u>

יש להגיש דיסקט/דיסק ועליו:

- קובץ ouplpr.exe.
- הדפסה של קובץ ouplpr.c.

שאלון למטלת מנחה (ממ״ין)15

שם הקורס: **קומפילציה**

שם המטלה: **ממ״ן 15**	**מס׳ הקורס: 20364**
משקל המטלה: **3 נקודות**	**מס׳ המטלה: 15**
מספר השאלות: **4**	**מחזור: א 2014**

מועד משלוח המטלה: 5.1.2014

אנא שים לב:
מלא בדייקנות את הטופס המלווה לממ״ן
בהתאם לדוגמה המצויה בפתח חוברת המטלות
העתק את מספר הקורס ומספר המטלה הרשומים לעיל

שאלה 1 (20%)

נתונה התכנית הבאה בפסאודו-קוד:

```
Program main()
    procedure p(x,y,z);
        begin
            y:=y+1;
            z:=z+x;
            x:=z+2;
        end;
    begin
        a:=2;
        b:=3;
        p(a+b,a,b);
        print(a);
    end.
```

מה תדפיס התכנית כאשר העברת הפרמטרים היא לפי:

א. ערך - call by value

ב. מען - call by reference

ג. העתקה - copy restore

אין צורך לנמק.

שאלה 2 (30%)

הראה כיצד ניתן לתרגם ביטויים בולאניים של שפת CPL:

1. תרגמו את הביטוי הבא לשפת
   a. לשפת הרביעיות (quadruples)
   b. לשפת השלשות (triples)
   c. לשפת QUAD

-(a+b)*(c*d)+(a-b*c)

2. הראו כיצד ניתן לתרגם לשפת Quad

א. את הכלל הבא מהפרויקט הסופי

VAL → id assignop ival(EXPRESSION);

ב. משפט מהסוג:

while (BOOLEXPR) do Statement

שאלה 3 (30%)

נגדיר שפה לכתיבת תוכניות פשוטות, המכילות הצהרות על משתנים ופקודות השמה.
לדוגמה:

```
1.   int   x , b ;
2.   x := b ;
3.   {  float   x  , c ;
4.      x := x + c  ; }
5.    b := x + b ;
6.   {  int   b , c ;
7.      x := c + b ;
8.      b := x;
9.        {   int   a ;
10.           a := x ;  }
11.  }
12.  x := x + b ;
```

המשתנים בשפה יכולים להיות מטיפוס int או real.
כל הצהרה מורכבת מטיפוס ואחריו רשימת משתנים.
בפקודות ההשמה מופיע באגף שמאל משתנה, ובאגף ימין ביטוי המורכב מסכום של כמה משתנים.

כמו בשפת C, ניתן להגדיר בשפה זו גושים (בלוקים) – כל גוש מתחיל בסימן "{" ומסתיים ב-"}".
בתוך כל גוש ניתן להגדיר משתנים, שהם מקומיים לאותו גוש, וייתכן קינון של גושים. תחום ההשפעה של ההצהרות נקבע לפי "כלל הקינון הקרוב ביותר" (most closely nested rule), כפי שמוסבר בספר בעמ' 412.

טיפוסו של ביטוי:
אם הביטוי מכיל רק משתנים מטיפוס int, הטיפוס של הביטוי כולו הוא int.
אם קיים בביטוי משתנה אחד לפחות מטיפוס real, הטיפוס של הביטוי כולו הוא real.

חוקיותה של השמה:
השמה היא חוקית אם הטיפוס של הביטוי באגף ימין זהה לטיפוס של המשתנה באגף שמאל.
נגדיר דקדוק G אשר גוזר תוכניות בשפה המתוארת:

1. **P → L S**
2. **L → L D**
3. **L → ε**
4. **D → T N ;** **/* Note the ";" at the end */**
5. **N → N , id**
6. **N → id**
7. **T → real**
8. **T → int**
9. **S → S C**
10. **S → ε**
11. **C → id := E ;** **/* Note the ";" at the end */**
12. **C → { L S }**
13. **E → E + id**
14. **E → id**

כתבו סכמת תרגום המבוססת על הדקדוק G, אשר מבצעת את הפעולות הבאות:

לכל הצהרה, המשתנים המוצהרים מוכנסים לטבלת הסמלים, יחד עם הטיפוס שלהם.
לפקודות ההשמה מתבצעת בדיקת חוקיות (לפי הכללים שתוארו לעיל).
בכל מקרה מודפסת הודעה, המציינת האם התוכנית היא חוקית (כלומר האם כל ההשמות שמופיעות בה הן חוקיות, על סמך ההצהרות).

הנחיות:
טבלת הסמלים צריכה להיות בנויה כך שתאפשר מימוש נכון של חוקי התחום. הסבירו את מבנה הטבלה שעליו מסתמכת סכמת התרגום שלכם (תוכלו להיעזר ברעיון הנתון בספר בעמ' 475, או להציע רעיונות אחרים).
הגדירו פעולות על טבלת הסמלים לפי הצורך, למשל: insert, lookup, make_table, init. תוכלו להשתמש בפעולות אלה בסכמת התרגום שלכם. לכל פעולה שאתם מגדירים, רשמו הסבר מפורט על מטרת הפעולה, כיצד היא מתבצעת, מה הפרמטרים שהיא מקבלת ומה הערך שהיא מחזירה (רשמו הסבר בעברית, אין צורך לכתוב קוד).
מותר להגדיר תכונות למשתני הדקדוק לפי הצורך
הניחו שהתכונה id.name מכילה את שמו של המשתנה id, והיא מסופקת על-ידי המנתח הלקסיקלי.

ב. ציירו את טבלת הסמלים כפי שתראה בסוף הידור התוכנית הנתונה בתחילת השאלה.

שאלה 4 (20%)

א. נתון מערך דו מימדי A:

type A[3][5]

נניח שגודל של type בזיכרון 5 ביט(byte).
נתון שאיבר ראשון במערך A[0][0] נמצא בתא 1000 בזכרון.
איפה יהיה האיבר A[2][2]?

ב. תרגמו את התוכנית הבאה בשפת C לשפת שלושה מענים:

```
int main()
{
int j;
real a[20];
j=0;
while(j<=20)   a[j++]=0;
}
```

הערות:

- קבוע c מסמן את בסיס המערך a (base).
- int דורש 2 bytes
- real דורש 4 bytes

שאלון למטלת מנחה (ממ״ין)16

שם הקורס: קומפילציה	**מס׳ הקורס: 20364**
שם המטלה: ממ״ן 16	**מס׳ המטלה: 16**
משקל המטלה: 15 נקודות	**מחזור: א 2014**

מספר השאלות: 1

מועד משלוח המטלה: 24.1.2014

אנא שים לב:
מלא בדייקנות את הטופס המלווה לממ״ן
בהתאם לדוגמה המצויה בפתח חוברת המטלות
העתק את מספר הקורס ומספר המטלה הרשומים לעיל

שאלה 1 (100%)

1. פרוייקט המהדר

בפרוייקט זה עליכם לתכנן ולממש חלק קדמי של מהדר, המתרגם תוכניות משפת המקור CPL לשפת הביניים Quad.
שפת המקור CPL (Compiler Project Language) היא שפה דמוית פסקל או C, אך מוגבלת מהן בהרבה.
שפת הביניים Quad היא שפת רביעיות דמוית אסמבלר.
השפות תוגדרנה בסוף המטלה.

2. תיאור פעולת המהדר

2.1. מה עושה המהדר?

המהדר יבצע את כל שלבי ההידור (החלק הקדמי) כפי שנלמדו בקורס, החל בניתוח לקסיקלי, דרך ניתוח תחבירי ובדיקות סמנטיות, ועד לייצור קוד ביניים בשפת Quad.
ההידור יכלול טיפול בשגיאות, כפי שיפורט בהמשך.

המהדר יקבל קובץ קלט המכיל תוכנית בשפת CPL. כפלט, ייצר המהדר קובץ המכיל תוכנית בשפת Quad.
בנוסף, ייצר המהדר קובץ ״רישום״ (listing), שבו יפורטו שגיאות שהתגלו במהלך ההידור.
(תוכלו להריץ בפועל את תוכניות ה-Quad הנוצרות, בעזרת המפרש qx שקיבלתם – ראו הסבר בסעיף המתאר את שפת Quad בסוף המטלה).

2.2. מפסק

בפרויקט זה עליכם לממש מפסק בשיטת הניתוח העולה, כלומר באחת משיטות LR שנלמדות בספר: SLR, LALR, או LR קנוני ((LR(1)).

תוכלו לבחור באחת משתי אפשרויות לבניית המפסק:
1. שימוש בתוכנת bison
2. בניית מנתח תחבירי לפי אחת מהשיטות המפורטות לעיל, ללא שימוש בכלים אוטומטיים

אם תבחרו לכתוב בעצמכם מנתח תחבירי, ללא שימוש ב-bison, יהיה עליכם לבנות את טבלת המפסק. כיון שמדובר בדקדוק גדול למדי, קשה לבנות את הטבלה באופן ידני.
ניתן להיעזר ב-bison לצורך בניית טבלת LALR (בלי להיעזר בו לבניית המפסק כולו).

2.3. הממשק

המהדר יהיה תוכנית המופעלת משורת הפקודה של DOS.
שמו של המהדר הוא cpq (קיצור של CPL to Quad).
קובץ הריצה צריך להיקרא cpq.exe.
הקובץ עם הפונקציה הראשית של המהדר (main) צריך להיקרא cpq.c.
קלט – המהדר מקבל כפרמטר יחיד שם של קובץ קלט (קובץ טקסט המכיל תוכנית בשפת CPL).
הסיומת של שם קובץ הקלט חייבת להיות cpl או CPL.
שורת הפקודה היא: cpq <file_name>.cpl
פלט – המהדר יוצר שני קובצי טקסט עם שם זהה לשם קובץ הקלט ועם סיומות כדלקמן:
קובץ "רישום" (listing), עם סיומת lst או LST: אל הקובץ הזה מעתיק המהדר את התוכנית (ללא ההערות), ומדפיס אותה כאשר בתחילת כל שורה רשום מספר השורה.
בקובץ זה מופיעות גם הערות השגיאה (אם ישנן).
אם תוכנית הקלט תקינה – קובץ רביעיות, עם סיומת qud או QUD: קובץ זה מכיל את תוכנית ה-Quad שנוצרה. אם תוכנית הקלט מכילה שגיאה כלשהי (לקסיקלית, תחבירית או סמנטית) אין לייצר קובץ qud, גם לא קובץ ריק.

טיפול בשגיאות ממשק – במקרה של שגיאה בפרמטר הקלט, בפתיחת קבצים וכדומה, יש לסיים את הביצוע בצירוף הודעת שגיאה מתאימה למסך (stderr). במקרה כזה אין לייצר קובצי פלט.
כחלק מהטיפול בשגיאות ממשק, יש לוודא שהסיומת של קובץ הקלט היא נכונה.

שורת חותמת – יש לכתוב שורת "חותמת" עם שם הסטודנט, אשר תופיע במקומות הבאים:
ב-stderr
בקובץ ה-LST (בתחילתו)
בקובץ ה-QUD – אחרי הוראת ה-HALT האחרונה, וזאת כדי לא להפריע למפרש של שפת Quad.

שימו לב: יש להקפיד היטב על כל הוראות הממשק
חשוב שממשק המהדר שתכתבו יהיה בדיוק כפי שמוגדר במטלה. ייתכן שבדיקת הריצה של תוכניתכם תיעשה בצורה אוטומטית, ובמקרה כזה תוכנית בעלת שם שונה או ממשק שונה תיכשל. וודאו גם שניתן להריץ את תוכניתכם כאשר נמצאים במדריך אחר, ולא המדריך שבו נמצאת התוכנית עצמה.

אין לכתוב מהדר המסתמך על קיומם של קבצים נוספים, כגון קובץ המכיל את טבלת הפיסוק.

אם המהדר מייצר קובצי-עזר בזמן הריצה, יש לדאוג למחיקת הקבצים הללו בסיום הריצה.

2.4. טיפול בשגיאות

ייתכן שתוכנית הקלט תכיל שגיאות מסוגים שונים:
שגיאות לקסיקליות
שגיאות תחביריות
שגיאות סמנטיות

שימו לב:
במקרה של קלט המכיל שגיאה (מכל סוג שהוא) אין לייצר קובץ qud, גם לא קובץ qud ריק.
לאחר זיהוי של שגיאה לקסיקלית , תחבירית או סמנטית, יש להמשיך בהידור מהנקודה שאחרי השגיאה.

3. מימוש המהדר – כלים, שיטות ומבני נתונים

3.1. שימוש בכלים flex ו-bison

במטלה זו יש באפשרותכם לשלב את השימוש בכלים האוטומטיים flex ו-bison. flex הוא כלי אשר מייצר באופן אוטומטי מנתחים לקסיקליים, bison הוא כלי לייצור אוטומטי של מנתחים תחביריים.
אין הכרח להשתמש בתוכנות אלו, וניתן גם לכתוב בעצמכם את המנתח הלקסיקלי והמנתח התחבירי, ללא שימוש בכלים אוטומטיים.

סטודנטים שרוצים להשתמש בכלים אוטומטיים ילמדו אותם בעצמם ממדריך שנשלח לביתכם.
אין לפנות למרכזת הוראה או מנחים בקשר לכלים האוטומטיים flex ו-bison.

3.2 חישוב יעדי קפיצה

בקוד הרביעיות שמייצר המהדר עשויות להופיע פקודות JUMP או JMPZ, כאשר יעד הקפיצה הוא מספר שורה. לצורך חישוב יעדי הקפיצה, ייתכן שתבחרו להשתמש בהטלאה לאחור, או בשיטה של ייצור קוד זמני המכיל תוויות סימבוליות (מחרוזות), ומעבר נוסף על הקוד כדי להחליף את התוויות הסימבוליות במספרי שורות.
לצורך מימוש השיטה שבה תבחרו תוכלו להחליט להחזיק בזיכרון את כל הקוד המיוצר, או שתוכלו לייצר קבצים זמניים, שבהם ייכתב הקוד בשלבי הביניים של הייצור.

3.4. שיקולי מימוש

כתיבת המהדר נועדה להיות תהליך לימודי, ותכנון מבני הנתונים והאלגוריתמים שלו צריך להיגזר מכך. אין לקבל החלטות מימוש עיקריות על סמך הנסיבות שבהן המהדר יורץ בפועל על ידי בודק המטלה. לדוגמה, נדרש שמימוש טבלת הסמלים יהיה מתוחכם יותר מאשר חיפוש לינארי ברשימה, למרות שכאשר יש מספר קטן של מזהים, זהו פתרון סביר.

ברוח זו נוסיף: במימוש המבנים שגודלם תלוי בקלט יש להעדיף הקצאת זיכרון דינמית על-פני הקצאה סטטית שגודלה חסום ונקבע מראש. לעומת זאת, במימוש המבנים שגודלם קבוע וידוע מראש עדיפה כמובן הקצאה סטטית. במבנים אלה יש גם להעדיף מימוש "מונחה טבלה" , שבו מאוחסן המידע ב"טבלה" נפרדת, והקוד משמש לגישה לטבלה ולקריאתה.

טבלת המפסק (במקרה של מימוש מפסק ללא bison) – לא הכרחי לממש דחיסה של הטבלה.
מחסנית המפסק (במקרה של מימוש ללא bison) – אסור לממש את המחסנית בצורה שעלולה להגביל את גודלה.
טבלת הסמלים – אסור לממש את הטבלה בצורה שעלולה להגביל את גודלה. בנוסף, יש לדאוג לכך שחיפוש והוספה בטבלה יהיו מהירים.
המנתח הלקסיקלי – ייקרא על ידי המנתח התחבירי, ויחזיר בכל פעם אסימון אחד בלבד.
חוצץ הקלט – מותר להשתמש בחוצץ (buffer), שלתוכו ייקרא קובץ הקלט כולו.
אם אתם מניחים חסם על גודלו של קובץ הקלט, חשוב מאוד לרשום הנחה זו בתיעוד, וכן בקובץ readme שאותו תגישו. כדאי שהגודל יהיה לפחות 10K.

בדיקות סמנטיות ויצירת קוד – יש לבצע על ידי מימוש של הגדרה מונחית-תחביר מתאימה.

3.5. סגנון תכנות

התוכנית שתכתבו צריכה לעמוד בכל הקריטריונים הידועים של תוכנית כתובה היטב: קריאות, מודולריות, תיעוד וכו'.

4. כיצד להגיש את הפרוייקט

4.1. תיעוד

יש לכתוב תיעוד בגוף התוכנית, כמקובל. תיעוד זה נועד להקל על קוראי התוכנית.

בנוסף, יש לכתוב **תיעוד נלווה**: מסמך נפרד, שאותו ניתן לקרוא באופן עצמאי, ללא קריאת התוכנית עצמה. ניתן לכתוב את התיעוד הנלווה בעברית או באנגלית.

לתיעוד הנלווה שתי מטרות עיקריות: הסברים על שיקולי המימוש, ותיאור מבנה הקוד. יש להציג דיון ענייני בשיקולי המימוש.

התיעוד אינו מיועד למשתמש נאיבי של המהדר, אלא לבודק המטלה, המכיר היטב (יש לקוות) את נושאי הקורס.

אין לחזור על דברים מובנים מאליהם, כגון שיש פעולות shift ו-reduce, אלא להבהיר את ההתלבטויות בין פתרונות שונים, ולהצדיק את ההחלטות שנעשו. בין השאר, יש לדון בנקודות אלה:

- מימוש טבלת האוטומט במנתח הלקסיקלי (אם אין שימוש ב-flex).
  אם מימשתם אוטומט ללא טבלה, יש להסביר את אופן המימוש.
- שיטת הניתוח התחבירי שנבחרה.
- מימוש טבלת המפסק ומחסנית המפסק (אם אין שימוש ב-bison).
  אם מימשתם מפסק ללא טבלה, יש להסביר את אופן המימוש.
- מבנה הנתונים שנבחר לשמש כטבלת סמלים.
- מתי מעודכנת טבלת הסמלים, והאם מלים שמורות מוכנסות לטבלה זו.
- כיצד מטפלים בשגיאות.
- שיטת החישוב של יעדי קפיצה בקוד הרביעיות.

לגבי הקוד, יש לתאר את החלוקה למודולים, תלויות הדדיות בין מודולים, מבנה זרימת הבקרה, פונקציות ראשיות, וכד'.

חשוב מאוד לציין בתיעוד הנלווה מגבלות שהוספתם (כגון הנחת חסם על מספר המשתנים שיופיעו בתוכנית הקלט) או חריגות מהממשק שתואר. חריגות כאלה אינן רצויות מלכתחילה (וישפיעו על הציון...), אך הן חמורות במיוחד אם אינן מתועדות, ומקשות על המשתמש במהדר.

4.2. מה להגיש

1. <u>הדפסה של התוכנית</u> – (קובצי המקור – ראו הסבר בהמשך).
   חשוב לארגן את ההדפסה בצורה שתקל על הקורא. למשל – ליצור הפרדה ברורה בין הקבצים השונים, ולסמן את שמות הקבצים.
2. <u>תיעוד נלווה</u> – יש להגיש עותק על נייר,וגם יש צורך להגיש קובץ.
3. <u>תרשים זרימה</u> – יש להגיש דיאגרמת זרימת נתונים (DFD).
4. <u>תקליטון DOS</u> – התקליטון צריך להכיל את המדריכים והקבצים הבאים:
   במדריך \ (root directory):

- readme : קובץ טקסט פשוט, שיכלול את המידע הבא:
  - הוראות מדויקות להידור של תוכניתכם ויצירת cpq.exe
  - הנחיות מיוחדות, אם קיימות, להרצת תוכניתכם, או מגבלות מיוחדות.
  - תיאור מבנה המדריכים בדיסקט/דיסק, ומה מכיל כל אחד מהם.

במדריך src\:

- קובצי המקור של התוכנית שכתבתם, כלומר כל קבצי ה-c וה-h. אם השתמשתם ב-flex או bison, יש לכלול גם את קובצי הקלט שיצרתם עבורם, וכמובן את קובצי הפלט שנוצרו.
  אין צורך להגיש הדפסה של קובצי הפלט שנוצרים מ-flex או bison.
- קבצים הנחוצים כדי להדר את תוכניתכם, כגון קובץ project או קובץ make.

קראו בעיון את סעיף 9 (העוסק בנוהל הגשת מטלות), בחלק הראשון של חוברת זו.

שימו לב: יש להפקיד היטב על כל הוראות ההגשה.

4.3. הידור

קראו בעיון את סעיף 4 (העוסק במחשבים ותוכנות לכתיבת מטלות), בחלק הראשון של חוברת זו.

חשוב שבודק המטלה יוכל לבצע הידור לתכניתכם. תוכלו לבחור כיצד תיעשה פעולת ההידור, מתוך האפשרויות הבאות:

- בעזרת make
- מתוך Visual C++ או Turbo C
- למשתמשי Turbo C – מתוך שורת הפקודה של DOS: "tc ..." .

בכל מקרה, עליכם לרשום בקובץ readme הוראות מדויקות להידור תכניתכם.

4.4. בדיקת התכנית לפני ההגשה

לקלטים תקינים, ייווצר קובץ qud המכיל תוכנית בשפת Quad. השתמשו במפרש של שפת Quad, שנקרא qx, אשר נמצא בתקליטון שקיבלתם בתחילת הקורס. בעזרתו תוכלו להריץ את תוכנית ה-Quad שיצרתם וכך לבדוק את תקינות הקוד המיוצר. כמו כן, תוכלו להיעזר בו כדי להבין את שפת Quad – תוכלו לכתוב תוכניות דוגמה קטנות בשפת Quad, ולהריץ אותן ב-qx.
בנוסף לקלטים תקינים, נסו תכניות קלט עם שגיאות (לקסיקליות, תחביריות וסמנטיות), כולל תוכניות המכילות יותר משגיאה אחת.

4.5. משלוח

גודלו הפיזי של הפרוייקט אינו מאפשר, בדרך כלל, לשלוח אותו במעטפת ממ"ן רגילה. אפשר לשלוח אותו במעטפה כלשהי, בגודל הנדרש.
יש לצרף טופס מלווה לממ"ן, כרגיל.
שימו לב: גודלו הפיזי של הפרוייקט גם אינו מאפשר למנחה לקבל אותו בתיבת הדואר בביתו. לכן אין לשלוח את הפרוייקט ישירות אל המנחה. את כל הפרוייקטים יש לשלוח אל מרכזת הקורס באו"פ – משם הם יועברו אל המנחים. הפרוייקט ייבדק על-ידי המנחה, כמו כל מטלה אחרת.
שילחו את הפרוייקט לכתובת הבאה:

ד"ר רינה צביאל-גירשין
מדעי המחשב
האוניברסיטה הפתוחה
רבוצקי 108
רעננה ת.ד. 808
43107

רשמו את שם המנחה שלכם במקום בולט על המעטפה.

5. כיצד ייבדק הפרוייקט

5.1. תהליך הבדיקה

הבדיקה תכלול הרצה של המהדר שלכם על קלטים רבים, קריאת התיעוד הנלווה, וקריאה חלקית של קובצי המקור. בדיקות הריצה יכללו, בין השאר:

- הרצה על קלט תקין ובדיקת הפלט (באמצעות המפרש של Quad, וגם בדיקה יבשה).
- בדיקת התגובות לשגיאות ממשק (פרמטר שגוי, סיומת השם שגויה, וכו').
- בדיקת ההתמודדות עם שגיאות לקסיקליות, סמנטיות ותחביריות.
- איכות מימוש טבלת הסמלים והמחסנית (לדוגמה, בדיקת יכולת הטיפול של המהדר בתכנית קלט שיש בה מספר גדול - 250 לפחות - של מזהים).

5.2. חלוקת הציון

30-35%: ביצועי המהדר על קלטים תקינים.
25-30%: טיפול בשגיאות מכל הסוגים.
20%: החלטות מימוש, בחירת מבני נתונים, מודולריות, כתיבת קוד כנדרש.
20%: תיעוד והגשה בהתאם לנדרש.

שפת המקור – שפת התכנות CPL
(Compiler Project Language)

1. מבנה לקסיקלי

בשפה CPL ישנם אסימונים הבאים:

Reserved words

```
do     for  if    int     ival    then
otherwise    print procedure     real     read    rval  variable
while
```

Reserved symbols

```
(   )   {   }
,   :   ;   !
```

Composed tokens

```
id:             letter (letter|digit)*
num:            digit+  | digit+.digit*
relop:          ==  |  != | <  |  >  | >= | <=
addop:          +   |  -
mulop:          *   |  /
assignop:         :=
orop:           or
andop:          and

Where:  (Note: digit and letter are not tokens)
   digit:  0  |  1  | ... |  9
   letter: a  |  b  | ... |  z  |  A  |  B  | ... | Z



   letter: a  |  b  | ... |  z  |  A  |  B  | ... | Z
```

הבהרות:

1. בין האסימונים יכולים להופיע תווי רווח (space), תווי טאב (\t) או תווים המסמנים שורה חדשה (\n).
   תווים כאלה חייבים להופיע כאשר הם נחוצים לצורך הפרדה בין אסימונים (למשל, בין מלה שמורה לבין מזהה). בשאר המקרים, האסימונים יכולים להיות צמודים זה לזה, ללא רווח.
2. אורכו של מזהה id מוגבל – עד 9 תווים בלבד.
3. הערות בתוכנית מופיעות בין הגבולות /* .... */ (כמו בשפת C).
   מותר להניח שבקלט שתקבלו אין הערה שגולשת מעבר לסוף שורה, ואין הערה שמכילה את הרצף "/*" (סימן של סוף הערה) בתוכה, לפני סופה.

2. מבנה תחבירי:

CPLG - Grammar for the programming language CPL

```
PROGRAM   →  procedure  id  {  DECLARATIONS  STMTLIST }

DECLARATIONS →  variable  DECLARLIST
               | ε

DECLARLIST →  DECLARLIST   TYPE : IDENTS
             | TYPE : IDENTS

IDENTS →  id , IDENTS
             | id;

TYPE →     int
           | real

STMTLIST  →  STMTLIST  STMT
               | ε

STMT → ASSIGNMENT_STMT
       | VAL
       | CONTROL_STMT
       | READ_STMT
       | WRITE_STMT
       | STMT_BLOCK

WRITE_STMT → print(EXPRESSION);

READ_STMT → read(id);

ASSIGNMENT_STMT → id assignop EXPRESSION;

VAL → id assignop ival(EXPRESSION);
      /* Returns integer value of expression*/
      id assignop rval(EXPRESSION);
  /* Returns real value of expression – converts integer to real*/


CONTROL_STMT → if (BOOLEXPR) then STMT otherwise STMT
               | while (BOOLEXPR) do STMT
               | for(ASSIGNMENT_STMT; BOOLEXPR; STEP) STMT

STMT_BLOCK → { STMTLIST }

STEP → id assignop id addop num
         | id assignop id mulop num

BOOLEXPR → BOOLEXPR orop BOOLTERM
         |   BOOLTERM

BOOLTERM → BOOLTERM andop BOOLFACTOR
         |   BOOLFACTOR

BOOLFACTOR →  ! (BOOLFACTOR) /*Meaning not BOOLFACTOR*/
```

```
            |EXPRESSION  relop  EXPRESSION

EXPRESSION → EXPRESSION  addop  TERM
             | TERM

TERM → TERM mulop FACTOR
        | FACTOR

FACTOR → ( EXPRESSION )
          | id
          | num
```

3. סמנטיקה

א. כל משתנה מוצהר רק פעם אחת במהלך התוכנית.

ב. קבועים מספריים שאין בהם נקודה עשרונית הם מטיפוס int. אחרת הם מטיפוס float.

ג. הטיפוס של ביטויים נקבע על ידי הארגומנטים המופיעים בהם.

1. כאשר בביטוי מופיע משתנה או קבוע מטיפוס float, הטיפוס של הביטוי כולו הוא float.
2. בכל מקרה אחר טיפוס הביטוי כולו הוא int.
3. חילוק בין שני שלמים נותן את המנה השלמה שלהם.

ד. פעולת השמה היא חוקית כאשר שני אגפיה הם מאותו טיפוס או שהאגף השמאלי הוא float.

ה. משמעות השפה וקדימות האופרטורים הם סטנדרטיים, כמו בשפת C.

4. תוכנית לדוגמה:

```
procedure  Min  /* Finding minimum between two numbers */
{
variable float:a,b;

      read(a);
      read(b);
      if (a<b) then print(a);
           otherwise print(b);
}
```

שפת המטרה – Quad

Quad היא שפת רביעיות דמוית אסמבלר.

היא מכילה הוראות שלהן בין אפס לבין שלושה ארגומנטים. תוכנית היא סדרה של הוראות בשפה. הפורמט המחייב של תוכנית הוא:

- הוראה אחת בכל שורה - ההוראות עצמן כתובות תמיד **באותיות גדולות**. שמות המשתנים מכילים רק **אותיות קטנות, מספרים ו/או קו תחתון** _.
- קוד ההוראה והארגומנטים מופרדים על ידי תו רווח אחד לפחות.
- בכל תוכנית מופיעה ההוראה HALT לפחות פעם אחת, בשורה האחרונה.

ישנם שלושה סוגי ארגומנטים להוראות השפה:

1. **משתנים**. המשתנים מיוצגים כמזהים. הגדרתם היא כמו בשפה CPL מלבד הבדל אחד: בשם המשתנה יכולים להופיע גם סימני קו תחתון _ (underscore).
2. **קבועים מספריים** (מטיפוס שלם או ממשי) הגדרתם זהה להגדרתם בשפת CPL.
3. **תוויות:** נרשמות כמספר שלם המסמן מספר סידורי של הוראה בתכנית (החל מ-1).

למשתנים ולקבועים בשפת Quad יש טיפוס - שלם או ממשי. טיפוס של משתנה איננו יכול להתחלף במהלך התוכנית. ישנן הוראות שונות עבור שלמים ועבור ממשיים. אין לערבב בין הטיפוסים. קיימות גם שתי הוראות המאפשרות מעבר בין שלמים וממשיים.

בשפה אין משתנים בולאניים, הוראות השוואה מחשבות מספר: 1 עבור True ו- 0 עבור False. כמו כן קיימת הוראת קפיצה בלתי מותנית והוראת קפיצה מותנית. (המבצעת למעשה הוראת "if not … goto …").

הוראות שפת Quad

בטבלה הבאה:
A מציין משתנה שלם
B ו- C מציינים משתנים שלמים או קבועים שלמים
D מציין משתנה ממשי
E ו- F מציינים משתנים ממשיים או קבועים ממשיים.
L מציין תווית (מספר שורה).

<u>שימו לב:</u> A, B, C, D, E, F הם סימנים מופשטים, שיכולים לציין משתנה כלשהו. המשתנים המופיעים בפועל בתוכנית צריכים להיכתב באותיות קטנות.

Opcode	Arguments	Description
IASN	A B	A := B
IPRT	B	Print the value of B
IINP	A	Read an integer into A
IEQL	A B C	If B=C then A:=1 else A:=0
INQL	A B C	If B<>C then A:=1 else A:=0
ILSS	A B C	If B<C then A:=1 else A:=0
IGRT	A B C	If B>C then A:=1 else A:=0
IADD	A B C	A:=B+C
ISUB	A B C	A:=B-C
IMLT	A B C	A:=B*C
IDIV	A B C	A:=B/C

RASN	D E	D := E
RPRT	E	Print the value of E
RINP	D	Read a real into D
REQL	A E F	If E=F then A:=1 else A:=0
RNQL	A E F	If E<>F then A:=1 else A:=0
RLSS	A E F	If E<F then A:=1 else A:=0
RGRT	A E F	If E>F then A:=1 else A:=0
RADD	D E F	D:=E+F
RSUB	D E F	D:=E-F
RMLT	D E F	D:=E*F
RDIV	D E F	D:=E/F

ITOR	D B	D:= real(B)
RTOI	A E	A:= integer(E)

JUMP	L	Jump to Instruction number L
JMPZ	L A	If A=0 then jump to instruction number L else continue.

HALT		Stop immediately.

תכנית לדוגמה בשפת Quad

```
procedure  Min  /* Finding minimum between two numbers */
{
variable float:a,b;

      read(a);
      read(b);
      if (a<b) then print(a);
           otherwise print(b);
}
```

תרגום לשפת QUAD נראה כך:

RINP a
RINP b
RLSS less a b
JMPZ 6 less
JMP 8
RPRT a
JMP 9
RPRT b
HALT

- הפקודות רשומות באותיות גדולות, שמות משתנים באותיות קטנות.
- אין צורך להצהיר על משתנים (טיפוס המשתנה מוגדר אוטומטית לפי סוג הפעולה שמופעל עליו).

QX – מפרש לשפת QUAD

התקליטון שקיבלתם מכיל מפרש (interpreter) לשפת Quad, שנקרא qx.
התוכנית נמצאת ב- \quad\qx.exe . באותו מדריך נמצא גם קובץ הסברים להפעלת qx.

המפרש qx מקבל קובץ המכיל תוכנית Quad, ומריץ את התוכנית. בעזרת qx תוכלו להריץ בפועל את תוכניות ה-Quad שייצור המהדר שלכם, או תוכניות Quad שתכתבו ידנית.

כדאי לשים לב לנקודות הבאות:

אין למספר את השורות בתוך קובץ ה-Quad. מספרי השורות אינם חלק מתוכנית ה-Quad, ו-qx יכריז עליהם כעל שגיאה.
כאשר qx נתקל בפקודת IINP או RINP, הוא מדפיס סימן "?" למסך, ומחכה לקבלת קלט מהמשתמש.

שימוש בתוכנת bison -

במטלה זו יש באפשרותכם לשלב את השימוש בתוכנת bison. זהו כלי עזר לבנייה אוטומטית של מנתחים תחביריים. תוכנת bison מקבלת כקלט קובץ עם הגדרת הדקדוק של השפה, ומייצרת תכנית בשפת C, שהיא מנתח תחבירי עבור השפה המוגדרת.

אם תחליטו להשתמש ב-bison, תוכלו להיעזר בחוברת "מדריך למשתמש ב-flex/bison" שנשלחה אליכם, וכן בסעיף 4.9 בספר הלימוד. **נושא זה לא יילמד במפגשי ההנחיה.**
(שימו לב: ייתכן ששמות הקבצים במערכת DOS שונים במקצת משמות הקבצים המוזכרים ב-"מדריך למשתמש").

תוכנת bison נמצאת במדריך bison\ בתקליטון שנשלח אליכם.
בסוף מטלה זו תמצאו כמה הערות לגבי השימוש ב-bison.

מימוש טבלת המפסק והמחסנית

(למי שאינו משתמש ב-bison).
מותר לממש את טבלת המפסק בעזרת מערך דו-ממדי פשוט. לא נדרשת דחיסה של הטבלה.
יש לממש את מחסנית המפסק בצורה שלא תגביל את גודלה, כלומר צריך להשתמש בהקצאת זיכרון דינמית.

סגנון תכנות

התוכנית שתכתבו צריכה לעמוד בכל הקריטריונים הידועים של תוכנית כתובה היטב: קריאות, מודולריות, תיעוד וכו'.

בדיקת התכנית לפני ההגשה

בדקו היטב את ריצת התוכנית שלכם על קלטים מגוונים (לא רק תוכניות הדוגמה המסופקות לכם בתקליטון). נסו תוכניות קלט עם שגיאות שונות, כולל מקרי קצה.

הגשה

במטלה זו יש להגיש <u>קובצי המקור</u>:

- תכנית שנכתבה ללא bison: יש להגיש את כל קובצי התוכנית (קובצי c ו-h).
- תכנית שנכתבה בעזרת bison: יש להגיש את קובץ הקלט של bison שכתבתם, את הקבצים ש-bison מייצר כפלט, וכל קובץ אחר (c או h) שהוא חלק מהתוכנית.
  אין צורך להגיש הדפסה של קובץ הפלט שמייצר bison.

הידור

קראו בעיון את סעיף 4 (העוסק במחשבים ותוכנות לכתיבת מטלות), בחלק הראשון של חוברת זו.

<u>חשוב שבודק המטלה יוכל לבצע הידור לתכניתכם</u>. **<u>תוכנית שלא תעבור הידור לא תיבדק.</u>**
רשמו בקובץ readme הוראות מדויקות להידור תכניתכם.

דוגמת הרצה: קלט

קובץ CPL (התוכנית מכילה שגיאה):

```
procedure    Example{
variable          int:x;

read(x)  ;
if    x >= 0) then
                        print(x);
otherwise           print(0-x);
}
```

דוגמת הרצה: פלט

קובץ LST:

```
1.          procedure  Example{
2.          variable    int:x;
3.
4.          read(x)  ;
5.          if     x >= 0) then
6.                          print(x);
7.          otherwise       print(0-x);
8.          }
```

ERROR line 5, column 4: Expected '(', found identifier instead.

הערות על השימוש ב-bison

ייתכן שתתקלו בבעיה כאשר תנסו להדר את קובץ ה-C שנוצר ע"י bison.

נניח שיצרתם קובץ קלט ל-bison ששמו pars.y. במקרה זה bison ייצר קובץ פלט בשם pars_tab.c. כאשר תנסו להדר את הקובץ הזה, ייתכן שתקבלו הודעות שגיאה על כך שהפונקציה alloca אינה מוגדרת (זוהי פונקצית ספריה של C).

ניתן לפתור את הבעיה על-ידי הכנסת שורות מתאימות לקובץ pars.y – יש לרשום שורות אלו באזור ה-"C declarations" שבתחילת הקובץ.

<u>למשתמשים ב- Turbo C:</u>
הוסיפו את השורה #define alloca malloc

<u>למשתמשים ב- Visual C++:</u>
הוסיפו את השורה #include <malloc.h>
(ייתכן שיש להוסיף גם #define alloca _alloc)

בהצלחה