שאלון למטלת מנחה (ממ״ן)16

שם הקורס: קומפילציה	**מס׳ הקורס: 20364**
שם המטלה: ממ״ן 16	**מס׳ המטלה: 16**
משקל המטלה: 15 נקודות	**מחזור: א 2013**

מספר השאלות: 1

מועד משלוח המטלה: 25.01.2013

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 (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.
<u>קלט</u> – 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
<u>פלט</u> – המהדר יוצר שני קובצי טקסט עם שם זהה לשם קובץ הקלט ועם סיומות כדלקמן:
<u>קובץ "רישום" (listing), עם סיומת lst או LST</u>: 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).
<u>אם תוכנית הקלט תקינה – קובץ רביעיות, עם סיומת qud או QUD</u>: 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.

<u>טיפול בשגיאות ממשק</u> – 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.

<u>שורת חותמת</u> – 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

<u>שימו לב:</u>
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 שהורדתם מאתר של GNU. 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. הדפסה של התוכנית – (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. תיעוד נלווה – יש להגיש עותק על נייר,וגם יש צורך להגיש קובץ.
3. תרשים זרימה – יש להגיש דיאגרמת זרימת נתונים (DFD).
4. תקליטון DOS – 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.

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

```
declarations do end  float  for     if    int     ival    then
otherwise    print  program   read    rval start 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
```

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   →  program id  {  DECLARATIONS start STMTLIST end }

DECLARATIONS →  declarations DECLARLIST
               | ε

DECLARLIST →  DECLARLIST   TYPE : IDENTS
              | TYPE : IDENTS

IDENTS →  id , IDENTS
              | id;

TYPE →      int
            | float

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 real.

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

1. כאשר בביטוי מופיע משתנה או קבוע מטיפוס real, הטיפוס של הביטוי כולו הוא real.
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 real.

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

4. תוכנית לדוגמה:

```
program    Min  /* Finding minimum between two numbers */
{
declarations float:a,b;
  start
      read(a);
      read(b);
      if (a<b) then print(a);
           otherwise print(b);
end }
```

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

```
program  Min  /* Finding minimum between two numbers */
{
declarations float:a,b;
start
      read(a);
      read(b);
      if (a<b) then print(a);
           otherwise print(b);
end }
```

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.

<u>כדאי לשים לב לנקודות הבאות</u>:

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):

```
program  Example{
declarations    int:x;
start
read(x)  ;
if    x >= 0) then
                    print(x);
otherwise           print(0-x);
end}
```

דוגמת הרצה: פלט

קובץ LST:

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

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