תורת הקומפילציה

תרגיל בית 2 – בנית מנתח תחבירי

מתרגל אחראי: הילה לוי – hilalevi@campus.technion.ac.il

ההגשה בזוגות

עבור כל שאלה על התרגיל, יש לעין ראשית **בפיאצה** ובמידה שלא פורסמה אותה השאלה, ניתן להוסיף אותה ולקבל מענה, אין לשלוח מיילים בנושא התרגיל בית כדי שנוכל לענות על השאלות שלכם ביעילות.

תיקונים לתרגיל יסומנו בצהוב, חובתכם להתעדכן בהם באמצעות קובץ התרגיל.

התרגיל ייבדק בבדיקה אוטומטית. **הקפידו למלא אחר ההוראות במדויק.**

כללי

בתרגיל זה עליכן לממש ניתוח תחבירי לשפת FanC, הכוללת פעולות אריתמטיות, והמרות מובנות מ-byte (בית אחד) ל-int (4 בתים).

מנתח זה ישמש אתכם גם בתרגילים הבאים ולכן המלצתנו היא שתממשו ותגישו תרגיל זה על אף שהוא מגן.

מנתח לקסיקלי

יש לכתוב מנתח לקסיקלי המתאים להגדרות הבאות:

אסימון	תבנית
INT	int
BYTE	byte
B	b
BOOL	bool
AND	and
OR	or
NOT	not
TRUE	true
FALSE	false
RETURN	return
IF	if
ELSE	else
WHILE	while
BREAK	break
CONTINUE	continue
SC	;
LPAREN	(
RPAREN	)
LBRACE	{
RBRACE	}
ASSIGN	=
RELOP	== | != | < | > | <= | >=
BINOP	+ | - | * | /
ID	[a-zA-Z][a-zA-Z0-9]*
NUM	0 | [1-9][0-9]*
STRING	"([^\n\r\"\\]|\\[rnt"\\])+"

ניתן לשנות את שמות האסימונים או להוסיף אסימונים נוספים במידת הצורך, כל עוד המנתח הלקסיקלי מזהה את כל התבניות לעיל.
יש להתעלם מרווחים, ירידות שורה משני הסוגים (LF ,CR) וטאבים כך שלא תתקבל עליהם שגיאה לקסיקלית.
יש להתעלם מהערות שורה (הערות ++C) המיוצגות ע"י התבנית //[^\r\n]*[\r|\n|\r\n]?

הערה: המנתח הזה שונה במקצת מהמנתח הלקסיקלי של תרגיל בית 1, אבל ניתן להשתמש במנתח שתופס אסימונים נוספים או מתמודד עם escape-ים נוספים, כמו המנתח של תרגיל בית 1, כל עוד תפסתם את כל התבניות בטבלה.

תחביר

יש לכתוב מנתח תחבירי שיתאים לדקדוק הבא:

1. $Program \rightarrow Statements$
2. $Statements \rightarrow Statement$
3. $Statements \rightarrow Statements\ Statement$
4. $Statement \rightarrow LBRACE\ Statements\ RBRACE$
5. $Statement \rightarrow Type\ ID\ SC$
6. $Statement \rightarrow Type\ ID\ ASSIGN\ Exp\ SC$
7. $Statement \rightarrow ID\ ASSIGN\ Exp\ SC$
8. $Statement \rightarrow Call\ SC$
9. $Statement \rightarrow RETURN\ SC$
10. $Statement \rightarrow IF\ LPAREN\ Exp\ RPAREN\ Statement$
11. $Statement \rightarrow IF\ LPAREN\ Exp\ RPAREN\ Statement\ ELSE\ Statement$
12. $Statement \rightarrow WHILE\ LPAREN\ Exp\ RPAREN\ Statement$
13. $Statement \rightarrow BREAK\ SC$
14. $Statement \rightarrow CONTINUE\ SC$
15. $Call \rightarrow ID\ LPAREN\ Exp\ RPAREN$
16. $Type \rightarrow INT$
17. $Type \rightarrow BYTE$
18. $Type \rightarrow BOOL$
19. $Exp \rightarrow LPAREN\ Exp\ RPAREN$
20. $Exp \rightarrow Exp\ BINOP\ Exp$
21. $Exp \rightarrow ID$
22. $Exp \rightarrow Call$
23. $Exp \rightarrow NUM$
24. $Exp \rightarrow NUM\ B$
25. $Exp \rightarrow STRING$
26. $Exp \rightarrow TRUE$
27. $Exp \rightarrow FALSE$
28. $Exp \rightarrow NOT\ Exp$
29. $Exp \rightarrow Exp\ AND\ Exp$
30. $Exp \rightarrow Exp\ OR\ Exp$
31. $Exp \rightarrow Exp\ RELOP\ Exp$
32. $Exp \rightarrow LPAREN\ Type\ RPAREN\ Exp$

הערות:

1. הדקדוק כפי שמוצג כאן אינו חד משמעי ב-Bison. יש להפכו לחד משמעי תוך שימור השפה. בעיה לדוגמה שיש לפתור: http://en.wikipedia.org/wiki/Dangling_else.
   יש לפתור את בעיית ה-Dangling else ללא שינוי הדקדוק אלא באמצעות מתן עדיפות לכללים או אסוציאטיביות מתאימה לאסימונים.

2. יש להקפיד על מתן עדיפויות ואסוציאטיביות מתאימים לאופרטורים השונים. יש להשתמש בטבלת העדיפויות כאן: http://introcs.cs.princeton.edu/java/11precedence

3. אין צורך לבצע שינויים בדקדוק, פרט לשם הבדלה בין האופרטורים השונים.

קלט ופלט המנתח

קובץ ההרצה של המנתח יקבל את הקלט מ-stdin.

יש להיעזר בקובץ output.hpp המצורף לתרגיל על מנת לייצר פלט הניתן לבדיקה אוטומטית.

על המנתח להדפיס את כללי הגזירה על פי סדר ביצוע פעולות הreduce. בעת ביצוע reduce לכלל יש לקרוא לפונקציה printProductionRule(ruleno) עם מספר הכלל שנגזר.

לתרגיל מסופקים 2 דוגמאות עם קבצי קלט ופלט מתאימים. יש לבדוק שפורמט ההדפסה שלכם זהה לדוגמאות. הבדלי פורמט יגרמו לכישלון הבדיקות האוטומטיות.

טיפול בשגיאות

בקובץ הקלט יכולות להיות שגיאות לקסיקליות ותחביריות. **על המנתח לסיים את ריצתו מיד עם זיהוי שגיאה** (כלומר בנקודה העמוקה ביותר בעץ הגזירה שבה ניתן לזהותה). ניתן להניח כי הקלט מכיל שגיאה אחת לכל היותר.

על מנת לדווח על שגיאות יש להשתמש בפונקציות הנתונות בקובץ output.hpp:

שגיאה לקסיקלית	errorLex(lineno)
שגיאה תחבירית	errorSyn(lineno)

בכל השגיאות הנ"ל lineno הוא מס' השורה בה מופיעה השגיאה.

- במקרה של שגיאה הפלט של המנתח יהיה כל הכללים שנעשה להם reduce והשגיאה שהתגלתה (כפי שניתן לראות בדוגמה t2).

הוראות הגשה

שימו לב כי קובץ ה-Makefile מאפשר שימוש ב-STL. אין לשנות את ה-Makefile.

יש להגיש קובץ אחד בשם ID1-ID2.zip, עם מספרי ת"ז של שתי המגישות. על הקובץ להכיל:

- קובץ flex בשם scanner.lex המכיל את כללי הניתוח הלקסיקלי
- קובץ בשם parser.ypp המכיל את המנתח
- את כל הקבצים הנדרשים לבניית המנתח, כולל *.output שסופקו כחלק מהתרגיל.

בנוסף, יש להקפיד שהקובץ **לא יכיל** את:

- קובץ ההרצה
- קבצי הפלט של flex ו-bison
- קובץ Makefile שסופק כחלק מהתרגיל

יש לוודא כי בביצוע unzip לא נוצרת תיקיה נפרדת. על המנתח להיבנות על השרת csComp ללא שגיאות באמצעות קובץ Makefile שסופק עם התרגיל. באתר הקורס מופיע קובץ zip המכיל קבצי בדיקה לדוגמה. יש לוודא כי פורמט הפלט זהה לפורמט הפלט של הדוגמאות הנתונות. כלומר, ביצוע הפקודות הבאות:

```
unzip id1-id2.zip
cp path-to/Makefile .
cp path-to/ hw2-tests.zip .

unzip hw2-tests.zip
make
./hw2 < t1.in 2>&1 > t1.res
diff t1.res path-to/t1.out
```

ייצור את קובץ ההרצה בתיקיה הנוכחית ללא שגיאות קומפילציה, יריץ אותו, ו-diff יחזיר 0.

הגשות שלא יעמדו בדרישות לעיל יקבלו ציון 0 ללא אפשרות לבדיקה חוזרת.

בדקו היטב שההגשה שלכן עומדת בדרישות הבסיסיות הללו לפני ההגשה עצמה.

שימו לב כי באתר מופיע script לבדיקה עצמית לפני ההגשה בשם selfcheck. תוכלו להשתמש בו על מנת לוודא כי ההגשה שלכם תקינה.

בתרגיל זה (כמו בתרגילים אחרים בקורס) **ייבדקו העתקות**. אנא כתבו את הקוד שלכם בעצמכם.

בהצלחה! ☺