תורת הקומפילציה

תרגיל בית 2 – ניתוח תחבירי top down

מתרגל אחראי לתרגיל: יואב צוריאל

הגשה בזוגות.

יש להפנות שאלות על התרגיל במייל ל-yoavzuriel@cs בלבד.

לתרגיל ייפתח דף FAQ באתר הקורס. כל הבהרה שתופיע בו עד יומיים ממועד ההגשה תהווה הוראה מחייבת.

<u>הנחיות כלליות</u>

- בתרגיל זה תממשו מנתח תחבירי עבור דקדוק המשתמש בחלק מאסימונים אשר זיהיתם בתרגיל הקודם.
- התרגיל ייבדק אוטומטית. **הקפידו למלא אחר ההוראות במדויק.** הבדיקה תתבצע על csl3.
- קראו את כל התרגיל עד סופו בטרם תחילת העבודה על פתרון.

להלן דקדוק במחלקה LL(1) המגדיר (באופן חלקי) תוכניות חוקיות בCSS, בהתבסס על האסימונים מתרגיל 1:

1. $S \rightarrow LImports\ LRules$
2. $LImports \rightarrow Import\ LImports$
3. $LImports \rightarrow \varepsilon$
4. $Import \rightarrow IMPORT\ STRING\ SEMICOLON$
5. $LRules \rightarrow Rule\ LRules$
6. $LRules \rightarrow \varepsilon$
7. $Rule \rightarrow Selectors\ LBRACE\ LDeclarations\ RBRACE$
8. $LDeclarations \rightarrow Declaration\ SEMICOLON\ MoreDeclarations$
9. $MoreDeclarations \rightarrow LDeclarations$
10. $MoreDeclarations \rightarrow \varepsilon$
11. $Selectors \rightarrow SimpleSelector\ MoreSelectors$
12. $MoreSelectors \rightarrow Selectors$
13. $MoreSelectors \rightarrow \varepsilon$
14. $SimpleSelector \rightarrow Astrisk\ SelectorModifier$
15. $Astrisk \rightarrow ASTERISK$
16. $Astrisk \rightarrow \varepsilon$
17. $SelectorModifier \rightarrow DOT\ NAME$
18. $SelectorModifier \rightarrow COLON\ NAME$
19. $SelectorModifier \rightarrow LBRACE\ NAME\ EQUAL\ Term\ RBRACE$
20. $SelectorModifier \rightarrow HASHID$
21. $SelectorModifier \rightarrow NAME$
22. $Declaration \rightarrow NAME\ COLON\ LTerm\ Important$
23. $Important \rightarrow IMPORTANT$
24. $Important \rightarrow \varepsilon$
25. $LTerm \rightarrow Term\ MoreTerms$
26. $MoreTerms \rightarrow LTerms$
27. $MoreTerms \rightarrow \varepsilon$
28. $Term \rightarrow NUMBER$
29. $Term \rightarrow UNIT$
30. $Term \rightarrow STRING$
31. $Term \rightarrow NAME$

32. $Term \rightarrow HASHID$

כך ש-S הוא הנונטרמינל ההתחלתי בדקדוק.

מבנה התרגיל

עם התרגיל נתונים לכם מספר קבצים:

- הקובץ tokens.h מכיל אנומרציה (enum) של שמות האסימונים מהתרגיל הקודם.
- הקבצים grammar.h ו-grammar.cpp מגדירים ומאתחלים את הדקדוק המופיע בתחילת התרגיל, ומכילים פונקציות הדפסה שישמשו אתכם כדי לייצר את הפלט הנדרש.
- הקובץ hw2.h מכיל את הצהרות הפונקציות שעליכם לממש כדי לענות על הסעיפים השונים בתרגיל.

אין צורך לממש פונקציית main. נתונה לכם פונקציית main בקובץ main.cpp.

קראו **היטב** את התיעוד של הפונקציות בקבצים השונים.

שלב מקדים – הכנת המנתח הלקסיקלי

עדכנו את הפתרון שלכם לתרגיל 1 כך שכל ההדפסות שביצעתם בזיהוי אסימונים יוחלפו בהחזרת ערך האנומרציה של האסימון המתאים (לשם כך תצטרכו לבצע include לקובץ tokens.h). שנו את שם קובץ הlex ל-lexer.lex.

לאסימון של COMMENT לא מוגדר ערך האנומרציה, וזו אינה טעות. כיוון שהערות הן חסרות משמעות בדקדוק, אין צורך להעביר אותן לניתוח התחבירי. (יש גם מספר אסימונים נוספים להם אין ערך בtokens.h, אבל אין מאחורי זה משמעות דידקטית עמוקה).

בנוסף, כפי שניתן לראות בקובץ tokens.h, הוגדר ערך אסימון נוסף שמסמן את סוף הקלט, EF. עליכם להוסיף לקובץ הניתוח הלקסיקלי את הכלל הבא בסופו (לפני כלל ברירת המחדל של השגיאות הלקסיקליות)

```
<<EOF>>                    {return EF;}
```

כדי שתוכלו לדעת מתי נגמר הקלט בצורה פשוטה, וכך לעצור את פעילות המנתח התחבירי.

סעיף א' – זיהוי משתנים אפיסים

המשתנה grammar (שמוגדר בקובץ grammar.h ומאותחל בקובץ grammar.cpp) מכיל את הדקדוק שמופיע בתחילת התרגיל. השתמשו בו כדי לזהות איזה משתנים (המוגדרים באנומרציה nonterminal בקובץ grammar.h) הם אפיסים ואיזה לא.

עליכם לממש אלגוריתם fixed-point לזיהוי המשתנים האפיסים בדקדוק. האלגוריתם ימומש בגוף הפונקציה compute_nullable המוצהרת בקובץ hw2.h.

בתום חישוב המשתנים האפיסים יש להדפיסם באמצעות הפונקציה print_nullable.

על מנת לבדוק סעיף זה יש להריץ את קובץ ההרצה עם הארגומנט -nullable.

סעיף ב' – חישוב פונקציית first

בסעיף זה עליכם לממש אלגוריתם fixed-point לחישוב פונקציית first לכל אחד מהמשתנים בדקדוק על פי האלגוריתם המופיע בשקף 18 בתרגול 2. האלגוריתם ימומש בגוף הפונקציה compute_first המוצהרת בקובץ hw2.h.

בתום חישוב פונקציית first יש להדפיס את ה-first של כל המשתנים באמצעות הפונקציה print_first.

על מנת לבדוק סעיף זה יש להריץ את קובץ ההרצה עם הארגומנט -first.

סעיף ג' – חישוב פונקציית follow

בסעיף זה עליכם לממש אלגוריתם fixed-point לחישוב פונקציית follow לכל אחד מהמשתנים בדקדוק על פי האלגוריתם המופיע בשקף 25 בתרגול 2. האלגוריתם ימומש בגוף הפונקציה compute_follow המוצהרת בקובץ hw2.h.

בתום חישוב פונקציית follow יש להדפיס את ה-follow של כל המשתנים באמצעות הפונקציה print_follow.

על מנת לבדוק סעיף זה יש להריץ את קובץ ההרצה עם הארגומנט follow-.

סעיף ד' – חישוב פונקציית select

בסעיף זה עליכם לממש אלגוריתם לחישוב פונקציית select לכל אחד מכללי הדקדוק על פי ההגדרה המופיעה בשקף 17 בתרגול 2. האלגוריתם ימומש בגוף הפונקציה compute_select המוצהרת בקובץ hw2.h.

בתום חישוב פונקציית select יש להדפיס את ה-select של כל הכללים באמצעות הפונקציה print_select.

על מנת לבדוק סעיף זה יש להריץ את קובץ ההרצה עם הארגומנט select-.

סעיף ה' – מימוש מנתח LL(1)

בסעיף זה תממשו מנתח LL(1) לדקדוק הנתון בתחילת התרגיל, בהסתמך על הסעיפים הקודמים ועל פי האלגוריתם המופיע בשקף 37 בתרגול 2.

את הפתרון לסעיף זה יש לממש בגוף הפונקציה parser המוגדרת בקובץ hw2.h.

על מנת לבדוק סעיף זה יש להריץ את קובץ ההרצה ללא ארגומנטים.

<u>אופן פעולת המנתח</u>

המנתח יבנה את רצף כללי הגזירה הנחוץ לבניית המילה המופיעה בקובץ הקלט.
המנתח יקבל כקלט מ-stdin רצף תווים העונה להגדרות מתרגיל 1 של האסימונים המופיעים בדקדוק.
השתמשו ב-()yylex על מנת לקרוא את רצף האסימונים (הקריאה הראשונה ל-yylex תחזיר את האסימון הראשון בקלט).

במהלך הניתוח, בכל הפעלת כלל גזירה (כחלק מפעולת predict) יש להדפיס את מספר כלל הגזירה בו השתמשתם (בהתאם לאנומרציה המופיע בתחילת התרגיל). למשל אם הופעל כלל הגזירה $LImports \rightarrow LImports\ Import$, מספרו 2 ולכן יש להדפיס:

```
2\n
```

במידה והניתוח מסתיים בהצלחה יש להדפיס את ההודעה:

```
Success\n
```

במידה ומתקבלת שגיאת ניתוח יש להדפיס את ההודעה:

```
Syntax error\n
```

ולסיים את ריצת המנתח **מיד**.

נתונות לכם שתי דוגמאות קלט/פלט עבור סעיף זה.

<u>המלצות מימוש</u>

- השתמשו ב-*std::vector<int>* למימוש המחסנית Q.
- השתמשו ב-*std::map<nonterminal, std::map<tokens, int>>* למימוש הטבלה M.

הערות לגבי הדפסות

פונקציות ההדפסה עבור סעיפים א'- ד' מקבלות *std::vector* המכיל את הערכים הצפויים.

עבור סעיפים א'-ג' ערכי ה-vector יהיו הערכים עבור איברי האנומרציה nonterminal, לפי הסדר. כלומר במקום ה-0 הערך עבור S, ובאופן כללי הערך עבור נונטרמינל i יופיע במקום ה-i.

עבור סעיף ד' ערכי ה-vector יהיו הערכים עבור כללי הדקדוק, לפי הסדר שהוגדר בתרגיל. כלומר במקום ה-0 הערך עבור הכלל הראשון ($S \rightarrow LImports\ LRules$), ובאופן כללי הערך עבור הכלל i יופיע במקום ה-(i-1).

הוראות הגשה

אין לשנות אף אחד מהקבצים המסופקים עם התרגיל למעט את hw2.h.

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

- קובץ lexer.lex
- קובץ hw2.h
- את כל הקבצים הנדרשים לבניית המנתח, כולל קבצי cpp/h שסופקו כחלק מהתרגיל.

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

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

יש לוודא כי בביצוע unzip לא נוצרת תיקיה נפרדת – כלומר כי קבצי הפתרון נמצאים בתיקיה בה בוצע unzip. על המנתח להיבנות על השרת csl3 ללא שגיאות באמצעות קובץ Makefile שסופק עם התרגיל. יש לוודא כי פורמט הפלט זהה לפורמט הפלט של הדוגמאות הנתונות. כלומר, ביצוע הפקודות הבאות:

```
unzip id1-id2.zip
cp path-to/Makefile .
make
./hw2 < path-to/t1.in >& t1.res
diff t1.res path-to/t1.out
```

ייצור את קובץ ההרצה בתיקיה הנוכחית ללא שגיאות קומפילציה, יריץ אותם, והרצת diff תחזיר 0. שימו לב כי במידה והמנתח שלכם לא עובר את כל קבצי הבדיקה שסופקו מראש, לא תתאפשר הגשה חוזרת של התרגיל.

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

בהצלחה!