گزارش تمرین سوم کامپایلر

علیرضا اسلامی خواه 99521064

برای سوال اول یک گرامر دلخواه طراحی کردیم که بدین صورت قابل مشاهده است :

```
grammar Q1;
//lexer
IF: 'if';
WHILE: 'while';
INT: 'int';
BOOL: 'boolean';
ID: LETTER(LETTER|DIGIT)*;
INTTYPE: ('-'?) DIGIT+;
BOOLTYPE: 'True' | 'False';
WS : [ \n\r\t] -> channel(HIDDEN);
RELOP: '<' | '==' | '>';
EQ: '==';
COMMENT : '#' ~[\n\r]+ -> channel(HIDDEN) ;
fragment LETTER: [A-Za-z_];
fragment DIGIT: [0-9];



//parser
start : (statement | compoundst)* EOF;

ifst returns [value_attr = str(), type_attr = str()]:
    IF '(' condition ')' '{' statement+ '}';
whilest returns [value_attr = str(), type_attr = str()]:
    WHILE '(' condition ')' '{' statement* '}';

statement returns [value_attr = str(), type_attr = str()]:
    ifst | assignment | whilest | compoundst  ;

compoundst returns [value_attr = str(), type_attr = str()]:
    '{' (statement*) '}' ;

assignment returns [value_attr = str(), type_attr = str()]:
    INT? ID '=' INTTYPE ';'
    | BOOL? ID '=' BOOLTYPE ';'
    ;

condition returns [value_attr = str(), type_attr = str()]:
    BOOLTYPE
    | ID RELOP ID
    | ID RELOP INTTYPE
```

```
    | ID EQ BOOLTYPE
    ;
```

سعی کردیم از گرامر های تمرین های قبلی نیز استفاده کنیم.

سوال دوم :

در اینجا این تکه کد را پیاده سازی کردیم ، بدین صورت که تمام lexer ها را به صورت توکن شده در اختیار داریم و با حلقه ای که میزنیم به کامنت میرسیم و آن را اصلاح میکنیم.

```
"Q2"
    lastname = "Eslamikhah"
    studentid = "99521064"
    token = lexer.nextToken()
    refactoredStr = ''
    while token.type != Token.EOF :
        if(token.type == lexer.COMMENT):
            refactoredStr += '#' + lastname + ' ' +token.text[1:] +' '
+studentid + '#'
            print(token.text+ '\n')

        else :
            refactoredStr += token.text
        token = lexer.nextToken()

    print(refactoredStr+'\n')
```

برای سوال سوم بدین صورت طراحی کردیم که بعد از هر ورود و خروج یک عدد به استک اضافه کند و در آخر بالاترین عدد را ثبت کند. مثلا وارد if که میشود عمق یکی اضافه میشود و وقتی از همان شرط خارج میشود عمق یکی کم میشود.

اینگونه هر statement ای که درون if وارد میشود یکی به عمق اضافه میکند.

کد های این بخش در custum listener قابل مشاهده است.

```
    def enterIfst(self, ctx:Q1Parser.IfstContext):
        print("entered")
        self.depth = self.depth + 1
        if self.depth > self.max_depth:
            self.max_depth = self.depth

    def exitIfst(self, ctx: Q1Parser.IfstContext):
        self.depth = self.depth - 1

    def enterWhilest(self, ctx:Q1Parser.WhilestContext):
```

```
        self.depth = self.depth + 1
        if self.depth > self.max_depth:
            self.max_depth = self.depth

    def exitWhilest(self, ctx:Q1Parser.WhilestContext):
        self.depth = self.depth - 1

    def exitStart(self, ctx:Q1Parser.StartContext):
        print(self.max_depth)
```