Bai 1:

grammar BKOOL;

@lexer*::header* {

from lexererr import \*

}

options {

    language = Python3;

}

*// Use ANTLR to write regular expressions describing Pascal strings are made up of a sequence of characters between single quotes: 'string'. The*

*// single quote itself can appear as two single quotes back to back in a string: 'isn''t'.*

program: (expr NEWLINE)\* EOF;

expr: expr ('\*' | '/') expr | expr ('+' | '-') expr | '(' expr ')';

NEWLINE: [\r\n]+;

BKNetID: NAME '.' SURNAME SUFFIX\_PART;

fragment NAME: [a-z]+;

fragment SURNAME: [a-z]+;

fragment SUFFIX\_PART: ( ([a-z] | [0-9] | '.' | '\_')+ [a-z0-9\_]) | ([a-z] | [0-9] | '\_');

*// REAL: ([0-9]+ '.' [0-9]+ ([eE] [+-]? [0-9]+)?) | ([0-9]+ [eE] [+-]? [0-9]+);*

*// example string: 'isn''t' STRING: '\'' ( ~('\'' | '\r' | '\n') | '\'\'')\* '\'';*

WS: [ \t\r\n]+ -> skip; *// skip spaces, tabs, newlines*

ERROR\_CHAR: . {raise ErrorToken(self.text)};

UNCLOSE\_STRING: .;

ILLEGAL\_ESCAPE: .;

Bai 2:

grammar BKOOL;

@lexer*::header* {

from lexererr import \*

}

options {

    language = Python3;

}

*// Use ANTLR to write regular expressions describing Pascal strings are made up of a sequence of characters between single quotes: 'string'. The*

*// single quote itself can appear as two single quotes back to back in a string: 'isn''t'.*

program: (expr NEWLINE)\* EOF;

expr: expr ('\*' | '/') expr | expr ('+' | '-') expr | '(' expr ')';

NEWLINE: [\r\n]+;

IPv4: OCTET '.' OCTET '.' OCTET '.' OCTET;

fragment OCTET: '0' | ([1-9][0-9]?) | ('1' [0-9][0-9]) | ('2' [0-4][0-9]) | ('25' [0-5]);

WS: [ \t\r\n]+ -> skip; *// skip spaces, tabs, newlines*

ERROR\_CHAR: . {raise ErrorToken(self.text)};

UNCLOSE\_STRING: .;

ILLEGAL\_ESCAPE: .;

Bai 3:

grammar BKOOL;

@lexer*::header* {

from lexererr import \*

}

options {

    language = Python3;

}

*// Use ANTLR to write regular expressions describing Pascal strings are made up of a sequence of characters between single quotes: 'string'. The*

*// single quote itself can appear as two single quotes back to back in a string: 'isn''t'.*

program: (expr NEWLINE)\* EOF;

expr: expr ('\*' | '/') expr | expr ('+' | '-') expr | '(' expr ')';

NEWLINE: [\r\n]+;

*// IPv4: OCTET '.' OCTET '.' OCTET '.' OCTET;*

*// fragment OCTET: '0' | ([1-9][0-9]?) | ('1' [0-9][0-9]) | ('2' [0-4][0-9]) | ('25' [0-5]);*

PHP\_INT: '0' | [1-9]([0-9] | '\_' [0-9])\* {self.text = self.text.replace('\_', '')};

WS: [ \t\r\n]+ -> skip; *// skip spaces, tabs, newlines*

ERROR\_CHAR: . {raise ErrorToken(self.text)};

UNCLOSE\_STRING: .;

ILLEGAL\_ESCAPE: .;

Bai 4:

grammar BKOOL;

@lexer*::header* {

from lexererr import \*

}

options {

    language = Python3;

}

*// Use ANTLR to write regular expressions describing Pascal strings are made up of a sequence of characters between single quotes: 'string'. The*

*// single quote itself can appear as two single quotes back to back in a string: 'isn''t'.*

program: (expr NEWLINE)\* EOF;

expr: expr ('\*' | '/') expr | expr ('+' | '-') expr | '(' expr ')';

NEWLINE: [\r\n]+;

*// IPv4: OCTET '.' OCTET '.' OCTET '.' OCTET;*

*// fragment OCTET: '0' | ([1-9][0-9]?) | ('1' [0-9][0-9]) | ('2' [0-4][0-9]) | ('25' [0-5]);*

*// PHP\_INT: '0' | [1-9]([0-9] | '\_' [0-9])\* {self.text = self.text.replace('\_', '')};*

SHEXA: [0-9] ([0-9a-fA-F])\* ([0248ACEace] | [0369BDFbdf][0-9a-fA-F]);

WS: [ \t\r\n]+ -> skip; *// skip spaces, tabs, newlines*

ERROR\_CHAR: . {raise ErrorToken(self.text)};

UNCLOSE\_STRING: .;

ILLEGAL\_ESCAPE: .;