Assignment 2

Program 1 -

```
program --> block,[.].
block --> [begin], declaration, [;], command, [end].

declaration --> single_declaration, [;], declaration.
declaration --> single_declaration.
single_declaration --> [const], identifier, [=], number.
single_declaration --> [var], identifier.

command --> single_command, [;], command.
command --> single_command.
single_command --> identifier, [:=], expression.
single_command --> [if], boolean_exp, [then], command, [else], command, [endif].
single_command --> [while], boolean_exp, [do], command, [endwhile].
single_command --> block.

boolean_exp --> [true].
boolean_exp --> [false].
boolean_exp --> expression, [=], expression.
boolean_exp --> [not], boolean_exp.

expression --> term1, expression_not.
expression_not --> [+], term1, expression_not | [].
term1 --> term2, term1_not.
term1_not --> [-], term2, term1_not | [].
term2 --> term3, term2_not.
term2_not --> [*], term3, term2_not | [].
term3 --> term4, term3_not.
term3_not --> [/], term4, term3_not | [].
term4 --> identifier.
term4 --> number.

identifier --> [x] | [y] | [z] | [u] | [v].
number --> [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9].
```

```
?- program([begin,const, x, =, 8,;,var, y,;,var, z,;,z, :=, 0,;,if, x, =, y, +, 2, then,
z,:=,8,else, z, :=, 3,endif,;, while, not, x, =,z, do,z, :=, z, +, 2, endwhile,end,.], []).
true ;
```

Program 2

```
program(program(X,.)) --> block(X),[.].
block(block(begin,X,;,Y,end)) --> [begin], declaration(X), [;], command(Y), [end].

declaration(declaration(X,;,Y)) --> single_declaration(X), [;], declaration(Y).
declaration(declaration(X)) --> single_declaration(X).
single_declaration(single_declaration(const,X,=,Y)) --> [const], identifier(X), [=],
number(Y).
single_declaration(single_declaration(var,X)) --> [var], identifier(X).

command(command(X,;,Y)) --> single_command(X), [;], command(Y).
command(command(X)) --> single_command(X).
single_command(single_command(X,:=,Y)) --> identifier(X), [:=], expression(Y).
single_command(single_command(if,X,then,Y,else,Z,endif)) --> [if], boolean_exp(X), [then],
command(Y), [else], command(Z), [endif].
single_command(single_command(while,X,do,Y,endwhile)) --> [while], boolean_exp(X), [do],
command(Y), [endwhile].
single_command(single_command(X)) --> block(X).

boolean_exp(boolean_exp(true)) --> [true].
boolean_exp(boolean_exp(false)) --> [false].
boolean_exp(boolean_exp(X,=,Y)) --> expression(X), [=], expression(Y).
boolean_exp(boolean_exp(not,X)) --> [not], boolean_exp(X).

expression(expression(X,Y)) --> term1(X), expression_not(Y).
expression_not(expression_not(+,X,Y)) --> [+], term1(X), expression_not(Y).
expression_not(expression_not(e)) --> [].
term1(term1(X,Y)) --> term2(X), term1_not(Y).
term1_not(term1_not(-,X,Y)) --> [-], term2(X), term1_not(Y).
term1_not(term1_not(e)) --> [].
term2(term2(X,Y)) --> term3(X), term2_not(Y).
term2_not(term2_not(*,X,Y)) --> [*], term3(X), term2_not(Y).
term2_not(term2_not(e)) --> [].
term3(term3(X,Y)) --> term4(X), term3_not(Y).
term3_not(term3_not(/,X,Y)) --> [/], term4(X), term3_not(Y).
term3_not(term3_not(e)) --> [].
term4(term4(X)) --> identifier(X).
term4(term4(X)) --> number(X).

identifier(identifier(x)) --> [x].
identifier(identifier(y)) --> [y].
identifier(identifier(z)) --> [z].
identifier(identifier(u)) --> [u].
identifier(identifier(v)) --> [v].

number(number(0)) --> [0].
number(number(1)) --> [1].
```

```
number(number(2)) --> [2].
number(number(3)) --> [3].
number(number(4)) --> [4].
number(number(5)) --> [5].
number(number(6)) --> [6].
number(number(7)) --> [7].
number(number(8)) --> [8].
number(number(9)) --> [9].
```

```
[debug]  ?- program(L,[begin,const, x, =, 8,;,var, y,;,var, z,;,z, :=, 0,;,if, x, =, y, +, 2,
then, z,:=,8,else, z, :=, 3,endif,;, while, not, x, =,z, do,z, :=, z, +, 2, endwhile,end,.],
[]).
L = program(block(begin, declaration(single_declaration(const, identifier(x), =, number(8)),
;, declaration(single_declaration(var, identifier(y)), ;, declaration(single_declaration(var,
identifier(z))))), ;, command(single_command(identifier(z), :=,
expression(term1(term2(term3(term4(number(0)), term3_not(e)), term2_not(e)), term1_not(e)),
expression_not(e))), ;, command(single_command(if,
boolean_exp(expression(term1(term2(term3(term4(identifier(x)), term3_not(e)), term2_not(e)),
term1_not(e)), expression_not(e)), =, expression(term1(term2(term3(term4(identifier(y)),
term3_not(e)), term2_not(e)), term1_not(e)), expression_not(+,
term1(term2(term3(term4(number(2)), term3_not(e)), term2_not(e)), term1_not(e)),
expression_not(e)))), then, command(single_command(identifier(z), :=,
expression(term1(term2(term3(term4(number(8)), term3_not(e)), term2_not(e)), term1_not(e)),
expression_not(e)))), else, command(single_command(identifier(z), :=,
expression(term1(term2(term3(term4(number(3)), term3_not(e)), term2_not(e)), term1_not(e)),
expression_not(e)))), endif), ;, command(single_command(while, boolean_exp(not,
boolean_exp(expression(term1(term2(term3(term4(identifier(x)), term3_not(e)), term2_not(e)),
term1_not(e)), expression_not(e)), =, expression(term1(term2(term3(term4(identifier(z)),
term3_not(e)), term2_not(e)), term1_not(e)), expression_not(e)))), do,
command(single_command(identifier(z), :=, expression(term1(term2(term3(term4(identifier(z)),
term3_not(e)), term2_not(e)), term1_not(e)), expression_not(+,
term1(term2(term3(term4(number(2)), term3_not(e)), term2_not(e)), term1_not(e)),
expression_not(e))))), endwhile)))), end), '.')
```