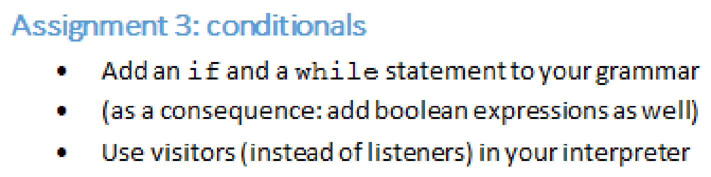
Week 3- AUT Assignment

Group 3

Table of contents

Assignment 3

Problem description



Assignment 3: conditionals

- Add an `if` and a `while` statement to your grammar
- (as a consequence: add boolean expressions as well)
- Use visitors (instead of listeners) in your interpreter

Implementation

For this assignment, we had to use visitors instead of listeners so that we can add statements to our grammar. First we had to reimplement the expressions we added while using listeners. Visitors are useful when it comes to creating statements because they only visit the specified branch of the Parse Tree and not the whole Parse Tree as the listener does. The grammar for the “if_statement” and “while_statement” are straight forward.
Regarding the implementation with the visitors, given the fact that both of this statements are executed if and only if their condition is true, we had to make sure that the expressions within the context of the condition_block is true (by converting the expression to boolean)

```
grammar Example2;

start2    : statement* EOF;

statement:   expression      #otherExpr
        |   value           #assign
        |   print_func      #print
        |   while_statement #while_stat
        |   if_statement    #if_stat
        ;

print_func:   Print op=(INT| BOOLEAN |ID |STRING)  #printVar
         |    Print mathExpression                 #printExpr
         ;
```

```
value : int_variable
     | bool_variable
     | string_variable
     ;

if_statement
       :   IF condition_block (ELSE code_block)?;
while_statement: WHILE condition_block;

condition_block: PARANL expression PARANR code_block;

code_block:   OPEN_CURLY_BRACKET statement* CLOSE_CURLY_BRACKET
         |   statement
         ;

string_variable : StringType ID (IS_EQUAL STRING)? # stringAssign
               | ID IS_EQUAL STRING  # stringAssignValue
               ;

bool_variable : BoolType ID (IS_EQUAL BOOLEAN)?  # boolAssign
             | ID IS_EQUAL BOOLEAN  # boolAssignValue
             ;

int_variable :  IntType ID (IS_EQUAL mathExpression)?        # intAssign
            |  ID IS_EQUAL mathExpression                   #intAssignValue
            |  ID (ADD_INCREMENT| SUB_INCREMENT | INCREMENT | DECREMENT)
mathExpression?  # incrementAndDecrementInt
            ;

expression:   mathExpression #MathExp
         |   BOOLEAN        #ValueBoolean
         |   STRING         #ValueString
         |   expression (GREATER_OR_EQUAL | SMALLER_OR_EQUAL |
GREATHER_THAN | SMALLER_THAN | EQUAL | NOT_EQUAL) expression   #
ComparisonExpression
         |   expression AND expression                            #
AndExpression
         |   expression OR expression                         #
OrExpression
         ;

mathExpression:  mathExpression op=MUL mathExpression #  Mul
```

```
            |   mathExpression op=DIV mathExpression #  Div
            |   mathExpression op=ADD mathExpression #  Add
            |   mathExpression op=SUB mathExpression #  Sub
            |   mathExpression op=POW mathExpression #  Pow
            |   mathExpression op=FACT  #  Fact
            |   PARANL mathExpression PARANR  # parens
            |   INT #ValueNumber
            |   ID    #ValueVariable
            ;


// tokens
IS_EQUAL: '=';
MUL:    '*';
DIV:    '/';
ADD:    '+';
SUB:    '-';
POW:    '^';
FACT:   '!';
PARANL: '(';
PARANR: ')';
DOT : '.';

ADD_INCREMENT : '+=';
SUB_INCREMENT : '-=';
INCREMENT : '++';
DECREMENT : '--';

OPEN_CURLY_BRACKET : '{';
CLOSE_CURLY_BRACKET : '}';

AND : 'and';
OR : 'or';

GREATER_OR_EQUAL : '>=';
SMALLER_OR_EQUAL : '<=';
GREATHER_THAN : '>';
SMALLER_THAN : '<';
EQUAL : '==';
NOT_EQUAL : '!=';


IntType: 'int';
```

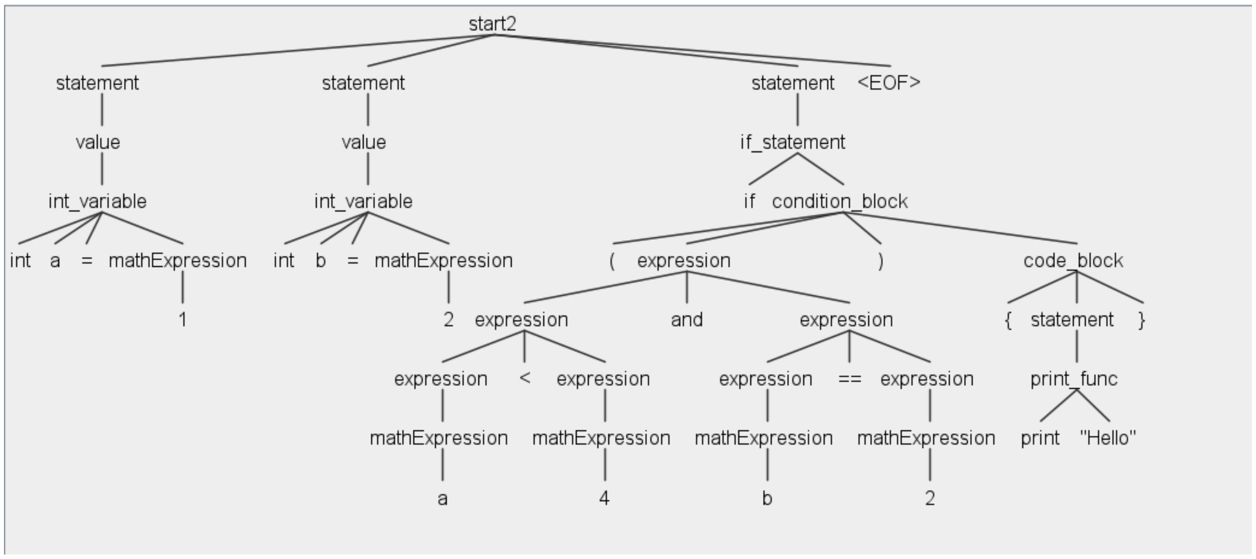
```
BoolType: 'bool';
StringType: 'string';

WHILE : 'while';
COMMA: ',';
SEMICOLON: ';';

IF : 'if';
ELSE : 'else';
FOR : 'for';

Print: 'print';

INT     : SUB?[0-9]+(DOT[0-9]+)? ;
BOOLEAN: 'True'|'False';
ID: [_A-Za-z][A-Za-z_!0-9.]* ;
STRING : '"' ~('\r' | '\n' | '"')* '"';

COMMENT : '//' .+? ('\n'|EOF) -> skip;
WS  : [ \t\r\n]+ -> skip;
```

Output

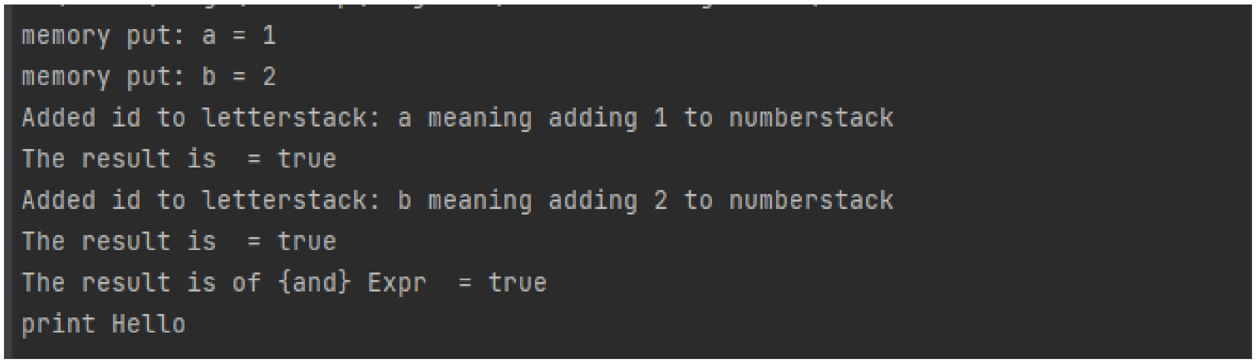
Below some test cases can be found to check the correctness of our grammar and implementations.

1. ***int a = 1***
   int b = 2
   if (a < 4 and b == 2){
   print "Hello"
   }

Parse Tree



Terminal

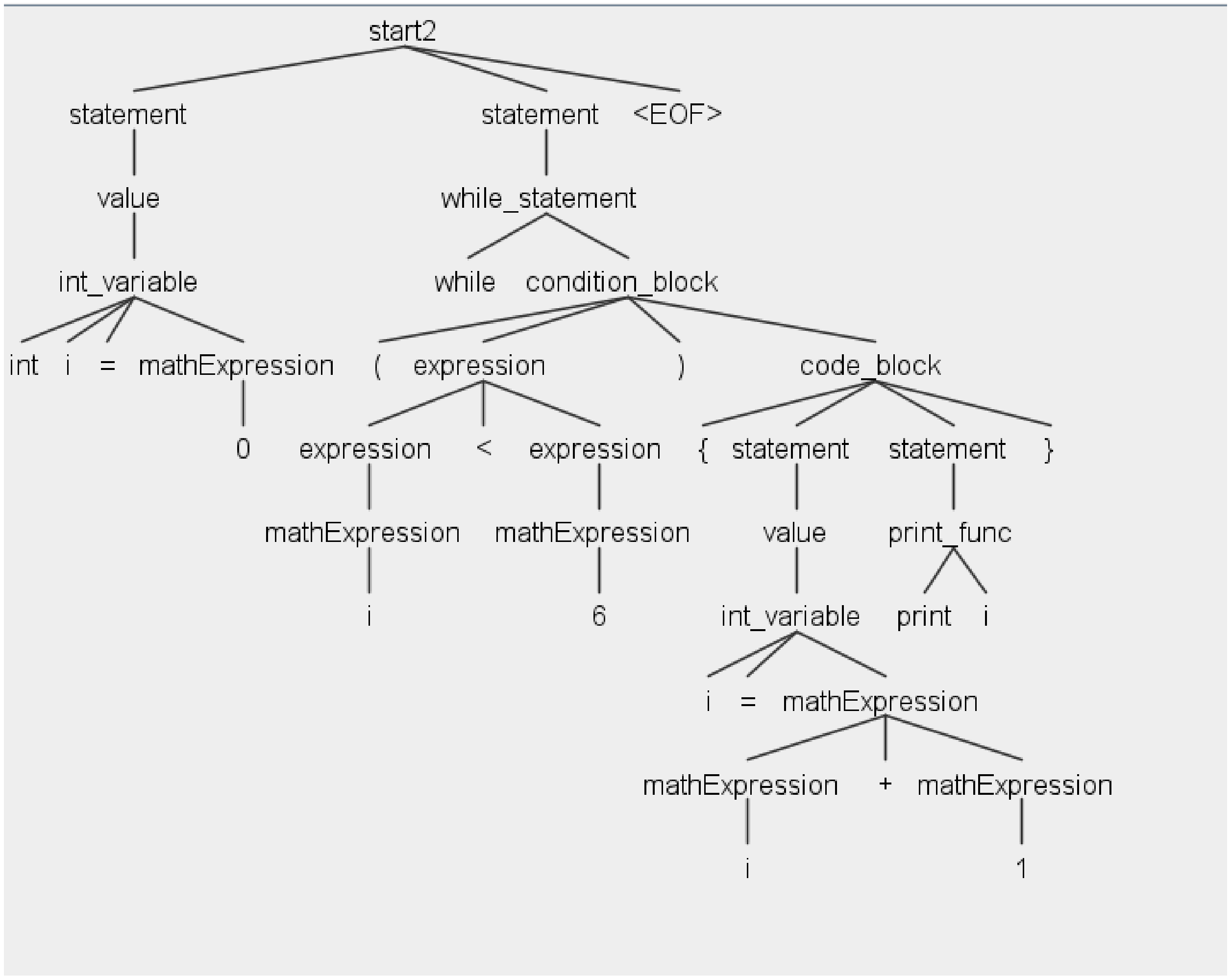


```
memory put: a = 1
memory put: b = 2
Added id to letterstack: a meaning adding 1 to numberstack
The result is  = true
Added id to letterstack: b meaning adding 2 to numberstack
The result is  = true
The result is of {and} Expr  = true
print Hello
```

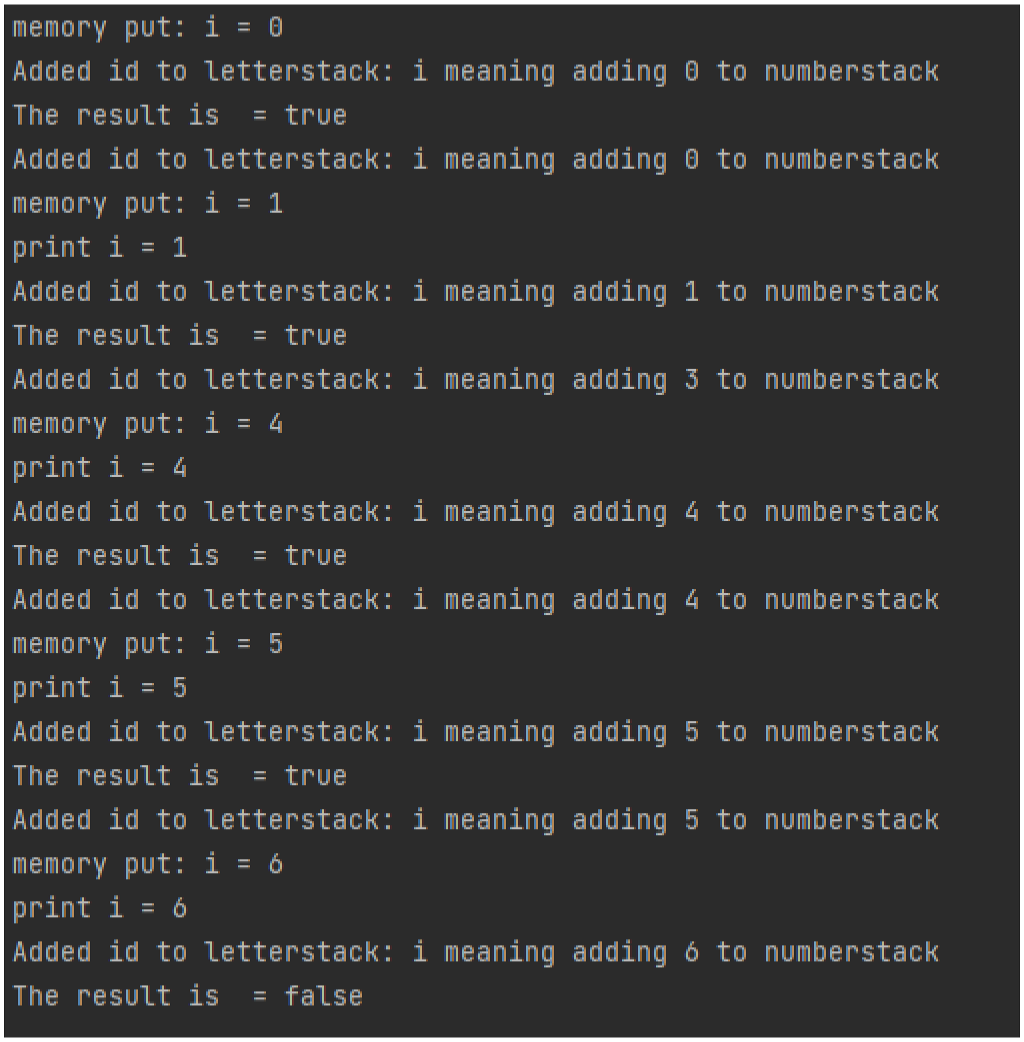
2. ***int i = 0***

```
while(i < 6){
      i = i + 1
      print i
}
```

Parse Tree



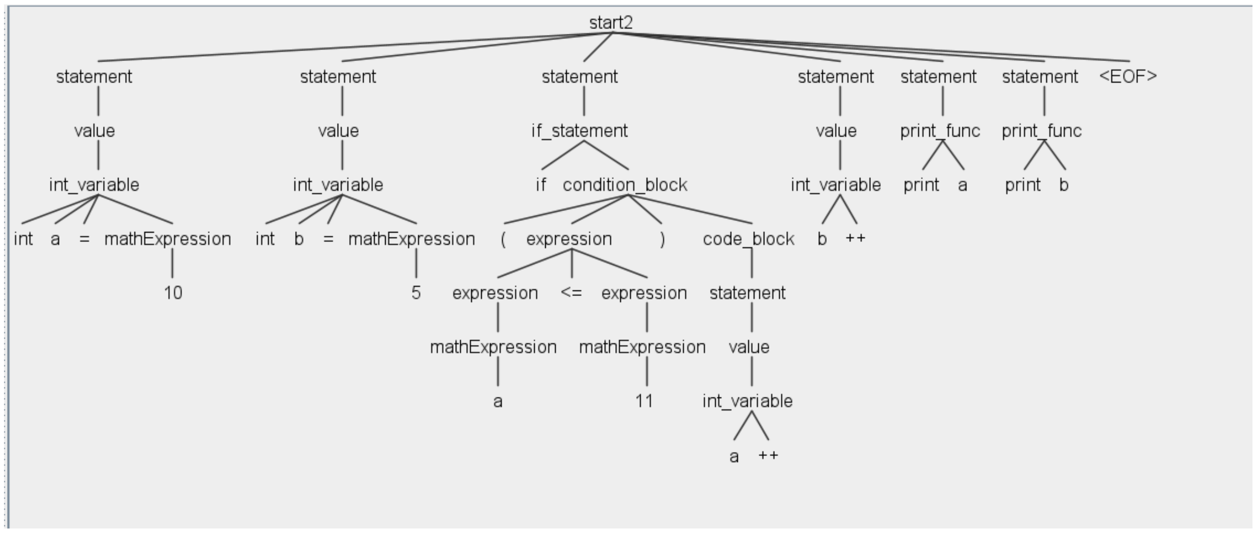
Terminal



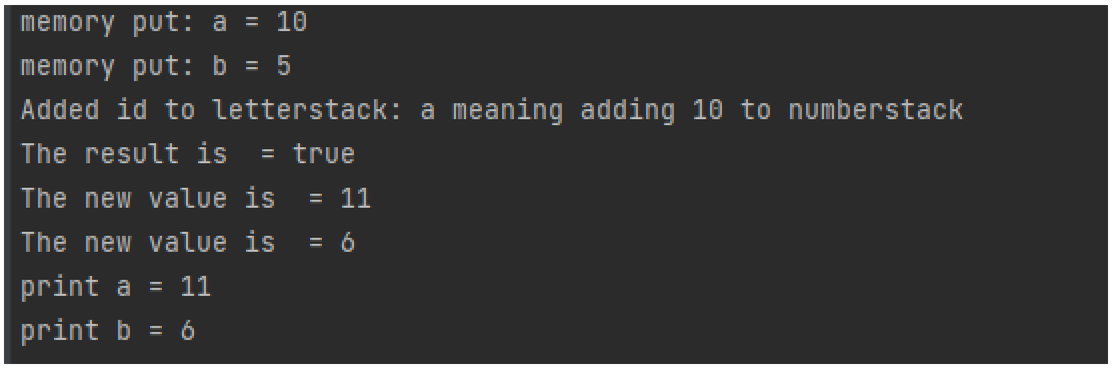
```
memory put: i = 0
Added id to letterstack: i meaning adding 0 to numberstack
The result is  = true
Added id to letterstack: i meaning adding 0 to numberstack
memory put: i = 1
print i = 1
Added id to letterstack: i meaning adding 1 to numberstack
The result is  = true
Added id to letterstack: i meaning adding 3 to numberstack
memory put: i = 4
print i = 4
Added id to letterstack: i meaning adding 4 to numberstack
The result is  = true
Added id to letterstack: i meaning adding 4 to numberstack
memory put: i = 5
print i = 5
Added id to letterstack: i meaning adding 5 to numberstack
The result is  = true
Added id to letterstack: i meaning adding 5 to numberstack
memory put: i = 6
print i = 6
Added id to letterstack: i meaning adding 6 to numberstack
The result is  = false
```

3.
   ```
int a = 10
int b = 5
if (a <= 11)
   a++
 b++
print a
print b
```

Parse Tree

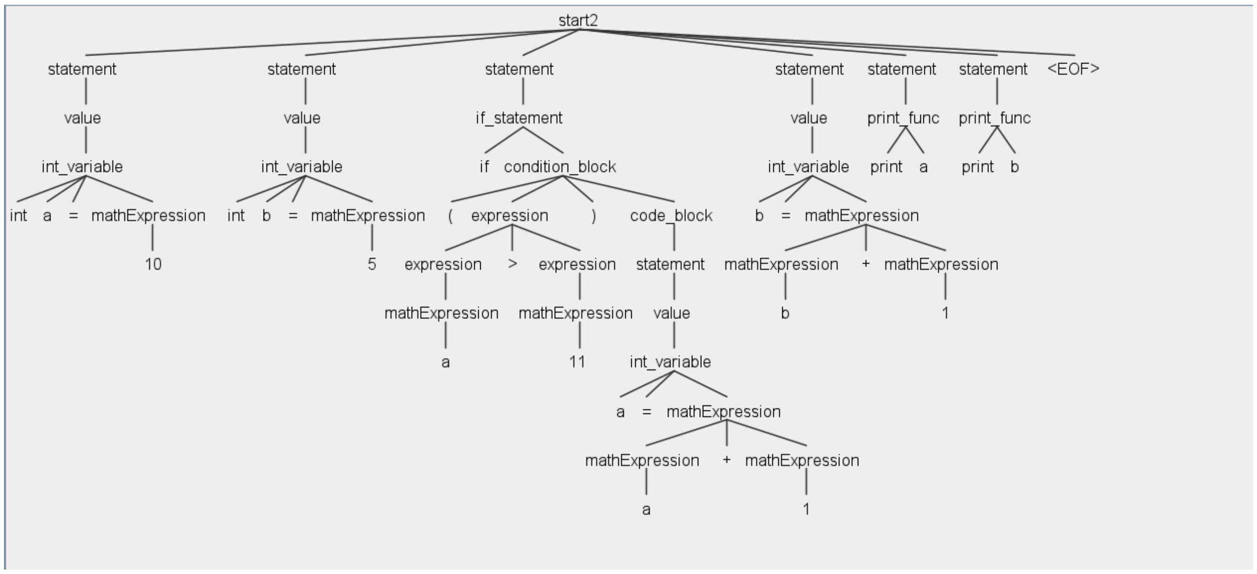


Terminal

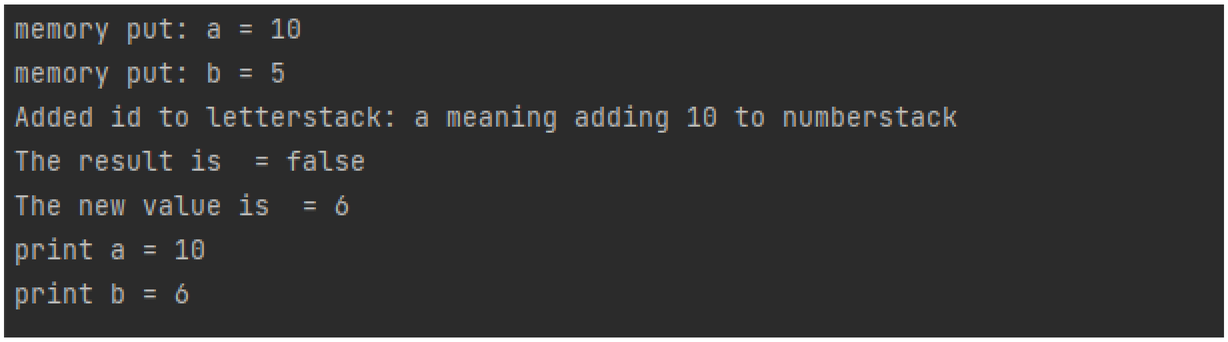

```
memory put: a = 10
memory put: b = 5
Added id to letterstack: a meaning adding 10 to numberstack
The result is  = true
The new value is  = 11
The new value is  = 6
print a = 11
print b = 6
```

4.
```
int a = 10
int b = 5
if (a > 11)
    a++
 b++

print a
print b
```

Parse Tree

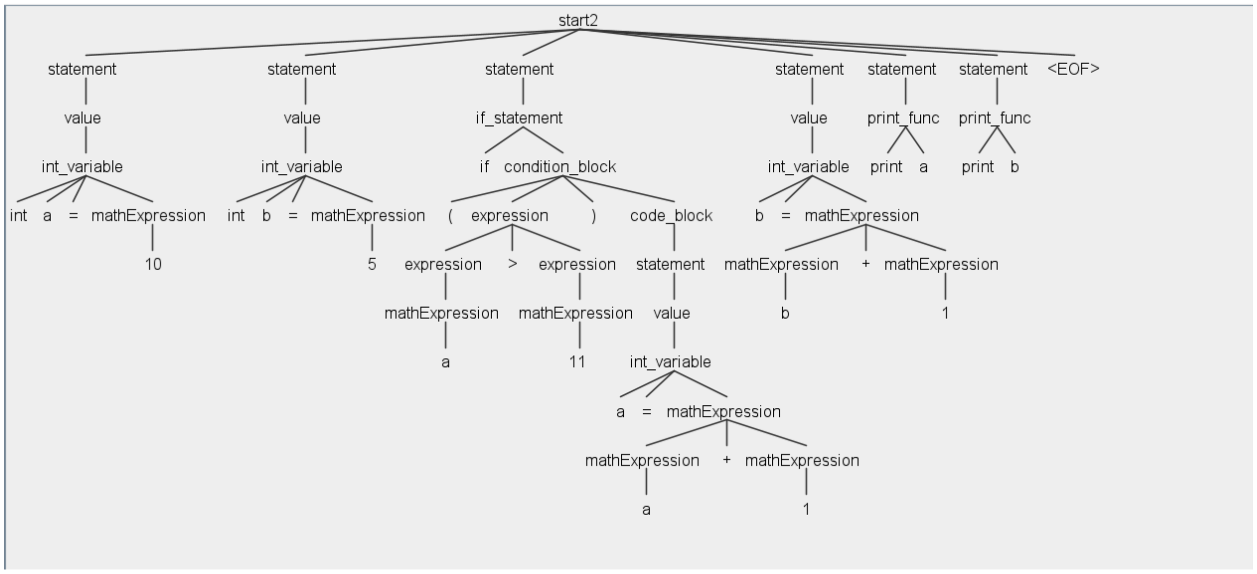

start2
statement statement statement statement statement statement <EOF>
value value if_statement value print_func print_func
int_variable int_variable if condition_block int_variable print a print b
int a = mathExpression int b = mathExpression (expression) code_block b = mathExpression
10 5 expression > expression statement mathExpression + mathExpression
mathExpression mathExpression value b 1
a 11 int_variable
a = mathExpression
mathExpression + mathExpression
a 1

Terminal

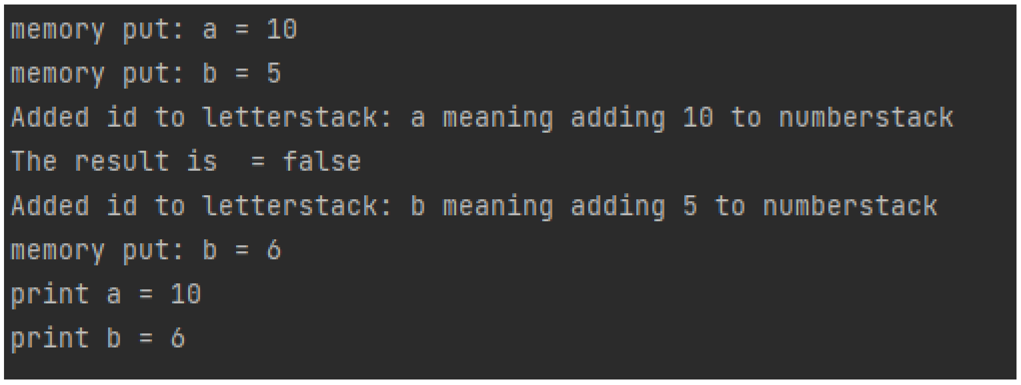

```
memory put: a = 10
memory put: b = 5
Added id to letterstack: a meaning adding 10 to numberstack
The result is  = false
The new value is  = 6
print a = 10
print b = 6
```

5.
```
int a = 10
int b = 5
if (a > 11)
    a = a + 1
 b = b + 1

print a
print b
```

Parse Tree



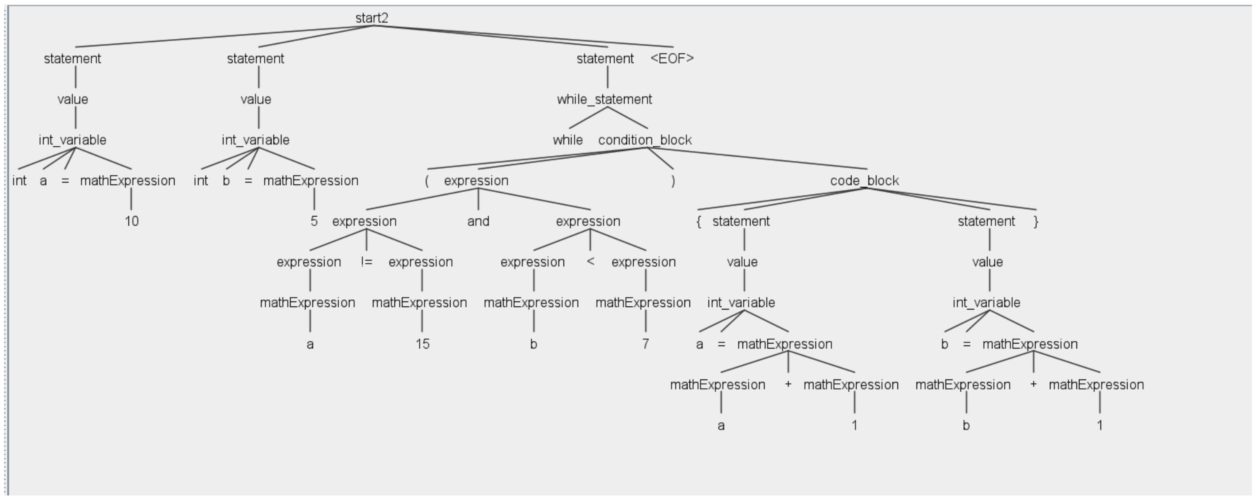
Terminal



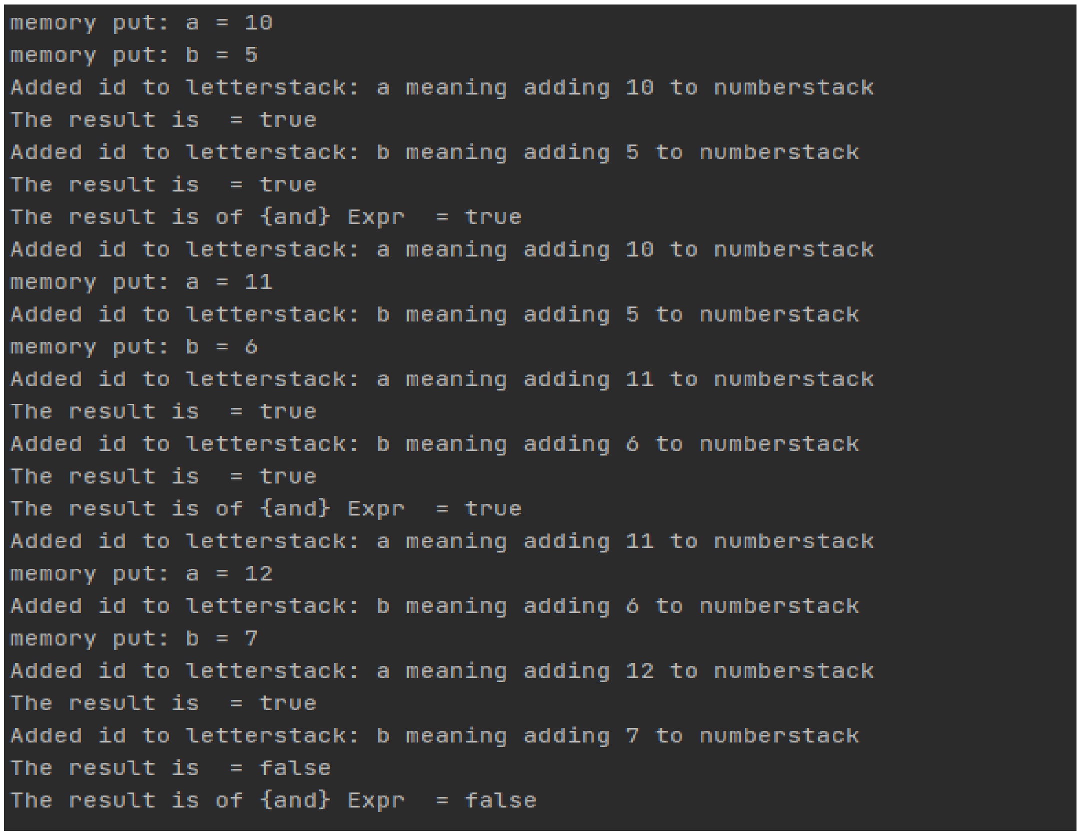
```
memory put: a = 10
memory put: b = 5
Added id to letterstack: a meaning adding 10 to numberstack
The result is  = false
Added id to letterstack: b meaning adding 5 to numberstack
memory put: b = 6
print a = 10
print b = 6
```

6.
```
int a = 10
int b = 5
while (a != 15 and b < 7)
{
    a = a + 1
    b = b + 1
}
```

Parse Tree



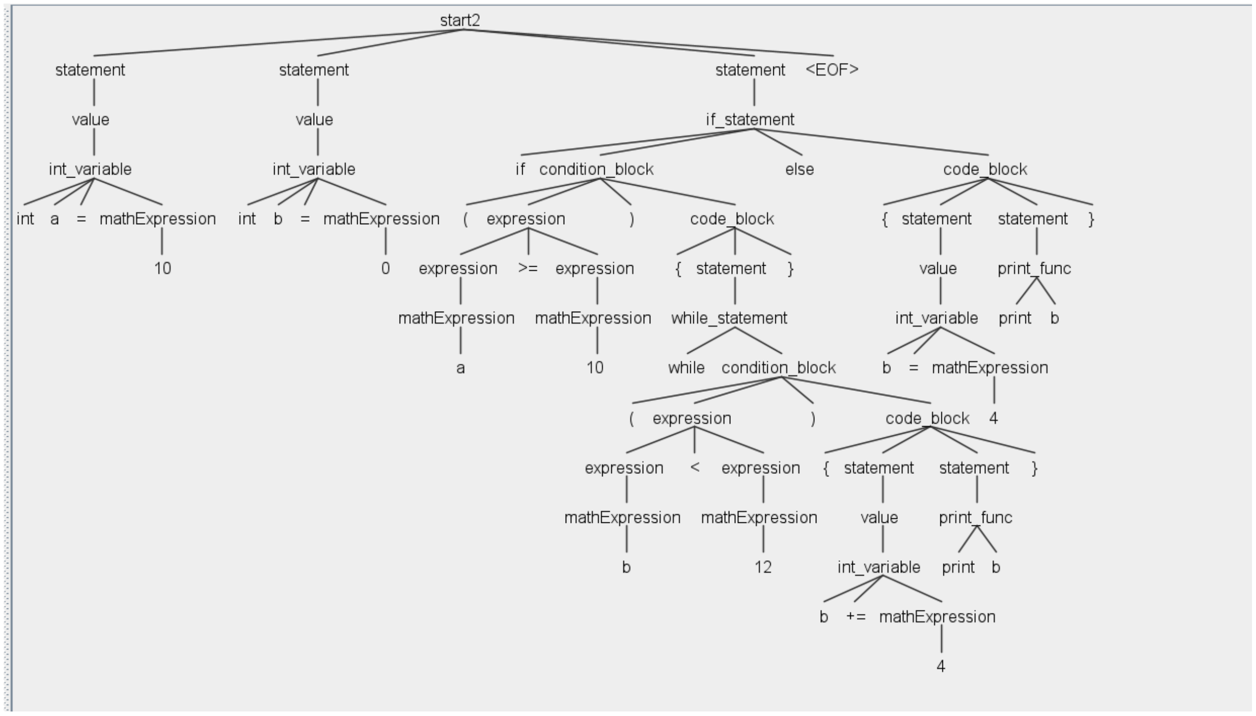
Terminal



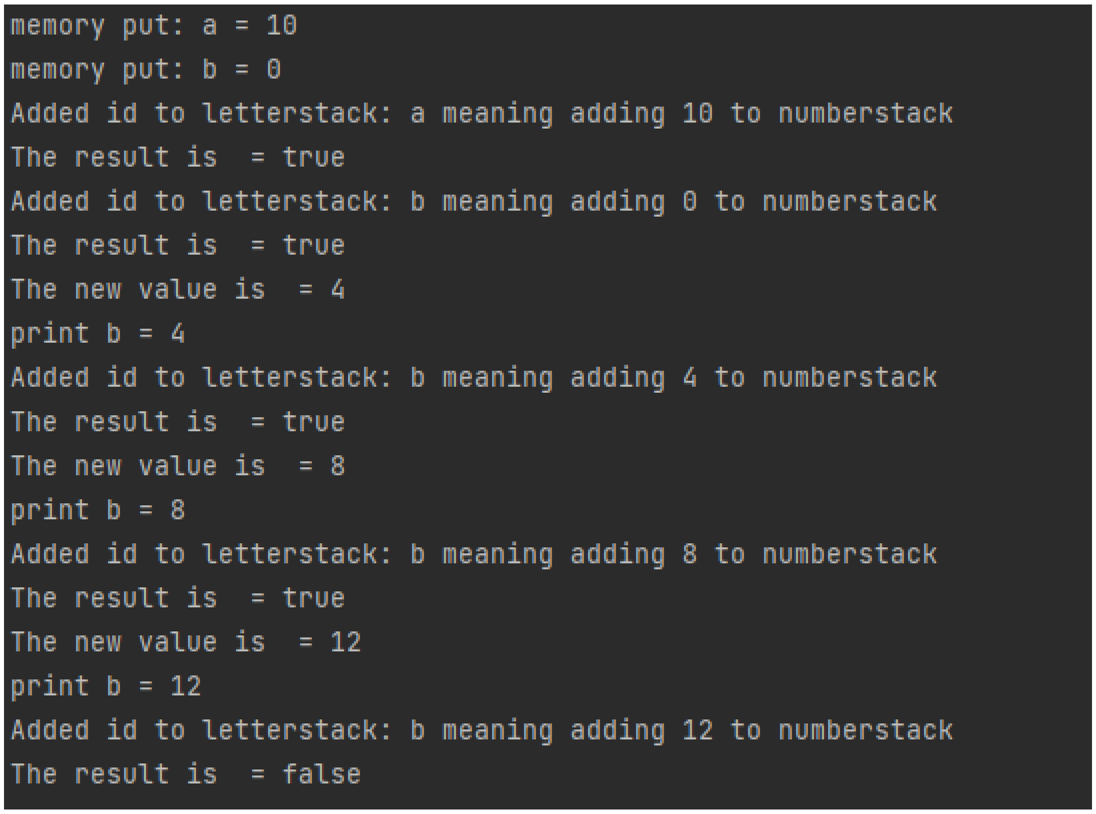
```
memory put: a = 10
memory put: b = 5
Added id to letterstack: a meaning adding 10 to numberstack
The result is  = true
Added id to letterstack: b meaning adding 5 to numberstack
The result is  = true
The result is of {and} Expr  = true
Added id to letterstack: a meaning adding 10 to numberstack
memory put: a = 11
Added id to letterstack: b meaning adding 5 to numberstack
memory put: b = 6
Added id to letterstack: a meaning adding 11 to numberstack
The result is  = true
Added id to letterstack: b meaning adding 6 to numberstack
The result is  = true
The result is of {and} Expr  = true
Added id to letterstack: a meaning adding 11 to numberstack
memory put: a = 12
Added id to letterstack: b meaning adding 6 to numberstack
memory put: b = 7
Added id to letterstack: a meaning adding 12 to numberstack
The result is  = true
Added id to letterstack: b meaning adding 7 to numberstack
The result is  = false
The result is of {and} Expr  = false
```

7.
```
int a = 10
int b = 0
if (a >= 10){
  while(b < 12){
      b += 4
      print b
   }
}
else{
      b = 4
      print b

}
```

Parse Tree



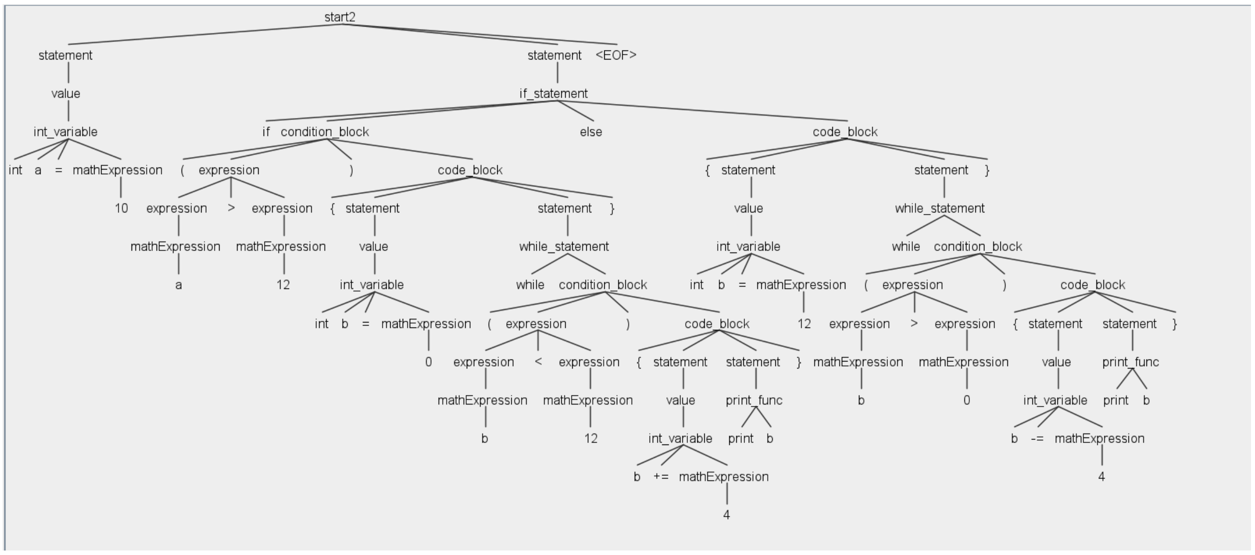
Terminal



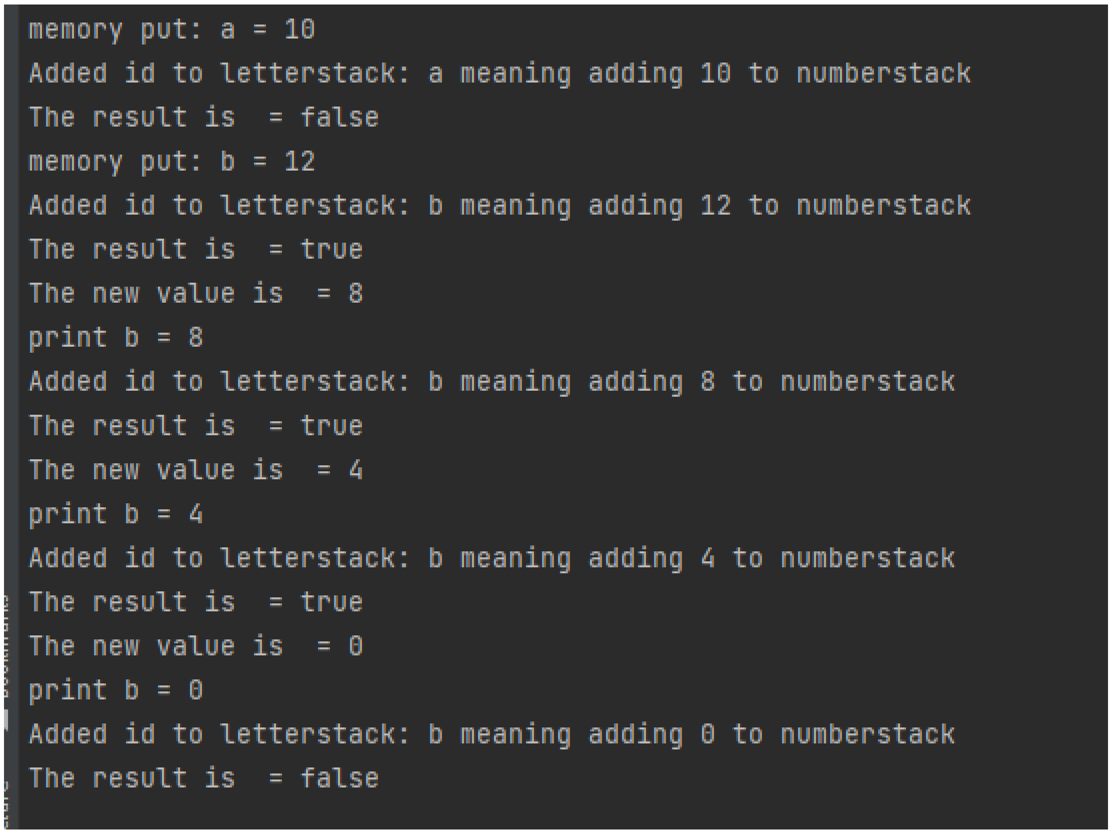
```
memory put: a = 10
memory put: b = 0
Added id to letterstack: a meaning adding 10 to numberstack
The result is  = true
Added id to letterstack: b meaning adding 0 to numberstack
The result is  = true
The new value is  = 4
print b = 4
Added id to letterstack: b meaning adding 4 to numberstack
The result is  = true
The new value is  = 8
print b = 8
Added id to letterstack: b meaning adding 8 to numberstack
The result is  = true
The new value is  = 12
print b = 12
Added id to letterstack: b meaning adding 12 to numberstack
The result is  = false
```

8.
```
int a = 10
if (a > 12){
   int b = 0
   while(b < 12){
       b += 4
       print b
    }
}
else{
   int b = 12
   while(b > 0){
       b -= 4
       print b
   }
}
```

Parse Tree


start2
statement statement <EOF>
value if_statement
int_variable if condition_block else code_block
int a = mathExpression (expression) code_block { statement statement }
10 expression > expression { statement statement } value while_statement
mathExpression mathExpression value while_statement int_variable while condition_block
a 12 int_variable while condition_block int b = mathExpression (expression) code_block
int b = mathExpression (expression) code_block 12 expression > expression { statement statement }
0 expression < expression { statement statement } mathExpression mathExpression value print_func
mathExpression mathExpression value print_func b 0 int_variable print b
b 12 int_variable print b b -= mathExpression
b += mathExpression 4
4

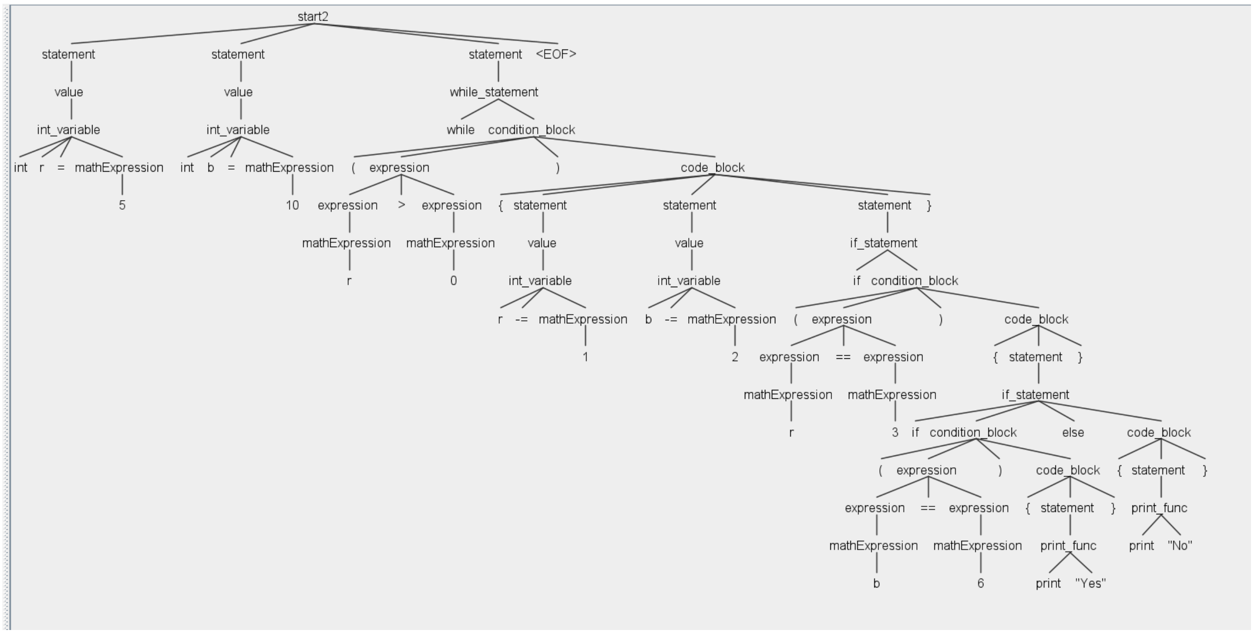
Terminal



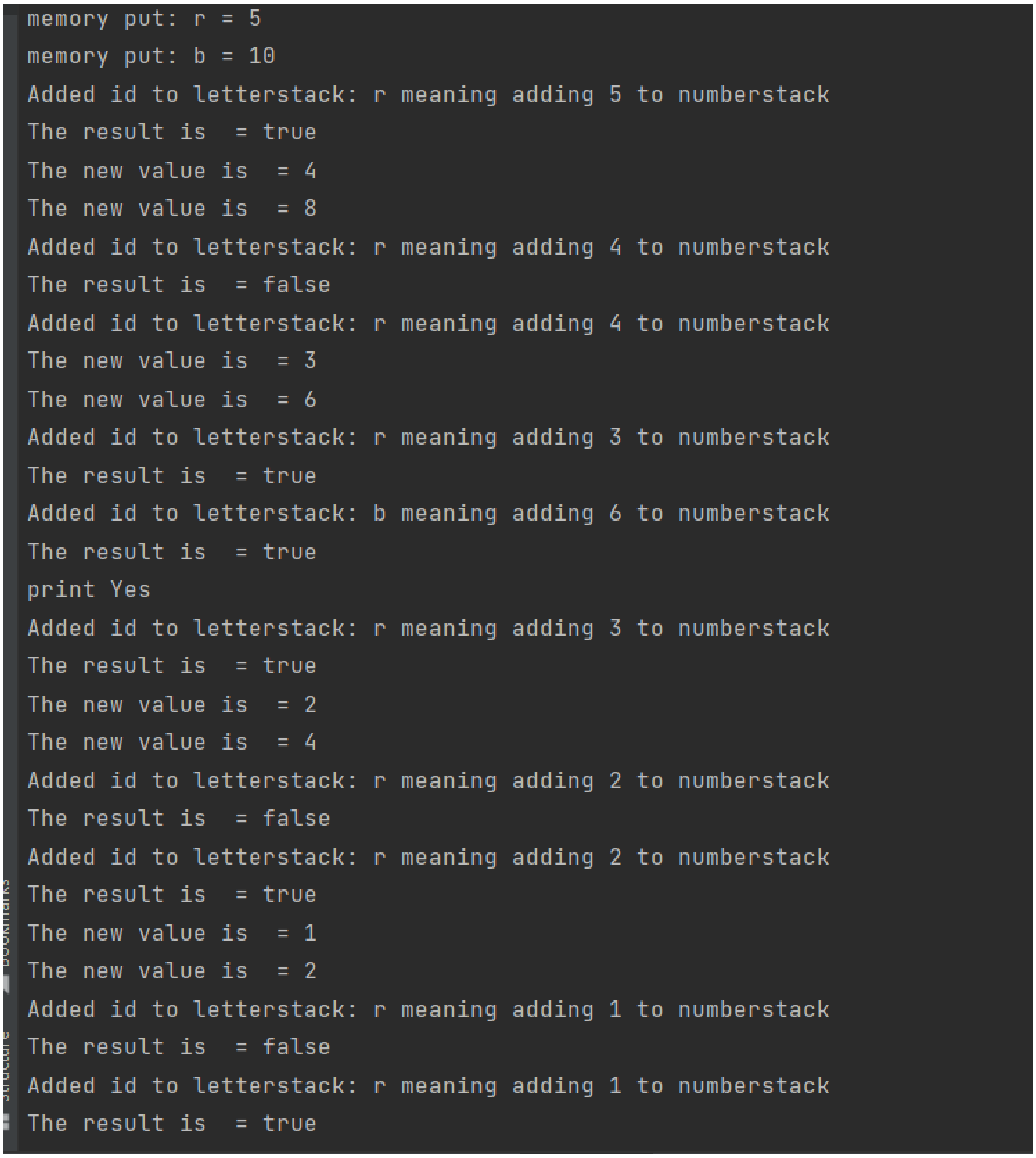
```
memory put: a = 10
Added id to letterstack: a meaning adding 10 to numberstack
The result is  = false
memory put: b = 12
Added id to letterstack: b meaning adding 12 to numberstack
The result is  = true
The new value is  = 8
print b = 8
Added id to letterstack: b meaning adding 8 to numberstack
The result is  = true
The new value is  = 4
print b = 4
Added id to letterstack: b meaning adding 4 to numberstack
The result is  = true
The new value is  = 0
print b = 0
Added id to letterstack: b meaning adding 0 to numberstack
The result is  = false
```

9.
```
int r = 5
int b = 10
while(r > 0){
    r -= 1
    b -= 2
    if(r == 3){
       if(b == 6){
         print "Yes"
       }
       else{
         print "No"
       }
    }
}
```

Parse Tree



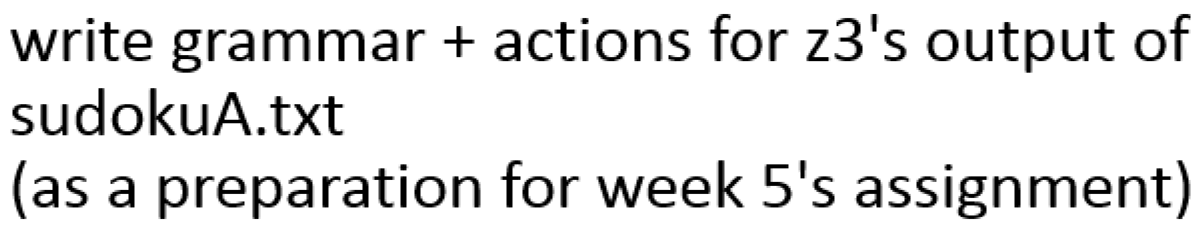
Terminal



```
memory put: r = 5
memory put: b = 10
Added id to letterstack: r meaning adding 5 to numberstack
The result is  = true
The new value is  = 4
The new value is  = 8
Added id to letterstack: r meaning adding 4 to numberstack
The result is  = false
Added id to letterstack: r meaning adding 4 to numberstack
The new value is  = 3
The new value is  = 6
Added id to letterstack: r meaning adding 3 to numberstack
The result is  = true
Added id to letterstack: b meaning adding 6 to numberstack
The result is  = true
print Yes
Added id to letterstack: r meaning adding 3 to numberstack
The result is  = true
The new value is  = 2
The new value is  = 4
Added id to letterstack: r meaning adding 2 to numberstack
The result is  = false
Added id to letterstack: r meaning adding 2 to numberstack
The result is  = true
The new value is  = 1
The new value is  = 2
Added id to letterstack: r meaning adding 1 to numberstack
The result is  = false
Added id to letterstack: r meaning adding 1 to numberstack
The result is  = true
```

Z3 SudokuA

Problem description



write grammar + actions for z3's output of sudokuA.txt
(as a preparation for week 5's assignment)

Implementation

Besides the extensions of our initial grammar with if statements and while statements, we have also been tasked to write a grammar for z3 output based on the given file "sudokuA.txt", and to also add functionality to display the output accordingly (as a matrix or as a grid as it can be seen in Output)

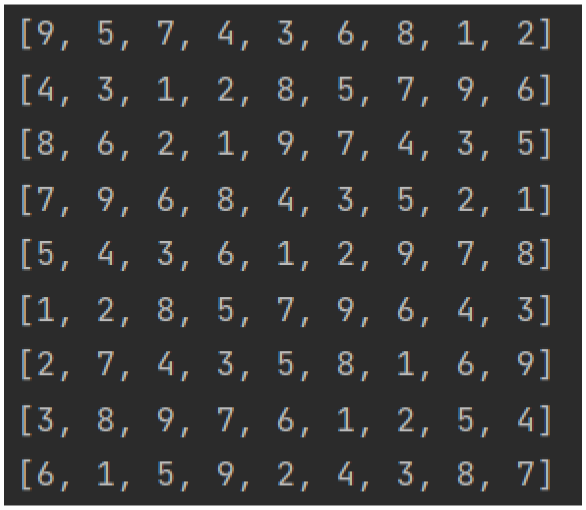
We defined the 9x9 board which will be used when we visit the model so that we can add the value of the variables into the board.

Finally, we have two methods which deal with printing the output either as a matrix or as a Sudoku grid.

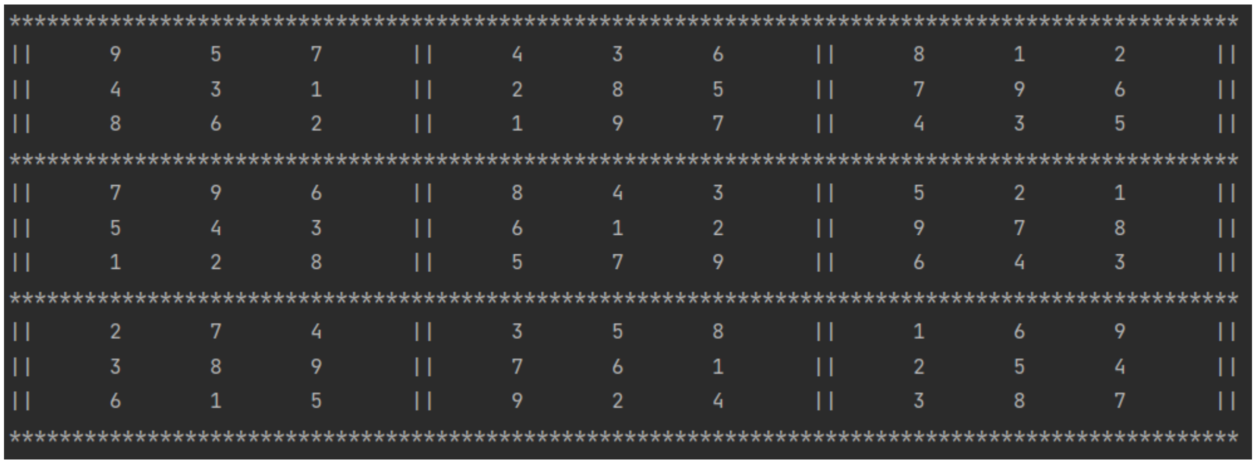
```
grammar Example2;

// rules
start2    : checkSatResponse checkModelResponse EOF;

checkSatResponse    : SAT | UNSAT;

checkModelResponse  : PARANL model* PARANR;

model : (PARANL DEFINE_FUN ID PARANL PARANR INT NUMBER PARANR)    #modelR;

//tokens
MODEL              : 'model';
INT                : 'Int';
NUMBER             : [0-9]+;
```

```
TEXT                : '"' ~('\r' | '\n' | '"')* '"';
UNSAT               : 'unsat';
SAT                 : 'sat';
DEFINE_FUN          : 'define-fun';
ID                  :  [_A-Za-z][A-Za-z_!0-9.]* ;
PARANL              : '(';
PARANR              : ')';
NEWLINE             : [ \t\r\n]+ -> skip;
```

Output



```
[9, 5, 7, 4, 3, 6, 8, 1, 2]
[4, 3, 1, 2, 8, 5, 7, 9, 6]
[8, 6, 2, 1, 9, 7, 4, 3, 5]
[7, 9, 6, 8, 4, 3, 5, 2, 1]
[5, 4, 3, 6, 1, 2, 9, 7, 8]
[1, 2, 8, 5, 7, 9, 6, 4, 3]
[2, 7, 4, 3, 5, 8, 1, 6, 9]
[3, 8, 9, 7, 6, 1, 2, 5, 4]
[6, 1, 5, 9, 2, 4, 3, 8, 7]
```



```
****************************************************************************************
||      9      5      7      ||     4      3      6      ||     8      1      2      ||
||      4      3      1      ||     2      8      5      ||     7      9      6      ||
||      8      6      2      ||     1      9      7      ||     4      3      5      ||
****************************************************************************************
||      7      9      6      ||     8      4      3      ||     5      2      1      ||
||      5      4      3      ||     6      1      2      ||     9      7      8      ||
||      1      2      8      ||     5      7      9      ||     6      4      3      ||
****************************************************************************************
||      2      7      4      ||     3      5      8      ||     1      6      9      ||
||      3      8      9      ||     7      6      1      ||     2      5      4      ||
||      6      1      5      ||     9      2      4      ||     3      8      7      ||
****************************************************************************************
```