Rachel Chiang
CS 411-01
Project #2
Due: 17.03.09, 1500

Conflict Resolution.

```
                                    +------+
                                    |RUN 01|
                                    +------+
$ yacc -d parser.y
parser.y: warning: 219 shift/reduce conflicts [-Wconflicts-sr]
parser.y:163.6: warning: rule useless in parser due to conflicts [-Wother]
 Field: // 9
      ^

FIX: I accidentally put in a | for the first rule of Field, which actually reduced
the conflicts by a little bit.

                                    +------+
                                    |RUN 02|
                                    +------+
$ yacc -d -v parser.y
parser.y: warning: 212 shift/reduce conflicts [-Wconflicts-sr]

FIX: Added operator precedence.

                                    +------+
                                    |RUN 03|
                                    +------+
$ yacc -d -v parser.y
parser.y: warning: 4 shift/reduce conflicts [-Wconflicts-sr]

   (1) State 57 conflicts: 1 shift/reduce
   (2) State 67 conflicts: 1 shift/reduce
   (3) State 87 conflicts: 1 shift/reduce
   (4) State 174 conflicts: 1 shift/reduce

Offending States:
       State 57 (1)
          36 StmtBlock: _leftbrace . StmtVarDecl StmtBlockStmt _rightbrace

           _boolean  shift, and go to state 1
           _double   shift, and go to state 3
           _int      shift, and go to state 4
           _string   shift, and go to state 6
           _id       shift, and go to state 8

           _id       [reduce using rule 37 (StmtVarDecl)]
           $default  reduce using rule 37 (StmtVarDecl)

           VariableDecl  go to state 67
           Variable      go to state 12
           Type          go to state 38
           StmtVarDecl   go to state 68

       State 67 (2)
          38 StmtVarDecl: VariableDecl . StmtVarDecl
```

```
        _boolean  shift, and go to state 1
        _double   shift, and go to state 3
        _int      shift, and go to state 4
        _string   shift, and go to state 6
        _id       shift, and go to state 8

        _id       [reduce using rule 37 (StmtVarDecl)]
        $default  reduce using rule 37 (StmtVarDecl)

        VariableDecl  go to state 67
        Variable      go to state 12
        Type          go to state 38
        StmtVarDecl   go to state 71

    State 87 (3)
       83 Lvalue: _id .
       86 Call: _id . _leftparen Actuals _rightparen
       87     | _id . _period _id _leftparen Actuals _rightparen

        _period     shift, and go to state 115
        _leftparen  shift, and go to state 116

        _period   [reduce using rule 83 (Lvalue)]
        $default  reduce using rule 83 (Lvalue)

    State 174 (4)
       51 IfStmt: _if _leftparen Expr _rightparen Stmt . OptionalElse

        _else  shift, and go to state 181

        _else     [reduce using rule 52 (OptionalElse)]
        $default  reduce using rule 52 (OptionalElse)

        OptionalElse  go to state 182

CONFLICT 1 State 57 S/R Conflict
Related States and Rules:
State 8
   14 Type: _id .

    $default  reduce using rule 14 (Type)
Rule 37 StmtVarDecl: %empty
Rule 38            | VariableDecl StmtVarDecl
```

Fix: I recall reading somewhere that Yacc encourages left recursion, so I will remove the right recursion in StmtVarDecl and see how it goes.

```
+------+
|RUN 04|
+------+
```

After rearranging the recursion in StmtVarDecl and StmtBlockStmt… One was resolved!

```
$ yacc -d -v parser.y
parser.y: warning: 3 shift/reduce conflicts [-Wconflicts-sr]
```

```
State 67 conflicts: 1 shift/reduce
State 90 conflicts: 1 shift/reduce <-- The Lvalue Call conflict
State 172 conflicts: 1 shift/reduce <-- The IfStmt Else conflict

       State 172
          51 IfStmt: _if _leftparen Expr _rightparen Stmt . OptionalElse

           _else  shift, and go to state 179

           _else     [reduce using rule 52 (OptionalElse)]
           $default  reduce using rule 52 (OptionalElse)

           OptionalElse  go to state 180
Rules:
   51 IfStmt: _if _leftparen Expr _rightparen Stmt OptionalElse

   52 OptionalElse: %empty
   53             | _else Stmt
```

In the provided PDF “Constructing LR Parsing Tables,” it states that to solve the shift-reduce if-else conflict, we “can just assert that for a shift-reduce conflict, we always shift and never reduce.” _else should latch on to the closest _if, so we want to make a lonely _if less important than a couple. Instead of having the OptionalElse production separate, I will combine the if and else together. Bison has a %prec that lets you label the rule so you can establish precedence, so I will make a new precedence label for a lonely _if and give the couple higher precedence.

```
+------+
|RUN 05|
+------+

Establishing precedence between lonely if’s and coupled if-else’s resolved the
conflict.
$ yacc -d -v parser.y
parser.y: warning: 2 shift/reduce conflicts [-Wconflicts-sr]

State 67 conflicts: 1 shift/reduce
State 90 conflicts: 1 shift/reduce <-- Lvalue Call conflict

       State 67
          36 StmtBlock: _leftbrace StmtVarDecl . StmtBlockStmt _rightbrace
          38 StmtVarDecl: StmtVarDecl . VariableDecl

           _boolean  shift, and go to state 1
           _double   shift, and go to state 3
           _int      shift, and go to state 4
           _string   shift, and go to state 6
           _id       shift, and go to state 8

           _id       [reduce using rule 39 (StmtBlockStmt)]
           $default  reduce using rule 39 (StmtBlockStmt)

           VariableDecl   go to state 70
           Variable       go to state 12
```

```
          Type           go to state 38
          StmtBlockStmt  go to state 71

Rules
   39 StmtBlockStmt: %empty
   40              | StmtBlockStmt Stmt

       State 90
          82 Lvalue: _id .
          85 Call: _id . _leftparen Actuals _rightparen
          86     | _id . _period _id _leftparen Actuals _rightparen

          _period     shift, and go to state 115
          _leftparen  shift, and go to state 116

          _period   [reduce using rule 82 (Lvalue)]
          $default  reduce using rule 82 (Lvalue)

Rules:
   60 Expr: Lvalue _assignop Expr
   61     | Constant
   62     | Lvalue
   63     | Call
   64     | _leftparen Expr _rightparen
   65     | Expr _add Expr
   66     | Expr _sub Expr
   67     | Expr _mult Expr
   68     | Expr _div Expr
   69     | Expr _mod Expr
   70     | _sub Expr
   71     | Expr _less Expr
   72     | Expr _lessequal Expr
   73     | Expr _greater Expr
   74     | Expr _greaterequal Expr
   75     | Expr _equal Expr
   76     | Expr _notequal Expr
   77     | Expr _and Expr
   78     | Expr _or Expr
   79     | _not Expr
   80     | _readln _leftparen _rightparen
   81     | _newarray _leftparen _intconstant _comma Type _rightparen

   85 Call: _id _leftparen Actuals _rightparen
   86     | _id _period _id _leftparen Actuals _rightparen

   82 Lvalue: _id
   83       | Lvalue _leftbracket Expr _rightbracket
   84       | Lvalue _period _id
```

I actually attempted several approaches, including rewriting the grammar, but most of them proved to be fruitless. For instance, I added labels similar to what was done in the if-else, or I would try to reduce the empty transitions or change the grammar, but the latter seems to be a great way to introduce reduce-reduce conflicts. I eventually just messed around with the precedence until it stopped giving me conflicts. I don't really consider that very effective conflict-resolution though.

Test Cases.
These are listed in alphabetical order by filename. My invalid test cases are admittedly not very sophisticated. I don't think it's practical to make a whole bunch of errors in one test file because the errors and rejections usually come before the end of the file.

Input: BadExpressions.toy
```
void main()
{
   a = 3 + 2 * 5; //
   int a; // Variable Declarations go before Stmts
   // if there aren't any {} denoting a new StmtBlock
}
```
Output: BadExpressions.out
```
void [shift]
id [shift]
([shift]
)[reduce 07.2][shift]
{[shift]
[reduce 12.1.1.1]id [reduce 12.1.2.1][shift]
= [reduce 21.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]add [shift]
intconst [shift]
[reduce 24.1][reduce 20.02]mult [shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.08]; [reduce 20.06][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]int [reject]
```

Input: Fail1.toy
```
/*
This toy program won't work.
*/
class Fail extends Sad implements Terrible, Bad
{
   int aGlobal;
   void stats(int first)
   {
       if (first < 100)
       {
           return;
       }
       else
       {
           aGlobal = first;
       }
   }
   println(e, f); // Rejected
}
```
Output: Fail1.out
```
class [shift]
id [shift]
extends [shift]
```

```
id [shift]
[reduce 08.1.1.2]implements [shift]
id [shift]
, [shift]
id [shift]
{[reduce 08.1.2.2.2][reduce 08.1.2.2.1][reduce 08.1.2.2][shift]
[reduce 08.1.3.1]int [shift]
[reduce 05.1]id [shift]
; [reduce 04.1][shift]
[reduce 03.1][reduce 09.1][reduce 08.1.3.2]void [shift]
id [shift]
([shift]
int [shift]
[reduce 05.1]id [shift]
[reduce 04.1])[reduce 07.1.2][reduce 07.1][shift]
{[shift]
[reduce 12.1.1.1]if [reduce 12.1.2.1][shift]
([shift]
id [shift]
< [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 20.12][shift]
{[shift]
[reduce 12.1.1.1]return [reduce 12.1.2.1][shift]
; [reduce 25.1][shift]
[reduce 18.1][reduce 13.6][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 13.8]else [shift]
{[shift]
[reduce 12.1.1.1]id [reduce 12.1.2.1][shift]
= [reduce 21.1][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 13.8][reduce 14.2][reduce 13.2][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 06.2][reduce 09.2][reduce 08.1.3.2]println [reject]
```

Input: InterfaceTest.toy

```
/*
Program => Decl [r 01.1]
   Decl => InterfaceDecl [r 02.4]
      InterfaceDecl => _interface _id _leftbrace Prototype _rightbrace [r 10.1]
         Prototype => Type _id _leftparen Formals _rightparen _semicolon [r 11.2]
            Formals => VarList [r 07.1]
               VarList => Variable [r 07.1.2]
                  Variable => Type _id [04.1]
                     Type => _double [05.2]
            Type => _int
*/
interface tester
{
   int round(double a);
}
```

Output: InterfaceTest.out

```
interface [shift]
```

```
id [shift]
{[shift]
int [shift]
[reduce 05.1]id [shift]
([shift]
double [shift]
[reduce 05.2]id [shift]
[reduce 04.1])[reduce 07.1.2][reduce 07.1][shift]
; [shift]
[reduce 11.2]}[shift]
[reduce 10.1][reduce 02.4][reduce 01.1][shift]
[accept]
```

Input: Nests.toy

```
//VariableDecl => Type _id _semicolon
double a;
//FunctionDecl => Type _id _leftparen Formals _rightparen StmtBlock
void someName(string s, boolean b, double d)
{
   int num;
//   num = s.length; // rejected
   //Stmt => StmtBlock
   {
      int localValue;
//      localValue = 10;
//      localValue = readln(); // rejected
      println("%d * 2 = ", localValue);
   }
   num = num + 10;
}

//ClassDecl => FunctionDecl
class ClassA extends ClassB implements ClassC, ClassD
{
   void aMethod(int a, int b)
   {
      if (a == b)
      {
         int i;
         for (i = 0; i < 3; i = i + 1)
         {
/* Case 1: if-else with empty blocks in brackets and an Expr after
This is ACCEPTED. Also accepts if the last Expr is left out
            if (a%2 == 0)
            {
            }
            else
            {
            }
            a = math.random();
*/
/* Case 2: if-else with Expr inside
This is REJECTED with and without the last Expr
            if (a%2 == 0)
```

```
            {
               b = b * a;
            }
            else
            {
               b = b * 2;
            }
            a = math.random();
*/
// Case 3: if-else without brackets with Exprs inside and an Expr after
//This is ACCEPTED with and without the last Expr and any amount or type of
subsequent Exprs
            if (a%2 == 0)
               b = b * a;
            else
               b = b * 2;
            a = math.random();
/* Case 4: No if-else and only Expr
This is REJECTED
            a = math.random();
*/
         }
      }
      super.run(b);
   }
}

//InterfaceDecl
interface SomeInterface
{
   // Empty is also accepted.
   void pt1();
   void pt2(int a);
   void pt3(Square[] squaresList);
   void pt4(int a, double b, Square s);
   boolean pt5();
   double pt6(Square sq);
   Square pt7(double a, double b);
}
```

Output: Nests.out

```
boolean [shift]
[reduce 05.3]id [shift]
([shift]
string [shift]
[reduce 05.4]id [shift]
[reduce 04.1])[reduce 07.1.2][reduce 07.1][shift]
{[shift]
[reduce 12.1.1.1]int [shift]
[reduce 05.1][[shift]
][shift]
[reduce 05.5]id [shift]
[reduce 04.1]; [shift]
[reduce 03.1][reduce 12.1.1.2]int [shift]
[reduce 05.1]id [shift]
[reduce 04.1]; [shift]
```

```
[reduce 03.1][reduce 12.1.1.2]int [shift]
[reduce 05.1]id [shift]
[reduce 04.1]; [shift]
[reduce 03.1][reduce 12.1.1.2]id [reduce 12.1.2.1][shift]
= [reduce 21.1][shift]
id [shift]
. [shift]
id [shift]
([shift]
)[reduce 23.1][shift]
[reduce 22.2][reduce 20.04]; [reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
id [shift]
. [shift]
id [shift]
([shift]
)[reduce 23.1][shift]
[reduce 22.2][reduce 20.04]; [reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]while [shift]
([shift]
id [shift]
!= [reduce 21.1][reduce 20.03][shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 20.17][shift]
{[shift]
[reduce 12.1.1.1]if [reduce 12.1.2.1][shift]
([shift]
id [shift]
[[reduce 21.1][shift]
id [shift]
][reduce 21.1][reduce 20.03][shift]
[reduce 21.2]!= [reduce 20.03][shift]
id [shift]
[[reduce 21.1][shift]
id [shift]
][reduce 21.1][reduce 20.03][shift]
[reduce 21.2])[reduce 20.03][reduce 20.17][shift]
{[shift]
[reduce 12.1.1.1]return [reduce 12.1.2.1][shift]
booleanconstF [shift]
[reduce 24.4][reduce 20.02]; [reduce 25.2][shift]
[reduce 18.1][reduce 13.6][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 13.8]id [reduce 14.1][reduce 13.2][reduce 12.1.2.2][shift]
= [reduce 21.1][shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.06][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
```

```
id [shift]
sub [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.07][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 13.8][reduce 15.1][reduce 13.3][reduce 12.1.2.2]return [shift]
booleanconstT [shift]
[reduce 24.4][reduce 20.02]; [reduce 25.2][shift]
[reduce 18.1][reduce 13.6][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 06.1][reduce 02.2][reduce 01.1][shift]
[accept]
```

Input: Nonsense.toy

Actually, during my rapid testing, many of these expressions were used, except they used to all just sit in Square.toy, which didn't make sense, but I had it open at a convenient time. In what used to be Square.toy, I tested all sorts of weird combinations of expressions and values together and in different orders. Anyway, Nonsense.toy is just what was left after I gutted the old amalgamation of expressions. Notably, there are nested if's and there is an incomplete while loop in this one.

```
void main()
{
   a = b + 5;
   a / 5;
   a * 4 + 3 / 6;
   a();
   b;
   a[5];
   a(5);
   a(b+c);
   a.b();
   foo.go();
   a = b;
   a + b;
   a + 5;
   a[5] = b;
   ridiculousNonsense(a, b);
}

void ridiculousNonsense(int a, int b)
{
      // accepted...
      (5+5);

      while (a == b)
            break;
      -5;
      println(a);
      readln();
      newarray (4, int);
      return;
      break;

      if (a != 1)
```

```
            a + a;
      else
            a * b;
      a * b;
      1 * 3;
      print(4 + 8);
      4 >= 5;
      5;
      !5;
      ;
      5 + a;
      5 * 4 + 3 / 6;
      5 * (a + 3) / 6;
      5 * 4 + a / 6;
      5 * 4 + 3 / a;
      (6 < a);
      // Nested if-if-else with brackets
      if (a == b)
      {
         if (a > 5)
             a % 2;
         else
             a % 3;
      }

      // Nested if-if-else without brackets
      if (a == b)
         if (a > 5)
             a % 2;
         else
             a % 3;

      a = 5 + 5;

      a = 5 + b;

      a = b + 5;

      for (; a<5;)
      a + b;

      for (i=5; i<10; i = i + 1)
         a = a + b;
}
```

Output: Nonsense.out

```
void [shift]
id [shift]
([shift]
)[reduce 07.2][shift]
{[shift]
[reduce 12.1.1.1]id [reduce 12.1.2.1][shift]
= [reduce 21.1][shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
intconst [shift]
```

```
[reduce 24.1][reduce 20.02]; [reduce 20.06][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
div [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.09]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
mult [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.08]add [shift]
intconst [shift]
[reduce 24.1][reduce 20.02]div [shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.09]; [reduce 20.06][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
([shift]
)[reduce 23.1][shift]
[reduce 22.1][reduce 20.04]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
; [reduce 21.1][reduce 20.03][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
[[reduce 21.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]][shift]
[reduce 21.2]; [reduce 20.03][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
([shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 26.1][reduce 23.2][shift]
[reduce 22.1][reduce 20.04]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
([shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 20.06][reduce 26.1][reduce 23.2][shift]
[reduce 22.1][reduce 20.04]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
. [shift]
id [shift]
([shift]
)[reduce 23.1][shift]
[reduce 22.2][reduce 20.04]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
. [shift]
id [shift]
([shift]
)[reduce 23.1][shift]
[reduce 22.2][reduce 20.04]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
add [reduce 21.1][reduce 20.03][shift]
id [shift]
```

```
; [reduce 21.1][reduce 20.03][reduce 20.06][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
add [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.06][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
[[reduce 21.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]][shift]
[reduce 21.2]= [shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
([shift]
id [shift]
, [reduce 21.1][reduce 20.03][shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 26.1][reduce 26.2][reduce 23.2][shift]
[reduce 22.1][reduce 20.04]; [shift]
[reduce 13.1][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 06.2][reduce 02.2][reduce 01.1]void [shift]
id [shift]
([shift]
int [shift]
[reduce 05.1]id [shift]
[reduce 04.1], [shift]
int [shift]
[reduce 05.1]id [shift]
[reduce 04.1])[reduce 07.1.2][reduce 07.1.1][reduce 07.1][shift]
{[shift]
[reduce 12.1.1.1]([reduce 12.1.2.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]add [shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 20.06][shift]
[reduce 20.05]; [shift]
[reduce 13.1][reduce 12.1.2.2]while [shift]
([shift]
id [shift]
== [reduce 21.1][reduce 20.03][shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 20.16][shift]
break [shift]
; [shift]
[reduce 17.1][reduce 13.5][reduce 15.1][reduce 13.3][reduce 12.1.2.2]sub [shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.11][shift]
[reduce 13.1][reduce 12.1.2.2]println [shift]
([shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 26.1][shift]
; [shift]
[reduce 19.1][reduce 13.7][reduce 12.1.2.2]readln [shift]
([shift]
)[shift]
```

```
[reduce 20.21]; [shift]
[reduce 13.1][reduce 12.1.2.2]newarray [shift]
([shift]
intconst [shift]
, [shift]
int [shift]
[reduce 05.1])[shift]
[reduce 20.22]; [shift]
[reduce 13.1][reduce 12.1.2.2]return [shift]
; [reduce 25.1][shift]
[reduce 18.1][reduce 13.6][reduce 12.1.2.2]break [shift]
; [shift]
[reduce 17.1][reduce 13.5][reduce 12.1.2.2]if [shift]
([shift]
id [shift]
!= [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 20.17][shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.06][shift]
[reduce 13.1]else [shift]
id [shift]
mult [reduce 21.1][reduce 20.03][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.08][shift]
[reduce 13.1][reduce 14.2][reduce 13.2][reduce 12.1.2.2]id [shift]
mult [reduce 21.1][reduce 20.03][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.08][shift]
[reduce 13.1][reduce 12.1.2.2]intconst [shift]
[reduce 24.1][reduce 20.02]mult [shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.08]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
([shift]
intconst [shift]
void [shift]
id [shift]
([shift]
)[reduce 07.2][shift]
{[shift]
[reduce 12.1.1.1]id [reduce 12.1.2.1][shift]
= [reduce 21.1][shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.06][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
div [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.09]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
mult [reduce 21.1][reduce 20.03][shift]
```

```
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.08]add [shift]
intconst [shift]
[reduce 24.1][reduce 20.02]div [shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.09]; [reduce 20.06][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
([shift]
)[reduce 23.1][shift]
[reduce 22.1][reduce 20.04]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
; [reduce 21.1][reduce 20.03][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
[[reduce 21.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]][shift]
[reduce 21.2]; [reduce 20.03][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
([shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 26.1][reduce 23.2][shift]
[reduce 22.1][reduce 20.04]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
([shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 20.06][reduce 26.1][reduce 23.2][shift]
[reduce 22.1][reduce 20.04]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
. [shift]
id [shift]
([shift]
)[reduce 23.1][shift]
[reduce 22.2][reduce 20.04]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
. [shift]
id [shift]
([shift]
)[reduce 23.1][shift]
[reduce 22.2][reduce 20.04]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
add [reduce 21.1][reduce 20.03][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.06][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
add [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.06][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
[[reduce 21.1][shift]
```

```
intconst [shift]
[reduce 24.1][reduce 20.02]][shift]
[reduce 21.2]= [shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
([shift]
id [shift]
, [reduce 21.1][reduce 20.03][shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 26.1][reduce 26.2][reduce 23.2][shift]
[reduce 22.1][reduce 20.04]; [shift]
[reduce 13.1][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 06.2][reduce 02.2][reduce 01.1]void [shift]
id [shift]
([shift]
int [shift]
[reduce 05.1]id [shift]
[reduce 04.1], [shift]
int [shift]
[reduce 05.1]id [shift]
[reduce 04.1])[reduce 07.1.2][reduce 07.1.1][reduce 07.1][shift]
{[shift]
[reduce 12.1.1.1]([reduce 12.1.2.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]add [shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 20.06][shift]
[reduce 20.05]; [shift]
[reduce 13.1][reduce 12.1.2.2]while [shift]
([shift]
id [shift]
== [reduce 21.1][reduce 20.03][shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 20.16][shift]
break [shift]
; [shift]
[reduce 17.1][reduce 13.5][reduce 15.1][reduce 13.3][reduce 12.1.2.2]sub [shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.11][shift]
[reduce 13.1][reduce 12.1.2.2]println [shift]
([shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 26.1][shift]
; [shift]
[reduce 19.1][reduce 13.7][reduce 12.1.2.2]readln [shift]
([shift]
)[shift]
[reduce 20.21]; [shift]
[reduce 13.1][reduce 12.1.2.2]newarray [shift]
([shift]
intconst [shift]
, [shift]
int [shift]
[reduce 05.1])[shift]
```

```
[reduce 20.22]; [shift]
[reduce 13.1][reduce 12.1.2.2]return [shift]
; [reduce 25.1][shift]
[reduce 18.1][reduce 13.6][reduce 12.1.2.2]break [shift]
; [shift]
[reduce 17.1][reduce 13.5][reduce 12.1.2.2]if [shift]
([shift]
id [shift]
!= [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 20.17][shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.06][shift]
[reduce 13.1]else [shift]
id [shift]
mult [reduce 21.1][reduce 20.03][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.08][shift]
[reduce 13.1][reduce 14.2][reduce 13.2][reduce 12.1.2.2]id [shift]
mult [reduce 21.1][reduce 20.03][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.08][shift]
[reduce 13.1][reduce 12.1.2.2]intconst [shift]
[reduce 24.1][reduce 20.02]mult [shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.08]; [shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
([shift]
intconst [shift]
[reduce 24.1][reduce 20.02]add [shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 20.06][reduce 26.1][reduce 23.2][shift]
[reduce 22.1][reduce 20.04]; [shift]
[reduce 13.1][reduce 12.1.2.2]intconst [shift]
[reduce 24.1][reduce 20.02]>= [shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.15][shift]
[reduce 13.1][reduce 12.1.2.2]intconst [shift]
[reduce 24.1][reduce 20.02]; [shift]
[reduce 13.1][reduce 12.1.2.2]![shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.20]; [shift]
[reduce 13.1][reduce 12.1.2.2]; [shift]
[reduce 13.0][reduce 12.1.2.2]intconst [shift]
[reduce 24.1][reduce 20.02]add [shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.06][shift]
[reduce 13.1][reduce 12.1.2.2]intconst [shift]
[reduce 24.1][reduce 20.02]mult [shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.08]add [shift]
intconst [shift]
[reduce 24.1][reduce 20.02]div [shift]
```

```
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.09]; [reduce 20.06][shift]
[reduce 13.1][reduce 12.1.2.2]intconst [shift]
[reduce 24.1][reduce 20.02]mult [shift]
([shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 20.06][shift]
[reduce 20.05][reduce 20.08]div [shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.09]; [shift]
[reduce 13.1][reduce 12.1.2.2]intconst [shift]
[reduce 24.1][reduce 20.02]mult [shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.08]add [shift]
id [shift]
div [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.09]; [reduce 20.06][shift]
[reduce 13.1][reduce 12.1.2.2]intconst [shift]
[reduce 24.1][reduce 20.02]mult [shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.08]add [shift]
intconst [shift]
[reduce 24.1][reduce 20.02]div [shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.09][reduce 20.06][shift]
[reduce 13.1][reduce 12.1.2.2]([shift]
intconst [shift]
[reduce 24.1][reduce 20.02]< [shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 20.12][shift]
[reduce 20.05]; [shift]
[reduce 13.1][reduce 12.1.2.2]if [shift]
([shift]
id [shift]
== [reduce 21.1][reduce 20.03][shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 20.16][shift]
{[shift]
[reduce 12.1.1.1]if [reduce 12.1.2.1][shift]
([shift]
id [shift]
> [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 20.14][shift]
id [shift]
mod [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.10]; [shift]
[reduce 13.1]else [shift]
id [shift]
mod [reduce 21.1][reduce 20.03][shift]
intconst [shift]
```

```
[reduce 24.1][reduce 20.02][reduce 20.10]; [shift]
[reduce 13.1][reduce 14.2][reduce 13.2][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 13.8]if [reduce 14.1][reduce 13.2][reduce 12.1.2.2][shift]
([shift]
id [shift]
== [reduce 21.1][reduce 20.03][shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 20.16][shift]
if [shift]
([shift]
id [shift]
> [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 20.14][shift]
id [shift]
mod [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.10]; [shift]
[reduce 13.1]else [shift]
id [shift]
mod [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02][reduce 20.10]; [shift]
[reduce 13.1][reduce 14.2][reduce 13.2]id [reduce 14.1][reduce 13.2][reduce
12.1.2.2][shift]
= [reduce 21.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]add [shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.06][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]add [shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.06][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.06][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]for [shift]
([shift]
; [reduce 25.1][shift]
id [shift]
< [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.12][shift]
)[reduce 25.1][shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.06][shift]
[reduce 13.1][reduce 16.1][reduce 13.4][reduce 12.1.2.2]for [shift]
```

```
([shift]
id [shift]
= [reduce 21.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.01][reduce 25.2][shift]
id [shift]
< [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.12][shift]
id [shift]
= [reduce 21.1][shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 20.06][reduce 20.01][reduce 25.2][shift]
id [shift]
= [reduce 21.1][shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.06][reduce 20.01][shift]
[reduce 13.1][reduce 16.1][reduce 13.4][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 06.2][reduce 02.2][reduce 01.2][shift]
[accept]
```

Input: P1Example.toy

I actually encountered a problem in this one, sadly. For whatever reason, the parser works on everything above “flag = true;”. Unfortunately, I haven’t really found out why, but during the building process, the StmtBlock and Lvalue gave me a lot of trouble, and I investigated it for a long time, but I couldn’t figure it out. It seems like that’s where the issue comes in here: after the VariableDecl’s it doesn’t parse.

```
int fact (int x) {
// recursive factorial function
    if (x>1) return x * fact(x-1);
    else return 1;
}
void main () {
    int x;
    int total;
    println ("factorial of 10 is ", fact (10), " from the recursive function");
    total = 1; x = 1;
    for ( ; x<=10; ) { total = total * x; x = x + 1; }
    println ("iterative result of 10! is ", total);
}
class cs411 {
    int Funny;
    double funny;
    boolean flag;
    string s;
    int [] a;
// Able to accept everything above this :c
    flag = true;
    Funny = 0X89aB; funny = 123456E+7;
```

```
    s = "hello world";
    while (x = (Funny/10) <0) println (s, " have fun !");
    a = newarray (20, int);
}
```

Output: P1Example.out

```
int [shift]
[reduce 05.1]id [shift]
([shift]
int [shift]
[reduce 05.1]id [shift]
[reduce 04.1])[reduce 07.1.2][reduce 07.1][shift]
{[shift]
[reduce 12.1.1.1]if [reduce 12.1.2.1][shift]
([shift]
id [shift]
> [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 20.14][shift]
return [shift]
id [shift]
mult [reduce 21.1][reduce 20.03][shift]
id [shift]
([shift]
id [shift]
sub [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 20.07][reduce 26.1][reduce 23.2][shift]
[reduce 22.1][reduce 20.04][reduce 20.08]; [reduce 25.2][shift]
[reduce 18.1][reduce 13.6]else [shift]
return [shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 25.2][shift]
[reduce 18.1][reduce 13.6][reduce 14.2][reduce 13.2][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 06.1][reduce 02.2][reduce 01.1]void [shift]
id [shift]
([shift]
)[reduce 07.2][shift]
{[shift]
[reduce 12.1.1.1]int [shift]
[reduce 05.1]id [shift]
[reduce 04.1]; [shift]
[reduce 03.1][reduce 12.1.1.2]int [shift]
[reduce 05.1]id [shift]
[reduce 04.1]; [shift]
[reduce 03.1][reduce 12.1.1.2]println [reduce 12.1.2.1][shift]
([shift]
stringconst [shift]
[reduce 24.3][reduce 20.02], [shift]
id [shift]
([shift]
intconst [shift]
[reduce 24.1][reduce 20.02])[reduce 26.1][reduce 23.2][shift]
[reduce 22.1][reduce 20.04], [shift]
stringconst [shift]
[reduce 24.3][reduce 20.02])[reduce 26.1][reduce 26.2][reduce 26.2][shift]
```

```
; [shift]
[reduce 19.1][reduce 13.7][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]for [shift]
([shift]
; [reduce 25.1][shift]
id [shift]
<= [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.13][shift]
)[reduce 25.1][shift]
{[shift]
[reduce 12.1.1.1]id [reduce 12.1.2.1][shift]
= [reduce 21.1][shift]
id [shift]
mult [reduce 21.1][reduce 20.03][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.08][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.06][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 13.8][reduce 16.1][reduce 13.4][reduce 12.1.2.2]println [shift]
([shift]
stringconst [shift]
[reduce 24.3][reduce 20.02], [shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 26.1][reduce 26.2][shift]
; [shift]
[reduce 19.1][reduce 13.7][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 06.2][reduce 02.2][reduce 01.2]class [shift]
id [shift]
{[reduce 08.1.1.1][reduce 08.1.2.1][shift]
[reduce 08.1.3.1]int [shift]
[reduce 05.1]id [shift]
; [reduce 04.1][shift]
[reduce 03.1][reduce 09.1][reduce 08.1.3.2]double [shift]
[reduce 05.2]id [shift]
; [reduce 04.1][shift]
[reduce 03.1][reduce 09.1][reduce 08.1.3.2]boolean [shift]
[reduce 05.3]id [shift]
; [reduce 04.1][shift]
[reduce 03.1][reduce 09.1][reduce 08.1.3.2]string [shift]
[reduce 05.4]id [shift]
; [reduce 04.1][shift]
[reduce 03.1][reduce 09.1][reduce 08.1.3.2]int [shift]
```

```
[reduce 05.1][[shift]
][shift]
[reduce 05.5]id [shift]
; [reduce 04.1][shift]
[reduce 03.1][reduce 09.1][reduce 08.1.3.2]id [shift]
[reduce 05.6]= [reject]
```

Input: Palindrome.toy

```
boolean checkPalindrome(string s)
{
   int[] asciiS;
   int i;
   int j;
   asciiS = s.toASCIIArray();
   i = 0;
   j = s.getLength();
   while (i != j)
   {
      if (asciiS[i] != asciiS[j])
      {
         return false;
      }
      i = i + 1;
      j = j - 1;
   }
   return true;
}
```

Output: PalindromeOut.txt

```
boolean [shift]
[reduce 05.3]id [shift]
([shift]
string [shift]
[reduce 05.4]id [shift]
[reduce 04.1])[reduce 07.1.2][reduce 07.1][shift]
{[shift]
[reduce 12.1.1.1]int [shift]
[reduce 05.1][[shift]
][shift]
[reduce 05.5]id [shift]
[reduce 04.1]; [shift]
[reduce 03.1][reduce 12.1.1.2]int [shift]
[reduce 05.1]id [shift]
[reduce 04.1]; [shift]
[reduce 03.1][reduce 12.1.1.2]int [shift]
[reduce 05.1]id [shift]
[reduce 04.1]; [shift]
[reduce 03.1][reduce 12.1.1.2]id [reduce 12.1.2.1][shift]
= [reduce 21.1][shift]
id [shift]
. [shift]
id [shift]
([shift]
)[reduce 23.1][shift]
[reduce 22.2][reduce 20.04]; [reduce 20.01][shift]
```

```
[reduce 13.1][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
id [shift]
. [shift]
id [shift]
([shift]
)[reduce 23.1][shift]
[reduce 22.2][reduce 20.04]; [reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]while [shift]
([shift]
id [shift]
!= [reduce 21.1][reduce 20.03][shift]
id [shift]
)[reduce 21.1][reduce 20.03][reduce 20.17][shift]
{[shift]
[reduce 12.1.1.1]if [reduce 12.1.2.1][shift]
([shift]
id [shift]
[[reduce 21.1][shift]
id [shift]
][reduce 21.1][reduce 20.03][shift]
[reduce 21.2]!= [reduce 20.03][shift]
id [shift]
[[reduce 21.1][shift]
id [shift]
][reduce 21.1][reduce 20.03][shift]
[reduce 21.2])[reduce 20.03][reduce 20.17][shift]
{[shift]
[reduce 12.1.1.1]return [reduce 12.1.2.1][shift]
booleanconstF [shift]
[reduce 24.4][reduce 20.02]; [reduce 25.2][shift]
[reduce 18.1][reduce 13.6][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 13.8]id [reduce 14.1][reduce 13.2][reduce 12.1.2.2][shift]
= [reduce 21.1][shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.06][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
id [shift]
sub [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.07][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 13.8][reduce 15.1][reduce 13.3][reduce 12.1.2.2]return [shift]
booleanconstT [shift]
[reduce 24.4][reduce 20.02]; [reduce 25.2][shift]
[reduce 18.1][reduce 13.6][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 06.1][reduce 02.2][reduce 01.1][shift]
[accept]
```

Input: simpleSum.toy

```
// A test program.
void main()
{
   int sum;
   int a;
   a = 2;
   sum = a + 3;
}
```

Output: SimpleSumOut.txt

```
void [shift]
id [shift]
([shift]
)[reduce 07.2][shift]
{[shift]
[reduce 12.1.1.1]int [shift]
[reduce 05.1]id [shift]
[reduce 04.1]; [shift]
[reduce 03.1][reduce 12.1.1.2]int [shift]
[reduce 05.1]id [shift]
[reduce 04.1]; [shift]
[reduce 03.1][reduce 12.1.1.2]id [reduce 12.1.2.1][shift]
= [reduce 21.1][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]id [shift]
= [reduce 21.1][shift]
id [shift]
add [reduce 21.1][reduce 20.03][shift]
intconst [shift]
[reduce 24.1][reduce 20.02]; [reduce 20.06][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 06.2][reduce 02.2][reduce 01.1][shift]
[accept]
```

Input: Square.toy

```
// A Toy Program that should be accepted
/*
Program => Program Decl
   Program => Program Decl
      Decl => ClassDecl
         ClassDecl => _class _id _extends _id { OptionalField }
            Field => VariableDecl
               VariableDecl => Variable _semicolon
                  Variable => Type _id
                     Type => _double
            Field => FunctionDecl
               FunctionalDecl => Type _id _leftparen Formals _rightparen StmtBlock
                  Type => _id
                  Formals => VarList
                     VarList => Variable
                        Variable => Type _id
                           Type => _double
```

```
                  StmtBlock => _leftbrace StmtVarDecl StmtBlockStmt _rightbrace
                     StmtVarDecl => empty
                     StmtBlockStmt => Stmt StmtBlockStmt
                        Stmt => OptionalExpr _semicolon
                           OptionalExpr => Expr
                              Expr => Lvalue _assignop Expr
                                 Lvalue => _id
                                 Expr => Lvalue
                                    Lvalue => _id
                        StmtBlockStmt => empty
            Field => FunctionDecl
               FunctionDecl => Type _id _leftparen Formals _rightparen StmtBlock
                  Type => _double
                  Formals => empty
                  StmtBlock => _leftbrace StmtVarDecl StmtBlockStmt _rightbrace
                     StmtVarDecl => VariableDecl StmtVarDecl
                        VariableDecl => Variable _semicolon
                           Variable => Type _id
                              Type => _double
                        StmtVarDecl => empty
                     StmtBlockStmt => Stmt StmtBlockStmt
                        Stmt => OptionalExpr _semicolon
                           OptionalExpr => Expr
                              Expr => Lvalue _assignop Expr
                                 Lvalue => _id
                                 Expr => Expr _mult Expr
                                    Expr => Lvalue
                                       Lvalue => _id
                                    Expr => Lvalue
                                       Lvalue => _id
                        StmtBlockStmt => Stmt StmtBlockStmt
                           Stmt => ReturnStmt
                              ReturnStmt => _return OptionalExpr _semicolon
                                 OptionalExpr => Expr
                                    Expr => Lvalue
                                       Lvalue => _id
                           StmtBlockStmt => empty
It was very much not fun to do that, and I apologize if there are mistakes.
*/
class Square extends Shape implements Geometry
{
   double side;

   Square Square(double s)
   {
      side = s; //rejected
   }

   double getArea()
   {
      double area;
      area = side * side;
      return area;
   }
}
```

Output: SquareOut.txt

```
class [shift]
id [shift]
extends [shift]
id [shift]
[reduce 08.1.1.2]implements [shift]
id [shift]
{[reduce 08.1.2.2.2][reduce 08.1.2.2][shift]
[reduce 08.1.3.1]double [shift]
[reduce 05.2]id [shift]
; [reduce 04.1][shift]
[reduce 03.1][reduce 09.1][reduce 08.1.3.2]id [shift]
[reduce 05.6]id [shift]
([shift]
double [shift]
[reduce 05.2]id [shift]
[reduce 04.1])[reduce 07.1.2][reduce 07.1][shift]
{[shift]
[reduce 12.1.1.1]id [reduce 12.1.2.1][shift]
= [reduce 21.1][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 06.1][reduce 09.2][reduce 08.1.3.2]double [shift]
[reduce 05.2]id [shift]
([shift]
)[reduce 07.2][shift]
{[shift]
[reduce 12.1.1.1]double [shift]
[reduce 05.2]id [shift]
[reduce 04.1]; [shift]
[reduce 03.1][reduce 12.1.1.2]id [reduce 12.1.2.1][shift]
= [reduce 21.1][shift]
id [shift]
mult [reduce 21.1][reduce 20.03][shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 20.08][reduce 20.01][shift]
[reduce 13.1][reduce 12.1.2.2]return [shift]
id [shift]
; [reduce 21.1][reduce 20.03][reduce 25.2][shift]
[reduce 18.1][reduce 13.6][reduce 12.1.2.2]}[shift]
[reduce 12.1][reduce 06.1][reduce 09.2][reduce 08.1.3.2]}[shift]
[reduce 08.1][reduce 02.3][reduce 01.1][shift]
[accept]
```

Input: Tiny.toy

If it wasn't obvious already, this one was the very first test file I used, and then I mostly used Square.toy.

```
// Program => Decl => FunctionDecl => void main(Formals) StmtBlock
// Formals => empty [reduce 07.2]
// StmtBlock => { StmtVarDecl StmtBlockStmt }
// StmtVarDecl => empty [reduce 12.1.1.1]
// StmtBlockStmt => empty [reduce 12.1.2.1]
// r07.2 r12.1.1.1 r12.1.2.1 r12.1 r06.2 r02.2 r01.1
```

```
void main() {}
```

Output: TinyOut.txt

```
void [shift]
id [shift]
([shift]
)[reduce 07.2][shift]
{[shift]
[reduce 12.1.1.1]}[reduce 12.1.2.1][shift]
[reduce 12.1][reduce 06.2][reduce 02.2][reduce 01.1][shift]
[accept]
```

Production Rules

Number	Left	Right
01.1	*Program*	*Decl*
01.2		*Program Decl*
02.1	*Decl*	*VariableDecl*
02.2		*FunctionDecl*
02.3		*ClassDecl*
02.4		*InterfaceDecl*
03.1	*VariableDecl*	*Variable* _semicolon
04.1	*Variable*	*Type* _id
05.1	*Type*	_int
05.2		_double
05.3		_boolean
05.4		_string
05.5		*Type* _leftbracket _rightbracket
05.6		_id
06.1	*FunctionDecl*	*Type* _id _leftparen *Formals* _rightparen StmtBlock
06.2		_void _id _leftparen *Formals* _rightparen StmtBlock
07.1	*Formals*	*VarList*
07.2		ε
07.1.1	*VarList*	*Variable* _comma *VarList*
07.1.2		*Variable*
08.1	*ClassDecl*	_class _id *Extension Implementation* _leftbrace *OptionalField* _rightbrace
08.1.1.1	*Extension*	ε
08.1.1.2		_extends _id
08.1.2.1	*Implementation*	ε
08.1.2.2		_implements *IDList*
08.1.2.2.1	*IDList*	_id _comma *IDList*
08.1.2.2.2		_id
08.1.3.1	*OptionalField*	ε
08.1.3.2		*OptionalField Field*
09.1	*Field*	*VariableDecl*
09.2		*FunctionDecl*
10.1	*InterfaceDecl*	_interface _id _leftbrace *Prototype* _rightbrace
11.1	*Prototype*	ε
11.2		*Type* _id _leftparen *Formals* _rightparen _semicolon
11.3		_void _id _leftparen *Formals* _rightparen _semicolon
12.1	*StmtBlock*	_leftbrace *StmtVarDecl StmtBlockStmt* _rightbrace
12.1.1.1	*StmtVarDecl*	ε
12.1.1.2		*StmtVarDecl VariableDecl*
12.1.2.1	*StmtBlockStmt*	ε
12.1.2.2		*StmtBlockStmt Stmt*
13.0	*Stmt*	_semicolon
13.1		*Expr* _semicolon
13.2		*IfStmt*
13.3		*WhileStmt*
13.4		*ForStmt*
13.5		*BreakStmt*
13.6		*ReturnStmt*
13.7		*PrintStmt*
13.8		*StmtBlock*

25.1	*OptionalExpr*	ε
25.2		*Expr*
14.1	*IfStmt*	_if _leftparen *Expr* _rightparen *Stmt*
14.2		_if _leftparen *Expr* _rightparen *Stmt* _else *Stmt*
15.1	*WhileStmt*	_while _leftparen *Expr* _rightparen *Stmt*
16.1	*ForStmt*	_for _leftparen *OptionalExpr* _semicolon *Expr* _semicolon *OptionalExpr* _rightparen *Stmt*
17.1	*BreakStmt*	_break _semicolon
18.1	*ReturnStmt*	_return *OptionalExpr* _semicolon
19.1	*PrintStmt*	_println _leftparen *ExprList* _rightparen _semicolon
26.1	*ExprList*	*Expr*
26.2		*Expr* _comma *ExprList*
20.01	*Expr*	*Lvalue* _assignop *Expr*
20.02		*Constant*
20.03		*Lvalue*
20.04		*Call*
20.05		_leftparen *Expr* _rightparen
20.06		*Expr* _add *Expr*
20.07		*Expr* _sub *Expr*
20.08		*Expr* _mult *Expr*
20.09		*Expr* _div *Expr*
20.10		*Expr* _mod *Expr*
20.11		_sub *Expr*
20.12		*Expr* _less *Expr*
20.13		*Expr* _lessequal *Expr*
20.14		*Expr* _greater *Expr*
20.15		*Expr* _greaterequal *Expr*
20.16		*Expr* _equal *Expr*
20.17		*Expr* _notequal *Expr*
20.18		*Expr* _and *Expr*
20.19		*Expr* _or *Expr*
20.20		_not *Expr*
20.21		_readln _leftparen _rightparen
20.22		_newarray _leftparen _intconstant _comma *Type* _rightparen
21.1	*Lvalue*	_id
21.2		*Lvalue* _leftbracket *Expr* _rightbracket
21.3		*Lvalue* _period _id
22.1	*Call*	_id _leftparen *Actuals* _rightparen
22.2		_id _period _id _leftparen *Actuals* _rightparen
23.1	*Actuals*	ε
23.2		*ExprList*
24.1	*Constant*	_intconstant
24.2		_doubleconstant
24.3		_stringconstant
24.4		_booleanconstant

ADJUSTED – Changed some rules
InterfaceDecl: // 10
    _interface _id _leftbrace Prototypes _rightbrace

```
        {printf("[reduce 10.1]");}
;

Prototypes:
// Prototypes* from InterfaceDecl
    /* empty */
        {printf("[reduce 10.1.1]");}
    | Prototypes Prototype
        {printf("[reduce 10.1.2]");}
;

Prototype: // 11
    Type _id _leftparen Formals _rightparen _semicolon
        {printf("[reduce 11.2]");}
    | _void _id _leftparen Formals _rightparen _semicolon
        {printf("[reduce 11.3]");}
;
```