```
joshua@192:~/Documents/Code/splang/midreport/examplePrograms (master)$ cat examplePrograms.log
*** Compiling ./examplePrograms/2D.spl ***
//Jan  Groothuijse (e727426)
Int foo (Int n)
{
        return (2, 2);
}

(Int, Int) transpose ((Int, Int) p1, (Int, Int) p2) {
        return ((fst(p1) + fst(p2)), (snd(p1) + snd(p2)));
}

(Int, Int) scale((Int, Int) p, Int scalar) {
        return (fst(p) * scalar, snd(p) * scalar);
}

*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
./examplePrograms/2D.spl:2:1: error: Cannot unify types Int and (Int, Int)
Int foo (Int n)
^
./examplePrograms/2D.spl:2:1: note: Type Int inferred here:
Int foo (Int n)
^~~
./examplePrograms/2D.spl:4:9: note: Type (Int, Int) inferred here:
        return (2, 2);
               ^~~~~
./examplePrograms/2D.spl:2:10: error: Type mismatch. Expected type Int declared here:
Int foo (Int n)
         ^~~
./examplePrograms/2D.spl:2:1: note: Actual type a7 inferred here:
Int foo (Int n)
^
./examplePrograms/2D.spl:2:1: error: Type mismatch. Expected type Int declared here:
Int foo (Int n)
^~~
./examplePrograms/2D.spl:4:9: note: Actual type (Int, Int) inferred here:
        return (2, 2);
               ^~~~~
*** Typing failed (but we try to continue!)
// foo :: Int(Int)
Int foo/*6*/(Int n/*9*/){
        return (2, 2);
}

// transpose :: (Int, Int)((Int, Int), (Int, Int))
(Int, Int) transpose/*7*/((Int, Int) p1/*10*/, (Int, Int) p2/*11*/){
        return (fst/*4*/(p1/*10*/) + fst/*4*/(p2/*11*/), snd/*5*/(p1/*10*/) + snd/*5*/(p2/*11*/));
}

// scale :: (Int, Int)((Int, Int), Int)
(Int, Int) scale/*8*/((Int, Int) p/*12*/, Int scalar/*13*/){
        return (fst/*4*/(p/*12*/) * scalar/*13*/, snd/*5*/(p/*12*/) * scalar/*13*/);
}
*** Done


*** Compiling ./examplePrograms/3D.spl ***
//Jan  Groothuijse (e727426)
(Int, (Int, Int)) transpose ((Int, (Int, Int))  p1, (Int, (Int, Int))  p2) {
        return ((fst(p1) + fst(p2)), (first(snd(p1)) + first(snd(p2)), snd(snd(p1)) + snd(snd(p2))));
}

(Int, Int) scale((Int, Int) p, Int scalar) {
        return (fst(p) * scalar, (first(snd(p)) * scalar, snd(snd(p)) * scalar));
}

*** Done reading file
*** Lexing is done
*** Parsing succeeded
./examplePrograms/3D.spl:3:32: error: Undeclared identifier "first"
        return ((fst(p1) + fst(p2)), (first(snd(p1)) + first(snd(p2)), snd(snd(p1)) + snd(snd(p2))));
                                     ^~~~~
./examplePrograms/3D.spl:3:32: note: Did you mean "fst"?
./examplePrograms/3D.spl:3:49: error: Undeclared identifier "first"
        return ((fst(p1) + fst(p2)), (first(snd(p1)) + first(snd(p2)), snd(snd(p1)) + snd(snd(p2))));
                                                    ^~~~~
./examplePrograms/3D.spl:3:49: note: Did you mean "fst"?
./examplePrograms/3D.spl:7:28: error: Undeclared identifier "first"
        return (fst(p) * scalar, (first(snd(p)) * scalar, snd(snd(p)) * scalar));
                                  ^~~~~
./examplePrograms/3D.spl:7:28: note: Did you mean "fst"?
*** Scoping failed (but we try to continue!)
./examplePrograms/3D.spl:3:32: error: Identifier is unknown: first
        return ((fst(p1) + fst(p2)), (first(snd(p1)) + first(snd(p2)), snd(snd(p1)) + snd(snd(p2))));
                                     ^~~~~
./examplePrograms/3D.spl:3:32: note: This is probably caused by earlier errors
*** Stopping due to fatal error


*** Compiling ./examplePrograms/Example.spl ***
/* Renze  Droog (s4290755)

        Length
   of
   months
*/

[Int] dcLengthOfMonth = 0 : 31 : 28 : 31 : 30 : 31 : 30 : 31 : 31 : 30 : 31 : 30 : 31 : [];


//Calculates in which month the given Int is. 1=January, 2=February, .., 12=December
Int month(Int dayOfYear){Int tempDay = dayOfYear;Int month = 0;
```

```
                        // unuseful comment

    while(month < 12)
    {
                                            if(tempDay > dcLengthOfMonth(month))
        {{{{
                        tempDay = tempDay - - - - - dcLengthOfMonth(month);
        }}}}
        else return month + 1;

       month = month + 1;
        }
return month + 1;
}
*** Done reading file
*** Lexing is done
*** Parsing succeeded
./examplePrograms/Example.spl:12:54: warning: "month" shadows previous declaration.
Int month(Int dayOfYear){Int tempDay = dayOfYear;Int month = 0;
                                                     ^~~~
./examplePrograms/Example.spl:12:5: note: Previous declaration was a global:
Int month(Int dayOfYear){Int tempDay = dayOfYear;Int month = 0;
    ^~~~
*** Scoping succeeded
./examplePrograms/Example.spl:23:50: error: dcLengthOfMonth used as function, but has type [Int]
                        tempDay = tempDay - - - - - dcLengthOfMonth(month);
                                                    ^~~~~~~~~~~~~~~~~~~~~~
./examplePrograms/Example.spl:23:50: error: Cannot unify types [Int] and a18(a19)
                        tempDay = tempDay - - - - - dcLengthOfMonth(month);
                                                    ^~~~~~~~~~~~~~~~~~~~~~
./examplePrograms/Example.spl:8:1: note: Type [Int] inferred here:
[Int] dcLengthOfMonth = 0 : 31 : 28 : 31 : 30 : 31 : 30 : 31 : 31 : 30 : 31 : 30 : 31 : [];
^~~~~
./examplePrograms/Example.spl:23:50: note: Type a18(a19) inferred here:
                        tempDay = tempDay - - - - - dcLengthOfMonth(month);
                                                    ^~~~~~~~~~~~~~~~~~~~~~
./examplePrograms/Example.spl:21:20: error: dcLengthOfMonth used as function, but has type [Int]
                                            if(tempDay > dcLengthOfMonth(month))
                                                         ^~~~~~~~~~~~~~~~~~~~~~
./examplePrograms/Example.spl:21:20: error: Cannot unify types [Int] and a21(a22)
                                            if(tempDay > dcLengthOfMonth(month))
                                                         ^~~~~~~~~~~~~~~~~~~~~~
./examplePrograms/Example.spl:8:1: note: Type [Int] inferred here:
[Int] dcLengthOfMonth = 0 : 31 : 28 : 31 : 30 : 31 : 30 : 31 : 31 : 30 : 31 : 30 : 31 : [];
^~~~~
./examplePrograms/Example.spl:21:20: note: Type a21(a22) inferred here:
                                            if(tempDay > dcLengthOfMonth(month))
                                                         ^~~~~~~~~~~~~~~~~~~~~~

*** Typing failed (but we try to continue!)
// dcLengthOfMonth :: [Int]
[Int] dcLengthOfMonth/*6*/ = 0 : (31 : (28 : (31 : (30 : (31 : (30 : (31 : (31 : (30 : (31 : (30 : (31 : [])))))))))))));

// month :: Int(Int)
Int month/*7*/(Int dayOfYear/*8*/){
        // tempDay :: Int
        Int tempDay/*9*/ = dayOfYear/*8*/;
        // month :: Int
        Int month/*10*/ = 0;
        while(month/*10*/ < 12){
                if(tempDay/*9*/ > dcLengthOfMonth/*6*/(month/*10*/)){
                        {
                                {
                                        {
                                                tempDay/*9*/ = tempDay/*9*/ - (-(-(-(-dcLengthOfMonth/*6*/(month/*10*/)))));
                                        }
                                }
                        }
                } else
                        return month/*10*/ + 1;
                month/*10*/ = month/*10*/ + 1;
        }
        return month/*10*/ + 1;
}
*** Done


*** Compiling ./examplePrograms/Fib.spl ***
/*
        Three ways to implement the factorial function in SPL.
        First the recursive version.
*/
Int fib(Int n)
{
        if ( n < 2 )
                return 1;
        else
                return fib(n-1) + fib(n-2);
}

*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
*** Typing succeeded
// fib :: Int(Int)
Int fib/*6*/(Int n/*7*/){
        if(n/*7*/ < 2)
                return 1;
        else
                return fib/*6*/(n/*7*/ - 1) + fib/*6*/(n/*7*/ - 2);
}
*** Done


*** Compiling ./examplePrograms/SumProduct.spl ***
//Jan  Groothuijse (e727426)
// SPL has overloading?
```

```
Int sum([Int] list) {
        if (isEmpty(list)) return 0;
        return head(list) + sum(tail(list));
}

Int product([Int] list) {
        if (isEmpty(list)) return 1;
        return head(list) * sum(tail(list));
}

Bool sum([Bool] list) {
        if (isEmpty(list)) return False;
        return head(list) || sum(tail(list));
}

Bool product([Bool] list) {
        if (isEmpty(list)) return True;
        return head(list) && sum(tail(list));
}
```

```
*** Done reading file
*** Lexing is done
*** Parsing succeeded
./examplePrograms/SumProduct.spl:14:6: error: Redeclaration of identifier "sum"
Bool sum([Bool] list) {
     ^~~
./examplePrograms/SumProduct.spl:4:5: note: Previous declaration here:
Int sum([Int] list) {
    ^~~
./examplePrograms/SumProduct.spl:19:6: error: Redeclaration of identifier "product"
Bool product([Bool] list) {
     ^~~~~~~
./examplePrograms/SumProduct.spl:9:5: note: Previous declaration here:
Int product([Int] list) {
    ^~~~~~~
*** Scoping failed (but we try to continue!)
./examplePrograms/SumProduct.spl:14:6: error: Identifier is unknown: sum
Bool sum([Bool] list) {
     ^~~
./examplePrograms/SumProduct.spl:14:6: note: This is probably caused by earlier errors
*** Stopping due to fatal error
```

```
*** Compiling ./examplePrograms/arguments.spl ***
//Tim  Steenvoorden (s0712663)

[Int] function (Int n, Bool b, [Int] l)
{
    print(n);
    print(b);
    print(l);
}

Void main ()
{
    [Int] list = 1:2:3:[];

    // No arguments
    function();
    // One argument
    function(5);
    // Two arguments
    function(10, True);
    // Three arguments
    function(15, False, list);
}
```

```
*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
./examplePrograms/arguments.spl:3:1: error: Cannot unify types [Int] and Void
[Int] function (Int n, Bool b, [Int] l)
^
./examplePrograms/arguments.spl:3:1: note: Type [Int] inferred here:
[Int] function (Int n, Bool b, [Int] l)
^~~~~
./examplePrograms/arguments.spl:3:1: note: Type Void inferred here:
[Int] function (Int n, Bool b, [Int] l)
^
./examplePrograms/arguments.spl:3:17: error: Type mismatch. Expected type Int declared here:
[Int] function (Int n, Bool b, [Int] l)
                ^~~
./examplePrograms/arguments.spl:3:1: note: Actual type a17 inferred here:
[Int] function (Int n, Bool b, [Int] l)
^
./examplePrograms/arguments.spl:3:24: error: Type mismatch. Expected type Bool declared here:
[Int] function (Int n, Bool b, [Int] l)
                       ^~~~
./examplePrograms/arguments.spl:3:1: note: Actual type a18 inferred here:
[Int] function (Int n, Bool b, [Int] l)
^
./examplePrograms/arguments.spl:3:32: error: Type mismatch. Expected type [Int] declared here:
[Int] function (Int n, Bool b, [Int] l)
                               ^~~~~
./examplePrograms/arguments.spl:3:1: note: Actual type a19 inferred here:
[Int] function (Int n, Bool b, [Int] l)
^
./examplePrograms/arguments.spl:3:1: error: Type mismatch. Expected type [Int] declared here:
[Int] function (Int n, Bool b, [Int] l)
^~~~~
./examplePrograms/arguments.spl:3:1: note: Actual type Void inferred here:
[Int] function (Int n, Bool b, [Int] l)
^
./examplePrograms/arguments.spl:15:5: error: Too few arguments given, 0 given, but 3 expected.
    function();
```

```
    ^~~~~~~~~~
./examplePrograms/arguments.spl:17:5: error: Too few arguments given, 1 given, but 3 expected.
    function(5);
    ^~~~~~~~~~
./examplePrograms/arguments.spl:19:5: error: Too few arguments given, 2 given, but 3 expected.
    function(10, True);
    ^~~~~~~~~~~~~~~~~~
*** Typing failed (but we try to continue!)
// function :: [Int](Int, Bool, [Int])
[Int] function/*6*/(Int n/*8*/, Bool b/*9*/, [Int] l/*10*/){
        print/*0*/(n/*8*/);
        print/*0*/(b/*9*/);
        print/*0*/(l/*10*/);
}

// main :: Void()
Void main/*7*/(){
        // list :: [Int]
        [Int] list/*11*/ = 1 : (2 : (3 : []));
        function/*6*/();
        function/*6*/(5);
        function/*6*/(10, True);
        function/*6*/(15, False, list/*11*/);
}
*** Done


*** Compiling ./examplePrograms/bool.spl ***
//Jan  Groothuijse (e727426)
Bool xor(Bool a, Bool b) {
        return (a || b) && !(a && b);
}

Bool implies(Bool a, Bool b) {
        if (!a) return True;
        else return b;
}

*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
*** Typing succeeded
// xor :: Bool(Bool, Bool)
Bool xor/*6*/(Bool a/*8*/, Bool b/*9*/){
        return (a/*8*/ || b/*9*/) && (!(a/*8*/ && b/*9*/));
}

// implies :: Bool(Bool, Bool)
Bool implies/*7*/(Bool a/*10*/, Bool b/*11*/){
        if(!a/*10*/)
                return True;
        else
                return b/*11*/;
}
*** Done


*** Compiling ./examplePrograms/booleans.spl ***
//Tim  Steenvoorden (s0712663)

Int   int1  = -3;
Bool  bool1 = True;
Bool  bool2 = False;
[Int] list  = [];

Void mainfunction()
{   Int x = 0;
    someType st = null;
    x = x+10;
    if( x <10) function1(st, False);
    else function1(st, True);
}

// Some bla bla bla
Void function1 (someType st, Bool b)
{
    if (bool1 && bool2 || bool2)
        int1 = 3 - 2 + 1;
    else
        int1 = 4 * 3 / 2 % 1;
}

/* Some more
bla bla
bla
*/
(Bool, Int) function2 ()
{
    while ( int1 > 0 || int1 < 1 &&
            int1 <= 0 || int1 >= 1 &&
            int1 != -6)
    {
        list = int1:list;
    }
    return (!bool2, someFunction());
}

// The difference between True and true, False and false, Int and int etc.
Void integer (int x, int bools)
{
    id = id;
    boolean = null;

}
Bool z = true;
Bool w = False;
Void boolean ()
```

```
{
Bool b= true;
b = fasle;
}
Bool z = true;
Bool w = False;
Void boolean ()
{
Bool b= true;
b = fasle;
}


*** Done reading file
*** Lexing is done
*** Parsing succeeded
./examplePrograms/booleans.spl:54:6: error: Redeclaration of identifier "z"
Bool z = true;
     ^
./examplePrograms/booleans.spl:47:6: note: Previous declaration here:
Bool z = true;
     ^
./examplePrograms/booleans.spl:55:6: error: Redeclaration of identifier "w"
Bool w = False;
     ^
./examplePrograms/booleans.spl:48:6: note: Previous declaration here:
Bool w = False;
     ^
./examplePrograms/booleans.spl:56:6: error: Redeclaration of identifier "boolean"
Void boolean ()
     ^~~~~~~
./examplePrograms/booleans.spl:49:6: note: Previous declaration here:
Void boolean ()
     ^~~~~~~
./examplePrograms/booleans.spl:10:19: error: Undeclared identifier "null"
    someType st = null;
                  ^~~~
./examplePrograms/booleans.spl:10:19: note: Did you mean "tail"?
./examplePrograms/booleans.spl:37:21: error: Undeclared identifier "someFunction"
    return (!bool2, someFunction());
                    ^~~~~~~~~~~~
./examplePrograms/booleans.spl:43:10: error: Undeclared identifier "id"
    id = id;
         ^~
./examplePrograms/booleans.spl:41:19: note: Did you mean "x"?
Void integer (int x, int bools)
                  ^
./examplePrograms/booleans.spl:43:5: error: Undeclared identifier "id"
    id = id;
    ^~
./examplePrograms/booleans.spl:41:19: note: Did you mean "x"?
Void integer (int x, int bools)
                  ^
./examplePrograms/booleans.spl:44:15: error: Undeclared identifier "null"
    boolean = null;
              ^~~~
./examplePrograms/booleans.spl:44:15: note: Did you mean "tail"?
./examplePrograms/booleans.spl:47:10: error: Undeclared identifier "true"
Bool z = true;
         ^~~~
./examplePrograms/booleans.spl:47:10: note: Did you mean "tail"?
./examplePrograms/booleans.spl:51:9: error: Undeclared identifier "true"
Bool b= true;
        ^~~~
./examplePrograms/booleans.spl:51:9: note: Did you mean "tail"?
./examplePrograms/booleans.spl:52:5: error: Undeclared identifier "fasle"
b = fasle;
    ^~~~~
./examplePrograms/booleans.spl:52:5: note: Did you mean "tail"?
./examplePrograms/booleans.spl:54:10: error: Undeclared identifier "true"
Bool z = true;
         ^~~~
./examplePrograms/booleans.spl:54:10: note: Did you mean "tail"?
./examplePrograms/booleans.spl:58:9: error: Undeclared identifier "true"
Bool b= true;
        ^~~~
./examplePrograms/booleans.spl:58:9: note: Did you mean "tail"?
./examplePrograms/booleans.spl:59:5: error: Undeclared identifier "fasle"
b = fasle;
    ^~~~~
./examplePrograms/booleans.spl:59:5: note: Did you mean "tail"?
*** Scoping failed (but we try to continue!)
./examplePrograms/booleans.spl:54:6: error: Identifier is unknown: z
Bool z = true;
     ^
./examplePrograms/booleans.spl:54:6: note: This is probably caused by earlier errors
*** Stopping due to fatal error


*** Compiling ./examplePrograms/empty.spl ***
//Tim  Steenvoorden (s0712663)

// Een functiebody kan niet leeg zijn, dus dit kan niet geparseerd worden.
Void main ()
{
}


*** Done reading file
*** Lexing is done
./examplePrograms/empty.spl:6:1: error: Unexpected token CurlyBracketClose
}
^
*** Stopping due to fatal error


*** Compiling ./examplePrograms/example10.spl ***
```

```
// Yannic  Smeets (s4244249)

Int int1 = -3;
Bool bool1 = True;
Bool bool2 = False;
[Int] list = [];

Void mainfunction()
{ Int x = 0;
        someType st = null;
{{{ x = x +10;
        if( x <10) function1(st,false);
        else function1(st,true);
}}}}

// Some bla bla bla
Void function1 (someType st, Bool b)
{
    if (bool1 && bool2 || bool2)
        int1 = 3 - 2 + 1;
    else
        int1 = 4 * 3 / 2 % 1;
}

/* Some more
bla bla
bla
*/
(Bool, Int) function2 ()
{
    while ( int1 > 0 || int1 < 1 &&
            int1 <= 0 || int1 >= 1 &&
            int1 != -6)
    {
        list = int1:list;
    }
    return (!bool2, someFunction());
}


*** Done reading file
*** Lexing is done
*** Parsing succeeded
./examplePrograms/example10.spl:10:16: error: Undeclared identifier "null"
        someType st = null;
                      ^~~~
./examplePrograms/example10.spl:10:16: note: Did you mean "tail"?
./examplePrograms/example10.spl:12:26: error: Undeclared identifier "false"
        if( x <10) function1(st,false);
                             ^~~~~
./examplePrograms/example10.spl:12:26: note: Did you mean "fst"?
./examplePrograms/example10.spl:13:20: error: Undeclared identifier "true"
        else function1(st,true);
                          ^~~~
./examplePrograms/example10.spl:13:20: note: Did you mean "tail"?
./examplePrograms/example10.spl:37:21: error: Undeclared identifier "someFunction"
    return (!bool2, someFunction());
                    ^~~~~~~~~~~~
*** Scoping failed (but we try to continue!)
./examplePrograms/example10.spl:10:16: error: Identifier is unknown: null
        someType st = null;
                      ^~~~
./examplePrograms/example10.spl:10:16: note: This is probably caused by earlier errors
*** Stopping due to fatal error


*** Compiling ./examplePrograms/factorial.spl ***
/*
 * Three ways to implement the factorial function in SPL.
 */

// First the recursive version.
Int facR (Int n) {
  if (n < 2)
    return 1;
  else
    return n * facR(n - 1);
}

// The iterative version of the factorial function
Int facI (Int n) {
  Int r = 1;

  while (n > 1) {
    r = r * n;
    n = n - 1;
  }

  return r;
}

// A main function to check the results
Void main () {
  Int n    = 0;
  Int facN = 1;
  Bool ok  = True;

  while (n < 20) {
    facN = facR(n);
    if (facN != facI(n) || facN != facL(n)) {
      print(n : facN : facI(n) : facL(n) : []);
      ok = False;
    }
    n = n + 1;
  }

  print(ok);
}
```

```
// A list based factorial function
// Defined here to show that functions can be given in any order (unlike C)
Int facL (Int n) {
  return product(fromTo(1, n));
}

// Computes the product of a list of integers
Int product ([Int] list) {
  if (isEmpty(list))
    return 1;
  else
    return head(list) * product(tail(list));
}

// Generates a list of integers from the first to the last argument
[Int] fromTo (Int from, Int to) {
  if (from <= to)
    return from : fromTo(from + 1, to);
  else
    return [];
}

// ft=c
```

```
*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
*** Typing succeeded
// facR :: Int(Int)
Int facR/*6*/(Int n/*12*/){
        if(n/*12*/ < 2)
                return 1;
        else
                return n/*12*/ * facR/*6*/(n/*12*/ - 1);
}

// facI :: Int(Int)
Int facI/*7*/(Int n/*13*/){
        // r :: Int
        Int r/*14*/ = 1;
        while(n/*13*/ > 1){
                r/*14*/ = r/*14*/ * n/*13*/;
                n/*13*/ = n/*13*/ - 1;
        }
        return r/*14*/;
}

// main :: Void()
Void main/*8*/(){
        // n :: Int
        Int n/*15*/ = 0;
        // facN :: Int
        Int facN/*16*/ = 1;
        // ok :: Bool
        Bool ok/*17*/ = True;
        while(n/*15*/ < 20){
                facN/*16*/ = facR/*6*/(n/*15*/);
                if((facN/*16*/ != facI/*7*/(n/*15*/)) || (facN/*16*/ != facL/*9*/(n/*15*/))){
                        print/*0*/(n/*15*/ : (facN/*16*/ : (facI/*7*/(n/*15*/) : (facL/*9*/(n/*15*/) : []))));
                        ok/*17*/ = False;
                }
                n/*15*/ = n/*15*/ + 1;
        }
        print/*0*/(ok/*17*/);
}

// facL :: Int(Int)
Int facL/*9*/(Int n/*18*/){
        return product/*10*/(fromTo/*11*/(1, n/*18*/));
}

// product :: Int([Int])
Int product/*10*/([Int] list/*19*/){
        if(isEmpty/*1*/(list/*19*/))
                return 1;
        else
                return head/*2*/(list/*19*/) * product/*10*/(tail/*3*/(list/*19*/));
}

// fromTo :: [Int](Int, Int)
[Int] fromTo/*11*/(Int from/*20*/, Int to/*21*/){
        if(from/*20*/ <= to/*21*/)
                return from/*20*/ : fromTo/*11*/(from/*20*/ + 1, to/*21*/);
        else
                return [];
}
*** Done
```

```
*** Compiling ./examplePrograms/integers.spl ***
//Tim  Steenvoorden (s0712663)

// Absolute value, in an strange layout
Int abs (Int n) { if (n<0) return-n; else return n ; }

// vim: ft=c
```

```
*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
*** Typing succeeded
// abs :: Int(Int)
Int abs/*6*/(Int n/*7*/){
        if(n/*7*/ < 0)
                return -n/*7*/;
```

```
        else
                return n/*7*/;
}
*** Done


*** Compiling ./examplePrograms/list.spl ***
//Jan  Groothuijse (e727426)
Bool equals([a] a, [a] b) {
        if (isEmpty(a) && isEmpty(b)) return true;
        if (isEmpty(a) || isEmpty(b)) return false;
        return (head(a) == head(b)) && equals (tail(a), tail(b));
}

*** Done reading file
*** Lexing is done
*** Parsing succeeded
./examplePrograms/list.spl:3:39: error: Undeclared identifier "true"
        if (isEmpty(a) && isEmpty(b)) return true;
                                            ^~~~
./examplePrograms/list.spl:3:39: note: Did you mean "tail"?
./examplePrograms/list.spl:4:39: error: Undeclared identifier "false"
        if (isEmpty(a) || isEmpty(b)) return false;
                                            ^~~~~
./examplePrograms/list.spl:4:39: note: Did you mean "fst"?
*** Scoping failed (but we try to continue!)
./examplePrograms/list.spl:3:39: error: Identifier is unknown: true
        if (isEmpty(a) && isEmpty(b)) return true;
                                            ^~~~
./examplePrograms/list.spl:3:39: note: This is probably caused by earlier errors
*** Stopping due to fatal error


*** Compiling ./examplePrograms/lists.spl ***
//Tim  Steenvoorden (s0712663)

// Reverse any list
[t] reverse ([t] list) {
    [t] accu = [];
    while (!isEmpty(list)) {
        accu = head(list) : accu;
        list = tail(list);
    }
    return accu;
}

// Calculate the sum of a list of integers
Int sum ([Int] list)
{
    return head(list) + sum(tail(list));
}

// Calculate the product of a list of integers
Int product ([Int] list)
{
    return head(list) * product(tail(list));
}

Void main ()
{
    [Int] list = 1:3:5:[];
    print(sum(list));
    print(product(list));
}

// vim: ft=c

*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
*** Typing succeeded
// reverse :: forall a0 . [a0]([a0])
[t] reverse/*6*/([t] list/*10*/){
        // accu :: [a0]
        [t] accu/*11*/ = [];
        while(!isEmpty/*1*/(list/*10*/)){
                accu/*11*/ = head/*2*/(list/*10*/) : accu/*11*/;
                list/*10*/ = tail/*3*/(list/*10*/);
        }
        return accu/*11*/;
}

// sum :: Int([Int])
Int sum/*7*/([Int] list/*12*/){
        return head/*2*/(list/*12*/) + sum/*7*/(tail/*3*/(list/*12*/));
}

// product :: Int([Int])
Int product/*8*/([Int] list/*13*/){
        return head/*2*/(list/*13*/) * product/*8*/(tail/*3*/(list/*13*/));
}

// main :: Void()
Void main/*9*/(){
        // list :: [Int]
        [Int] list/*14*/ = 1 : (3 : (5 : []));
        print/*0*/(sum/*7*/(list/*14*/));
        print/*0*/(product/*8*/(list/*14*/));
}
*** Done


*** Compiling ./examplePrograms/mergeSort.spl ***
//Jascha  Neutelings (s0610054)
Int length([t] list)
{
    if (isEmpty(list))
```

```
        return 0;
    else
        return 1 + length(tail(list));
}

[t] append([t] list, t value)
{
    if (isEmpty(list))
        return value : [];
    else
        return head(list) : append(tail(list), value);
}

([t],[t]) split_aux([t] list, Int n)
{
    ([t],[t]) tl = split(tail(list), n - 1);
    return (head(list) : fst(tl),snd(tl));
}

([t],[t]) split([t] list, Int n)
{
    if (isEmpty(list) || n == 0)
        return ([], list);
    else
        return split_aux(list, n);
}

[Int] merge_sort_aux([Int] m)
{
    Int middle = length(m) / 2;
    ([Int],[Int]) parts = split(m, middle);
    [Int] left = fst(parts);
    [Int] right = snd(parts);

    // recursively call merge_sort() to further split each sublist
    // until sublist size is 1
    left = merge_sort(left);
    right = merge_sort(right);
    // merge the sublists returned from prior calls to merge_sort()
    // and return the resulting merged sublist
    return merge(left, right);
}

[Int] merge_sort([Int] m)
{
    // if list size is 1, consider it sorted and return it
    if (isEmpty(m) || isEmpty(tail(m)))
        return m;
    // else list size is > 1, so split the list into two sublists
    return merge_sort_aux(m);
}

[Int] merge([Int] left, [Int] right)
{
    [Int] result = [];
    while (!isEmpty(left) || !isEmpty(right))
    {
        if (!isEmpty(left) && !isEmpty(right))
        {
            if (head(left) <= head(right))
            {
                result = append(result, head(left));
                left = tail(left);
            }
            else
            {
                result = append(result, head(right));
                right = tail(right);
            }
        }
        else if (!isEmpty(left))
        {
            result = append(result, head(left));
            left = tail(left);
        }
        else if (!isEmpty(right))
        {
            result = append(result, head(right));
            right = tail(right);
        }
    }
    return result;
}

Void main()
{
    [Int] list = 8 : 10 : 3 : 7 : 9 : 6 : 4 : 1 : 2 : 5 : [];
    print(merge_sort(list));
}
*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
*** Typing succeeded
// length :: forall a0 . Int([a0])
Int length/*6*/([t] list/*14*/){
        if(isEmpty/*1*/(list/*14*/))
                return 0;
        else
                return 1 + length/*6*/(tail/*3*/(list/*14*/));
}

// append :: forall a1 . [a1]([a1], a1)
[t] append/*7*/([t] list/*15*/, t value/*16*/){
        if(isEmpty/*1*/(list/*15*/))
                return value/*16*/ : [];
        else
                return head/*2*/(list/*15*/) : append/*7*/(tail/*3*/(list/*15*/), value/*16*/);
```

```
}

// split_aux :: forall a2 . ([a2], [a2])([a2], Int)
([t], [t]) split_aux/*8*/([t] list/*17*/, Int n/*18*/){
        // tl :: ([a2], [a2])
        ([t], [t]) tl/*19*/ = split/*9*/(tail/*3*/(list/*17*/), n/*18*/ - 1);
        return (head/*2*/(list/*17*/) : fst/*4*/(tl/*19*/), snd/*5*/(tl/*19*/));
}

// split :: forall a3 . ([a3], [a3])([a3], Int)
([t], [t]) split/*9*/([t] list/*20*/, Int n/*21*/){
        if(isEmpty/*1*/(list/*20*/) || (n/*21*/ == 0))
                return ([], list/*20*/);
        else
                return split_aux/*8*/(list/*20*/, n/*21*/);
}

// merge_sort_aux :: [Int]([Int])
[Int] merge_sort_aux/*10*/([Int] m/*22*/){
        // middle :: Int
        Int middle/*23*/ = length/*6*/(m/*22*/) / 2;
        // parts :: ([Int], [Int])
        ([Int], [Int]) parts/*24*/ = split/*9*/(m/*22*/, middle/*23*/);
        // left :: [Int]
        [Int] left/*25*/ = fst/*4*/(parts/*24*/);
        // right :: [Int]
        [Int] right/*26*/ = snd/*5*/(parts/*24*/);
        left/*25*/ = merge_sort/*11*/(left/*25*/);
        right/*26*/ = merge_sort/*11*/(right/*26*/);
        return merge/*12*/(left/*25*/, right/*26*/);
}

// merge_sort :: [Int]([Int])
[Int] merge_sort/*11*/([Int] m/*27*/){
        if(isEmpty/*1*/(m/*27*/) || isEmpty/*1*/(tail/*3*/(m/*27*/)))
                return m/*27*/;
        return merge_sort_aux/*10*/(m/*27*/);
}

// merge :: [Int]([Int], [Int])
[Int] merge/*12*/([Int] left/*28*/, [Int] right/*29*/){
        // result :: [Int]
        [Int] result/*30*/ = [];
        while((!isEmpty/*1*/(left/*28*/)) || (!isEmpty/*1*/(right/*29*/))){
                if((!isEmpty/*1*/(left/*28*/)) && (!isEmpty/*1*/(right/*29*/))){
                        if(head/*2*/(left/*28*/) <= head/*2*/(right/*29*/)){
                                result/*30*/ = append/*7*/(result/*30*/, head/*2*/(left/*28*/));
                                left/*28*/ = tail/*3*/(left/*28*/);
                        } else {
                                result/*30*/ = append/*7*/(result/*30*/, head/*2*/(right/*29*/));
                                right/*29*/ = tail/*3*/(right/*29*/);
                        }
                } else
                        if(!isEmpty/*1*/(left/*28*/)){
                                result/*30*/ = append/*7*/(result/*30*/, head/*2*/(left/*28*/));
                                left/*28*/ = tail/*3*/(left/*28*/);
                        } else
                                if(!isEmpty/*1*/(right/*29*/)){
                                        result/*30*/ = append/*7*/(result/*30*/, head/*2*/(right/*29*/));
                                        right/*29*/ = tail/*3*/(right/*29*/);
                                }
        }
        return result/*30*/;
}

// main :: Void()
Void main/*13*/(){
        // list :: [Int]
        [Int] list/*31*/ = 8 : (10 : (3 : (7 : (9 : (6 : (4 : (1 : (2 : (5 : [])))))))));
        print/*0*/(merge_sort/*11*/(list/*31*/));
}
*** Done


*** Compiling ./examplePrograms/mklinik.spl ***
// Markus  Klinik (s4220315)
Int foobar()
{
  return ((((((((((-10))))))))));
}

*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
*** Typing succeeded
// foobar :: Int()
Int foobar/*6*/(){
        return -10;
}
*** Done


*** Compiling ./examplePrograms/op.spl ***
//Jan  Groothuijse (e727426)
Int n = 5;
Bool op = 1 + n/2 - n % 2 < n;
Bool op2 = ((1 + n)/2) - n % 2 < n;

*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
*** Typing succeeded
// n :: Int
Int n/*6*/ = 5;

// op :: Bool
```

```
Bool op/*7*/ = ((1 + (n/*6*/ / 2)) - (n/*6*/ % 2)) < n/*6*/;

// op2 :: Bool
Bool op2/*8*/ = (((1 + n/*6*/) / 2) - (n/*6*/ % 2)) < n/*6*/;
*** Done


*** Compiling ./examplePrograms/pass_nested_expr.spl ***
//Wouter  Geraedts (s0814857)

// Test: Should parse quickly (with monadic LL* parsers, this would be slow)

Void main()
{
        ((((((((((((((5))))))))))))));
}

*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
*** Typing succeeded
// main :: Void()
Void main/*6*/(){
        5;
}
*** Done


*** Compiling ./examplePrograms/problematic.spl ***
// Dennis  Brentjes (s0815489)

Int deducting_a_minus_id()
{
    return 1;
}

Void doing_just_a_funCall()
{
    print(4 - -deducting_a_minus_id);
}

Int main()
{
    doing_just_a_funCall();
}

*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
./examplePrograms/problematic.spl:10:16: error: Cannot unify types Int and Int()
    print(4 - -deducting_a_minus_id);
               ^~~~~~~~~~~~~~~~~~~~
./examplePrograms/problematic.spl:10:16: note: Type Int inferred here:
    print(4 - -deducting_a_minus_id);
               ^~~~~~~~~~~~~~~~~~~~
./examplePrograms/problematic.spl:3:1: note: Type Int() inferred here:
Int deducting_a_minus_id()
^
./examplePrograms/problematic.spl:13:1: error: Cannot unify types Int and Void
Int main()
^
./examplePrograms/problematic.spl:13:1: note: Type Int inferred here:
Int main()
^~~
./examplePrograms/problematic.spl:13:1: note: Type Void inferred here:
Int main()
^
./examplePrograms/problematic.spl:13:1: error: Type mismatch. Expected type Int declared here:
Int main()
^~~
./examplePrograms/problematic.spl:13:1: note: Actual type Void inferred here:
Int main()
^
*** Typing failed (but we try to continue!)
// deducting_a_minus_id :: Int()
Int deducting_a_minus_id/*6*/(){
        return 1;
}

// doing_just_a_funCall :: Void()
Void doing_just_a_funCall/*7*/(){
        print/*0*/(4 - (-deducting_a_minus_id/*6*/));
}

// main :: Int()
Int main/*8*/(){
        doing_just_a_funCall/*7*/();
}
*** Done


*** Compiling ./examplePrograms/problematic_programs.spl ***
/*
Problematic programs - Compiler Construction

Stein Keijzers (3004856)
Peter Maandag (3047121)
*/

Void NegativeAssociativity(Int n) {return n - 3 - 2 - 1;}
Void AndAssociativity(Bool b1, Bool b2, Bool b3) {return b1 && b2 && b3;}
Void PlusAssociativity(Int n) {return n + 3 + 2 + 1;}
Void DivAssociativity(Int n) {return n / 3 / 2 / 1;}
Void PlusMulPriority(Int n) {return n * n + n * n;}
Void NegativeNumber(Int n) {return n + -5;}
*** Done reading file
```

```
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
./examplePrograms/problematic_programs.spl:8:1: error: Cannot unify types Void and Int
Void NegativeAssociativity(Int n) {return n - 3 - 2 - 1;}
^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
./examplePrograms/problematic_programs.spl:8:1: note: Type Void inferred here:
Void NegativeAssociativity(Int n) {return n - 3 - 2 - 1;}
^~~~
./examplePrograms/problematic_programs.spl:8:51: note: Type Int inferred here:
Void NegativeAssociativity(Int n) {return n - 3 - 2 - 1;}
                                                  ^~
./examplePrograms/problematic_programs.spl:8:1: error: Type mismatch. Expected type Void declared here:
Void NegativeAssociativity(Int n) {return n - 3 - 2 - 1;}
^~~~
./examplePrograms/problematic_programs.spl:8:51: note: Actual type Int inferred here:
Void NegativeAssociativity(Int n) {return n - 3 - 2 - 1;}
                                                  ^~
./examplePrograms/problematic_programs.spl:9:1: error: Cannot unify types Void and Bool
Void AndAssociativity(Bool b1, Bool b2, Bool b3) {return b1 && b2 && b3;}
^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
./examplePrograms/problematic_programs.spl:9:1: note: Type Void inferred here:
Void AndAssociativity(Bool b1, Bool b2, Bool b3) {return b1 && b2 && b3;}
^~~~
./examplePrograms/problematic_programs.spl:9:58: note: Type Bool inferred here:
Void AndAssociativity(Bool b1, Bool b2, Bool b3) {return b1 && b2 && b3;}
                                                         ^~~~~~~~~~~~~~
./examplePrograms/problematic_programs.spl:9:1: error: Type mismatch. Expected type Void declared here:
Void AndAssociativity(Bool b1, Bool b2, Bool b3) {return b1 && b2 && b3;}
^~~~
./examplePrograms/problematic_programs.spl:9:58: note: Actual type Bool inferred here:
Void AndAssociativity(Bool b1, Bool b2, Bool b3) {return b1 && b2 && b3;}
                                                         ^~~~~~~~~~~~~~
./examplePrograms/problematic_programs.spl:10:1: error: Cannot unify types Void and Int
Void PlusAssociativity(Int n) {return n + 3 + 2 + 1;}
^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
./examplePrograms/problematic_programs.spl:10:1: note: Type Void inferred here:
Void PlusAssociativity(Int n) {return n + 3 + 2 + 1;}
^~~~
./examplePrograms/problematic_programs.spl:10:47: note: Type Int inferred here:
Void PlusAssociativity(Int n) {return n + 3 + 2 + 1;}
                                              ^~
./examplePrograms/problematic_programs.spl:10:1: error: Type mismatch. Expected type Void declared here:
Void PlusAssociativity(Int n) {return n + 3 + 2 + 1;}
^~~~
./examplePrograms/problematic_programs.spl:10:47: note: Actual type Int inferred here:
Void PlusAssociativity(Int n) {return n + 3 + 2 + 1;}
                                              ^~
./examplePrograms/problematic_programs.spl:11:1: error: Cannot unify types Void and Int
Void DivAssociativity(Int n) {return n / 3 / 2 / 1;}
^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
./examplePrograms/problematic_programs.spl:11:1: note: Type Void inferred here:
Void DivAssociativity(Int n) {return n / 3 / 2 / 1;}
^~~~
./examplePrograms/problematic_programs.spl:11:46: note: Type Int inferred here:
Void DivAssociativity(Int n) {return n / 3 / 2 / 1;}
                                              ^~
./examplePrograms/problematic_programs.spl:11:1: error: Type mismatch. Expected type Void declared here:
Void DivAssociativity(Int n) {return n / 3 / 2 / 1;}
^~~~
./examplePrograms/problematic_programs.spl:11:46: note: Actual type Int inferred here:
Void DivAssociativity(Int n) {return n / 3 / 2 / 1;}
                                              ^~
./examplePrograms/problematic_programs.spl:12:1: error: Cannot unify types Void and Int
Void PlusMulPriority(Int n) {return n * n + n * n;}
^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
./examplePrograms/problematic_programs.spl:12:1: note: Type Void inferred here:
Void PlusMulPriority(Int n) {return n * n + n * n;}
^~~~
./examplePrograms/problematic_programs.spl:12:37: note: Type Int inferred here:
Void PlusMulPriority(Int n) {return n * n + n * n;}
                                    ^~~~~~~
./examplePrograms/problematic_programs.spl:12:1: error: Type mismatch. Expected type Void declared here:
Void PlusMulPriority(Int n) {return n * n + n * n;}
^~~~
./examplePrograms/problematic_programs.spl:12:37: note: Actual type Int inferred here:
Void PlusMulPriority(Int n) {return n * n + n * n;}
                                    ^~~~~~~
./examplePrograms/problematic_programs.spl:13:1: error: Cannot unify types Void and Int
Void NegativeNumber(Int n) {return n + -5;}
^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
./examplePrograms/problematic_programs.spl:13:1: note: Type Void inferred here:
Void NegativeNumber(Int n) {return n + -5;}
^~~~
./examplePrograms/problematic_programs.spl:13:36: note: Type Int inferred here:
Void NegativeNumber(Int n) {return n + -5;}
                                    ^~
./examplePrograms/problematic_programs.spl:13:1: error: Type mismatch. Expected type Void declared here:
Void NegativeNumber(Int n) {return n + -5;}
^~~~
./examplePrograms/problematic_programs.spl:13:36: note: Actual type Int inferred here:
Void NegativeNumber(Int n) {return n + -5;}
                                    ^~
*** Typing failed (but we try to continue!)
// NegativeAssociativity :: Void(Int)
Void NegativeAssociativity/*6*/(Int n/*12*/){
        return ((n/*12*/ - 3) - 2) - 1;
}

// AndAssociativity :: Void(Bool, Bool, Bool)
Void AndAssociativity/*7*/(Bool b1/*13*/, Bool b2/*14*/, Bool b3/*15*/){
        return b1/*13*/ && (b2/*14*/ && b3/*15*/);
}

// PlusAssociativity :: Void(Int)
Void PlusAssociativity/*8*/(Int n/*16*/){
        return ((n/*16*/ + 3) + 2) + 1;
}
```

```
// DivAssociativity :: Void(Int)
Void DivAssociativity/*9*/(Int n/*17*/){
        return ((n/*17*/ / 3) / 2) / 1;
}

// PlusMulPriority :: Void(Int)
Void PlusMulPriority/*10*/(Int n/*18*/){
        return (n/*18*/ * n/*18*/) + (n/*18*/ * n/*18*/);
}

// NegativeNumber :: Void(Int)
Void NegativeNumber/*11*/(Int n/*19*/){
        return n/*19*/ + (-5);
}
*** Done


*** Compiling ./examplePrograms/stress.spl ***
//Marc  Schoolderman (s0620866)
// our pretty printer should not only print code correctly, but not
// introduce parentheses or braces where they are not needed

fnord/**/main()  // should not paste tokens together!
{{
    if([])
      if (([] : -(1*[])-(2*[]-3*[])-4/[]/-5): (1*[])-(2+[])--3*(4+[]))
      {
        return !1:2; /* should parse as (!1):2, not !(1:2) */
        [([foo],bar)] y = this_is_fine((1,(2,3)));
      }
    else Void//
x = not_allowed((1),2);
/* this declarion will be made illegal in the next compilation phase */
}}

/* output:

fnord main()
{
    {
        if([])
            if(([] : -(1*[])-(2*[]-3*[])-4/[]/-5) : 1*[]-(2+[])--3*(4+[])) {
                return !1 : 2;
                [([foo],bar)] y = this_is_fine((1,(2,3)));
            } else
                [Void] x = not_allowed(1, 2);
    }
}


*/

*** Done reading file
*** Lexing is done
./examplePrograms/stress.spl:11:3: error: Unexpected token ParenthesesOpen
        [([foo],bar)] y = this_is_fine((1,(2,3)));
         ^
*** Stopping due to fatal error


*** Compiling ./examplePrograms/sum.spl ***
//Jan  Groothuijse (e727426)
Int sum([Int] l) {
        if (!isEmpty(tl(l))) {
                return hd(l) + sum(tl(l));
        }
        return hd(l);
}

Int sum1([Int] l) {
        if (isEmpty(tl(l))) {
                return hd(l);
        }
        return hd(l) + sum(tl(l));
}

Int sum2([Int] l) {
        if (isEmpty(tl(l))) {
                return hd(l);
        } else {
                return hd(l) + sum(tl(l));
        }
}

Void main()
{
        print (sum([1:2:3:[]]));
        print (sum1([1:2:3:[]]));
        print (sum2([1:2:3:[]]));
        return;
}
*** Done reading file
*** Lexing is done
./examplePrograms/sum.spl:26:14: error: Unexpected token Integer 1
        print (sum([1:2:3:[]]));
                   ^
*** Stopping due to fatal error


*** Compiling ./examplePrograms/while.spl ***
//Jan  Groothuijse (e727426)
Void keepGoingI (Int n) {
        while (true) {
                n = n + 1;
        }
}
```

```
Void keepGoingR (Int n) {
        return keepGoingR (n + 1);
}

*** Done reading file
*** Lexing is done
*** Parsing succeeded
./examplePrograms/while.spl:3:9: error: Undeclared identifier "true"
        while (true) {
               ^~~~
./examplePrograms/while.spl:3:9: note: Did you mean "tail"?
*** Scoping failed (but we try to continue!)
./examplePrograms/while.spl:3:9: error: Identifier is unknown: true
        while (true) {
               ^~~~
./examplePrograms/while.spl:3:9: note: This is probably caused by earlier errors
*** Stopping due to fatal error


*** Compiling ./examplePrograms/whitespaces.spl ***
//Tim  Steenvoorden (s0712663)

/* Some code in a comment

   Void int ()
   {id = null;}

   if(x == < 6) {}
*/

// A strange name, but parsable
Void CAPPS (Int X , Bool S)
{
        // Everthing on one line
        X = 0; S = True; return;
}

Void tabbed ()
{
        // Indented with tabs
        return;
}

*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
*** Typing succeeded
// CAPPS :: Void(Int, Bool)
Void CAPPS/*6*/(Int X/*8*/, Bool S/*9*/){
        X/*8*/ = 0;
        S/*9*/ = True;
        return;
}

// tabbed :: Void()
Void tabbed/*7*/(){
        return;
}
*** Done


*** Compiling ./examplePrograms/x.spl ***
//Jan  Groothuijse (e727426)
a xMarksTheSpot (a x) {
        return (((((((((((((((((((((((((((((((((((((((((((((((((((((((((x)))))))))))))))))))))))))))))))))))))))))))))))))))))))));
}




*** Done reading file
*** Lexing is done
*** Parsing succeeded
*** Scoping succeeded
*** Typing succeeded
// xMarksTheSpot :: forall a0 . a0(a0)
a xMarksTheSpot/*6*/(a x/*7*/){
        return x/*7*/;
}
*** Done

joshua@192:~/Documents/Code/splang/midreport/examplePrograms (master)$
```