

# **Programming Paradigms Final Project: Test Report**

GROUP 33  
MARTIJN VERKLEIJ & WOUTER BOS

University of Twente  
m.f.verkleij@student.utwente.nl, w.f.a.bos@student.utwente.nl  
s1466895 s1606824

July 7, 2017

# Contents

<b>1</b>	<b>Syntax Tests</b>	<b>4</b>
1.1	Syntax 1 . . . . .	4
1.1.1	Source . . . . .	4
1.1.2	Output . . . . .	4
1.2	Syntax 2 . . . . .	4
1.2.1	Source . . . . .	4
1.2.2	Output . . . . .	4
1.3	Syntax 3 . . . . .	4
1.3.1	Source . . . . .	4
1.3.2	Output . . . . .	4
1.4	Syntax 4 . . . . .	5
1.4.1	Source . . . . .	5
1.4.2	Output . . . . .	5
1.5	Syntax 5 . . . . .	5
1.5.1	Source . . . . .	5
1.5.2	Output . . . . .	5
<b>2</b>	<b>Contextual Tests</b>	<b>6</b>
2.1	Wrong Type . . . . .	6
2.1.1	Source . . . . .	6
2.1.2	Output . . . . .	6
2.2	Not Declared . . . . .	6
2.2.1	Source . . . . .	6
2.2.2	Output . . . . .	6
<b>3</b>	<b>Semantic Tests</b>	<b>7</b>
3.1	Banking . . . . .	7
3.1.1	Source . . . . .	7
3.1.2	Results . . . . .	8
3.2	Blocks . . . . .	9
3.2.1	Source . . . . .	9
3.2.2	Results . . . . .	10
3.3	Call-by-reference . . . . .	11
3.3.1	Source . . . . .	11
3.3.2	Results . . . . .	12
3.4	Cyclic Recursion . . . . .	12
3.4.1	Source . . . . .	12

3.4.2	Results	12
3.5	Deep Expression	13
3.5.1	Source	13
3.5.2	Results	13
3.6	Fib	13
3.6.1	Source	13
3.6.2	Results	14
3.7	If	14
3.7.1	Source	14
3.7.2	Results	14
3.8	If Else	14
3.8.1	Source	14
3.8.2	Results	14
3.9	Infinite Busy Loop	15
3.9.1	Source	15
3.9.2	Results	15
3.10	Infinite Empty Loop	17
3.10.1	Source	17
3.10.2	Results	17
3.11	Join Test	17
3.11.1	Source	17
3.11.2	Results	17
3.12	Multiple Globals	17
3.12.1	Source	17
3.12.2	Results	18
3.13	Nested Procedures	18
3.13.1	Source	18
3.13.2	Results	19
3.14	Peterson	20
3.14.1	Source	20
3.14.2	Results	21
3.15	Recursion	22
3.15.1	Source	22
3.15.2	Results	22
3.16	Simple Concurrency	22
3.16.1	Source	22
3.16.2	Results	23
3.17	Simple Procedures	23
3.17.1	Source	23
3.17.2	Results	23
3.18	While	23
3.18.1	Source	23
3.18.2	Results	24
3.19	Enumerations	24
3.19.1	Source	24
3.19.2	Results	25

# Chapter 1

## Syntax Tests

### 1.1 Syntax 1

#### 1.1.1 Source

```
1 int number;  
2  
3 procedure p() {}
```

#### 1.1.2 Output

```
1 Main: tokenList not fully parsed
```

### 1.2 Syntax 2

#### 1.2.1 Source

```
1 procedure(j, int i) {};
```

#### 1.2.2 Output

```
1 Main: tokenList not fully parsed
```

### 1.3 Syntax 3

#### 1.3.1 Source

```
1 int 5num = 5;
```

#### 1.3.2 Output

```
1 Main: tokenList not fully parsed
```

## 1.4 Syntax 4

### 1.4.1 Source

```
1 if (true) else {// do nothing}
```

### 1.4.2 Output

```
1 Main: tokenList not fully parsed
```

## 1.5 Syntax 5

### 1.5.1 Source

```
1 print(5*(3&&||/1));
```

### 1.5.2 Output

```
1 Main: tokenList not fully parsed
```

## Chapter 2

# Contextual Tests

### 2.1 Wrong Type

#### 2.1.1 Source

```
1 int i = 1;
2 if (i) {
3     print(i);
4 }
```

#### 2.1.2 Output

```
1 Main: Condition in if statement should be of type: bool, but isnt, in: ASTVar "i" ([],[],[],[])
```

### 2.2 Not Declared

#### 2.2.1 Source

```
1 if (i) {
2     print(1);
3 }
```

#### 2.2.2 Output

```
1 Main: Variable: i not declared in Checker.getExprType.iterVar
```

# Chapter 3

## Semantic Tests

### 3.1 Banking

This test has been run on 3 Sprockells.

#### 3.1.1 Source

```
1  global int john = 10000;
2  global int jane = 2000;
3  global int martijn = 99999;
4
5  procedure deposit(int account, int amount) {
6      account = (account + amount);
7  }
8
9  procedure withdraw(int account, int amount) {
10     if ((account >= amount)) {
11         account = (account - amount);
12     }
13 }
14
15 procedure transfer(int sender, int target, int amount) {
16     if ((sender >= amount)) {
17         sender = (sender - amount);
18         target = (target + amount);
19     }
20 }
21
22 procedure test1() {
23     print(john, jane, martijn);
24     fork deposit(jane, 100);
25     fork deposit(john, 100);
26     fork deposit(martijn, 1);
27     join;
28     print(john, jane, martijn);
```

```
29
30     fork deposit(jane, 200);
31     fork withdraw(john, 200);
32     fork deposit(martijn, 2000);
33     join;
34     print(john, jane, martijn);
35     fork withdraw(jane, 10);
36     fork withdraw(john, 20);
37     fork deposit(martijn, 100);
38     join;
39     print(john, jane, martijn);
40     fork withdraw(jane, 300);
41     fork withdraw(john, 30000);
42     fork withdraw(martijn, 50);
43     join;
44     print(john, jane, martijn);
45     fork withdraw(jane, 35);
46     fork transfer(martijn, john, 1000);
47     join;
48     print(john, jane, martijn);
49     fork transfer(martijn, jane, 100);
50     join;
51     print(john, jane, martijn);
52
53 }
54
55 test1();
```

### 3.1.2 Results

```
1  Sprockell 0 says 10000
2  Sprockell 0 says 2000
3  Sprockell 0 says 99999
4  Sprockell 0 says 10100
5  Sprockell 0 says 2100
6  Sprockell 0 says 100000
7  Sprockell 0 says 9900
8  Sprockell 0 says 2300
9  Sprockell 0 says 102000
10 Sprockell 0 says 9880
11 Sprockell 0 says 2290
12 Sprockell 0 says 102100
13 Sprockell 0 says 9880
14 Sprockell 0 says 1990
15 Sprockell 0 says 102050
16 Sprockell 0 says 10880
17 Sprockell 0 says 1955
18 Sprockell 0 says 101050
```



```
19 Sprockell 0 says 10880
20 Sprockell 0 says 2055
21 Sprockell 0 says 100950
```

## 3.2 Blocks

### 3.2.1 Source

```
1  int x = 1;
2  int y = 100;
3  bool a = false;
4  bool b = false;
5  print(x,y,a,b);
6  {
7      int x = 2;
8      int y = 120;
9      bool a = true;
10     bool b = false;
11     print(x,y,a,b);
12     { }
13     int x = 3;
14     int y = 123;
15     bool a = false;
16     bool b = true;
17     print(x,y,a,b);
18     {
19         int x = 4;
20         int y = 423;
21         bool a = true;
22         bool b = true;
23         print(x,y,a,b);
24     }
25     print(x,y,a,b);
26     {
27         int x = 5;
28         int y = 453;
29         bool a = true;
30         bool b = false;
31         print(x,y,a,b);
32         {
33             int x = 5;
34             int y = 453;
35             bool a = false;
36             bool b = true;
37             print(x,y,a,b);
38         }
39         print(x,y,a,b);
40         {
```

```
41         int x = 6;
42         int y = 456;
43         bool a = false;
44         bool b = false;
45         print(x,y,a,b);
46     }
47     print(x,y,a,b);
48 }
49 print(x,y,a,b);
50 }
51 print(x,y,a,b);
52 {
53     print(x,y,a,b);
54     {
55         print(x,y,a,b);
56     }
57 }
58 print(x,y,a,b);
59 }
60 print(x,y,a,b);
```

### 3.2.2 Results

```
1  Sprockell 0 says 1
2  Sprockell 0 says 100
3  Sprockell 0 says 0
4  Sprockell 0 says 0
5  Sprockell 0 says 2
6  Sprockell 0 says 120
7  Sprockell 0 says 1
8  Sprockell 0 says 0
9  Sprockell 0 says 3
10 Sprockell 0 says 123
11 Sprockell 0 says 0
12 Sprockell 0 says 1
13 Sprockell 0 says 4
14 Sprockell 0 says 423
15 Sprockell 0 says 1
16 Sprockell 0 says 1
17 Sprockell 0 says 3
18 Sprockell 0 says 123
19 Sprockell 0 says 0
20 Sprockell 0 says 1
21 Sprockell 0 says 5
22 Sprockell 0 says 453
23 Sprockell 0 says 1
24 Sprockell 0 says 0
25 Sprockell 0 says 5
```

```
26 Sprockell 0 says 453
27 Sprockell 0 says 0
28 Sprockell 0 says 1
29 Sprockell 0 says 5
30 Sprockell 0 says 453
31 Sprockell 0 says 1
32 Sprockell 0 says 0
33 Sprockell 0 says 6
34 Sprockell 0 says 456
35 Sprockell 0 says 0
36 Sprockell 0 says 0
37 Sprockell 0 says 5
38 Sprockell 0 says 453
39 Sprockell 0 says 1
40 Sprockell 0 says 0
41 Sprockell 0 says 3
42 Sprockell 0 says 123
43 Sprockell 0 says 0
44 Sprockell 0 says 1
45 Sprockell 0 says 2
46 Sprockell 0 says 120
47 Sprockell 0 says 1
48 Sprockell 0 says 0
49 Sprockell 0 says 2
50 Sprockell 0 says 120
51 Sprockell 0 says 1
52 Sprockell 0 says 0
53 Sprockell 0 says 2
54 Sprockell 0 says 120
55 Sprockell 0 says 1
56 Sprockell 0 says 0
57 Sprockell 0 says 2
58 Sprockell 0 says 120
59 Sprockell 0 says 1
60 Sprockell 0 says 0
61 Sprockell 0 says 1
62 Sprockell 0 says 100
63 Sprockell 0 says 0
64 Sprockell 0 says 0
```

### 3.3 Call-by-reference

This test has been run on 3 Sprockells.

#### 3.3.1 Source

```
1 global int var = 1337;
2 global int y = 42;
```

```
3
4 procedure write(int input, int output) {
5     output = input;
6 }
7
8 print (y);
9 fork write(var, y);
10 join;
11 print(y);
```

### 3.3.2 Results

```
1 Sprockell 0 says 42
2 Sprockell 0 says 1337
```

## 3.4 Cyclic Recursion

### 3.4.1 Source

```
1 procedure prod(int i) {
2     i = (i + 1);
3     //print (i);
4     cons(i);
5 }
6
7 procedure cons(int i) {
8     if ((i > 1)) {
9         i = (i - 2);
10        print(i);
11        prod(i);
12    } else if ((i > 0)) {
13        i = (i - 1);
14        print(i);
15    }
16 }
17
18 prod(18);
```

### 3.4.2 Results

```
1 Sprockell 0 says 17
2 Sprockell 0 says 16
3 Sprockell 0 says 15
4 Sprockell 0 says 14
5 Sprockell 0 says 13
6 Sprockell 0 says 12
7 Sprockell 0 says 11
8 Sprockell 0 says 10
```

```
9  Sprockell 0 says 9
10 Sprockell 0 says 8
11 Sprockell 0 says 7
12 Sprockell 0 says 6
13 Sprockell 0 says 5
14 Sprockell 0 says 4
15 Sprockell 0 says 3
16 Sprockell 0 says 2
17 Sprockell 0 says 1
18 Sprockell 0 says 0
19 Sprockell 0 says 0
```

## 3.5 Deep Expression

### 3.5.1 Source

```
1  int a = 100;
2  //100000;
3  //200000;
4  //300000;
5  //400000;
6  a = ((a + (((10 * (-15)) * 42) * (3 + 2))) * (2 * (7 + 11 - 98))); // comment test
7  a = 3 * 5 + 100 - 1 - 1 + a;
8  print(a);
9  print(--a);
10 print(a);
```

### 3.5.2 Results

```
1  Sprockell 0 says 5024113
2  Sprockell 0 says 5024112
3  Sprockell 0 says 5024112
```

## 3.6 Fib

### 3.6.1 Source

```
1  procedure fib(int i, int res) {
2      if ((i < (3))) {
3          res = 1;
4      } else {
5          int a;
6          int b;
7          fib((i-1), a);
8          fib((i-2), b);
9          res = (a + b);
10
11      }
```

```
12 }
13
14 int a = 0;
15 fib(8, a);
16 print(a);
```

### 3.6.2 Results

```
1 Sprockell 0 says 5024113 21
```

## 3.7 If

### 3.7.1 Source

```
1 bool condition = true;
2 if (condition) print(1); else print(0);
```

### 3.7.2 Results

```
1 Sprockell 0 says 1
```

## 3.8 If Else

### 3.8.1 Source

```
1 int i = 4;
2 print (i);
3 if ((i == 2)) {
4     print(3,i);
5 } else if ((i == 1)) {{{
6     print(4,i);
7 }}} else {{{
8     print(5,i);
9 }}}
10
11
12 if ((i == 4)) {
13     print(3,i);
14 } else if ((i == 1)) {{{
15     print(4,i);
16 }}}}
```

### 3.8.2 Results

```
1 Sprockell 0 says 4
2 Sprockell 0 says 5
3 Sprockell 0 says 4
```

```
4 Sprockell 0 says 3
5 Sprockell 0 says 4
```

## 3.9 Infinite Busy Loop

### 3.9.1 Source

```
1 int i = 0;
2 int j = 1;
3 while (true) {
4     i = (i + j);
5     j = (j * i);
6     print(i,j);
7 }
```

### 3.9.2 Results

Gets stuck in an infinite loop, repeating the same output.

```
1 Sprockell 0 says 1
2 Sprockell 0 says 1
3 Sprockell 0 says 2
4 Sprockell 0 says 2
5 Sprockell 0 says 4
6 Sprockell 0 says 8
7 Sprockell 0 says 12
8 Sprockell 0 says 96
9 Sprockell 0 says 108
10 Sprockell 0 says 10368
11 Sprockell 0 says 10476
12 Sprockell 0 says 108615168
13 Sprockell 0 says 108625644
14 Sprockell 0 says 11798392572168192
15 Sprockell 0 says 11798392680793836
16 Sprockell 0 says -5570361874949185536
17 Sprockell 0 says -5558563482268391700
18 Sprockell 0 says 3671369242980155392
19 Sprockell 0 says -1887194239288236308
20 Sprockell 0 says -4483044364780175360
21 Sprockell 0 says -6370238604068411668
22 Sprockell 0 says -8730959061097906176
23 Sprockell 0 says 3345546408543233772
24 Sprockell 0 says -6745737849034768384
25 Sprockell 0 says -3400191440491534612
26 Sprockell 0 says -6096120617457680384
27 Sprockell 0 says 8950432015760336620
28 Sprockell 0 says -1019520187243692032
29 Sprockell 0 says 7930911828516644588
```

30 Sprockell 0 says -4809903748681826304  
31 Sprockell 0 says 3121008079834818284  
32 Sprockell 0 says 5865085819223539712  
33 Sprockell 0 says 8986093899058357996  
34 Sprockell 0 says 2740241432517279744  
35 Sprockell 0 says -6720408742133913876  
36 Sprockell 0 says 3246081813541552128  
37 Sprockell 0 says -3474326928592361748  
38 Sprockell 0 says -1859074291971129344  
39 Sprockell 0 says -5333401220563491092  
40 Sprockell 0 says 681350175863603200  
41 Sprockell 0 says -4652051044699887892  
42 Sprockell 0 says -8143132099134619648  
43 Sprockell 0 says 5651560929875044076  
44 Sprockell 0 says 6951259845357993984  
45 Sprockell 0 says -5843923298476513556  
46 Sprockell 0 says 4700992750881865728  
47 Sprockell 0 says -1142930547594647828  
48 Sprockell 0 says -8561800288468467712  
49 Sprockell 0 says 8742013237646436076  
50 Sprockell 0 says 7566188111470788608  
51 Sprockell 0 says -2138542724592326932  
52 Sprockell 0 says -6956372574427152384  
53 Sprockell 0 says -9094915299019479316  
54 Sprockell 0 says -8878846665360932864  
55 Sprockell 0 says 472982109329139436  
56 Sprockell 0 says 1756403854674493440  
57 Sprockell 0 says 2229385964003632876  
58 Sprockell 0 says -5152117973711847424  
59 Sprockell 0 says -2922732009708214548  
60 Sprockell 0 says 1585267068834414592  
61 Sprockell 0 says -1337464940873799956  
62 Sprockell 0 says 5188146770730811392  
63 Sprockell 0 says 3850681829857011436  
64 Sprockell 0 says 6917529027641081856  
65 Sprockell 0 says -7678533216211458324  
66 Sprockell 0 says -9223372036854775808  
67 Sprockell 0 says 1544838820643317484  
68 Sprockell 0 says 0  
69 Sprockell 0 says 1544838820643317484  
70 Sprockell 0 says 0  
71 Sprockell 0 says 1544838820643317484  
72 Sprockell 0 says 0  
73 ...



## 3.10 Infinite Empty Loop

### 3.10.1 Source

```
1 while (true) {  
2     // do nothing  
3 }
```

### 3.10.2 Results

No output, gets stuck in an infinite loop.

## 3.11 Join Test

This test has been run on 2 Sprockells.

### 3.11.1 Source

```
1 global int after_a_very_long_time = 10000;  
2  
3 procedure ending(int j) {  
4     while( (j > 0) ){  
5         j = (j-1);  
6  
7         if ( (j == 5000) ) {  
8             print(j);  
9         }  
10    }  
11  
12 }  
13  
14 fork ending(after_a_very_long_time);  
15 join;  
16 print(10000);
```

### 3.11.2 Results

Sprockell 1 says 5000 Sprockell 0 says 10000

## 3.12 Multiple Globals

This test has been run on 3 Sprockells.

### 3.12.1 Source

```
1 global int a = 8;  
2 global int b = 9;  
3 global int c = 10;
```

```
4  global int d = 11;
5  global int e = 12;
6  global int f = 13;
7
8  procedure printAll() {
9      print(a,b,c,d,e,f);
10 }
11
12 procedure printAllBW() {
13     print(f,e,d,c,b,a);
14 }
15
16
17 fork printAll();
18 fork printAllBW();
19 join;
```

### 3.12.2 Results

```
1  Sprockell 1 says 8
2  Sprockell 1 says 9
3  Sprockell 1 says 10
4  Sprockell 1 says 11
5  Sprockell 1 says 12
6  Sprockell 1 says 13
7  Sprockell 2 says 13
8  Sprockell 2 says 12
9  Sprockell 2 says 11
10 Sprockell 2 says 10
11 Sprockell 2 says 9
12 Sprockell 2 says 8
```

## 3.13 Nested Procedures

### 3.13.1 Source

```
1  procedure p0() {
2      print(90);
3      p1();
4      print(91);
5      p2();
6      print(92);
7      p3();
8      print(93);
9      p4();
10     print(94);
11 }
12
```

```
13 procedure p1() {
14     print(10);
15     p2();
16     print(12);
17     p3();
18     print(13);
19     p4();
20     print(14);
21 }
22
23 procedure p2() {
24     print(20);
25     p3();
26     print(23);
27     p4();
28     print(24);
29 }
30
31 procedure p3() {
32     print(30);
33     p4();
34     print(34);
35 }
36
37 procedure p4() {
38     print(40);
39 }
40
41 p0();
```

### 3.13.2 Results

```
1 Sprockell 0 says 90
2 Sprockell 0 says 10
3 Sprockell 0 says 20
4 Sprockell 0 says 30
5 Sprockell 0 says 40
6 Sprockell 0 says 34
7 Sprockell 0 says 23
8 Sprockell 0 says 40
9 Sprockell 0 says 24
10 Sprockell 0 says 12
11 Sprockell 0 says 30
12 Sprockell 0 says 40
13 Sprockell 0 says 34
14 Sprockell 0 says 13
15 Sprockell 0 says 40
16 Sprockell 0 says 14
```

```
17 Sprockell 0 says 91
18 Sprockell 0 says 20
19 Sprockell 0 says 30
20 Sprockell 0 says 40
21 Sprockell 0 says 34
22 Sprockell 0 says 23
23 Sprockell 0 says 40
24 Sprockell 0 says 24
25 Sprockell 0 says 92
26 Sprockell 0 says 30
27 Sprockell 0 says 40
28 Sprockell 0 says 34
29 Sprockell 0 says 93
30 Sprockell 0 says 40
31 Sprockell 0 says 94
```

### 3.14 Peterson

This test has been run on 3 Sprockells.

#### 3.14.1 Source

```
1  global bool flag_0 = false;
2  global bool flag_1 = false;
3  global int turn = 0;
4  global int i = 0;
5
6  enum test = {die};
7
8  procedure p_0() {
9      flag_0 = true;
10     turn = 1;
11     while ((flag_1 && (turn == 1))) {
12         // wait
13     }
14     // begin critical section
15     int j = 5;
16     while ((j > 0)) {
17         i = ++i;
18         j = --j;
19     }
20     // end critical section
21     flag_0 = false;
22 }
23
24 procedure p_1() {
25     flag_1 = true;
26     turn = 0;
```

```
27     while ((flag_0 && (turn == 0))) {
28         // wait
29     }
30     // begin critical section
31     int j = 5;
32     while ((j > 0)) {
33         i = --i;
34         j = --j;
35     }
36     // end critical section
37     flag_1 = false;
38 }
39
40 procedure test1(int j) {
41     while ((j > 0)) {
42         fork p_0();
43         fork p_1();
44         join;
45         print(i);
46
47         fork p_1();
48         fork p_0();
49         join;
50         print(i);
51
52         j = --j;
53     }
54 }
55
56 test1(10);
```

### 3.14.2 Results

```
1  Sprockell 0 says 0
2  Sprockell 0 says 0
3  Sprockell 0 says 0
4  Sprockell 0 says 0
5  Sprockell 0 says 0
6  Sprockell 0 says 0
7  Sprockell 0 says 0
8  Sprockell 0 says 0
9  Sprockell 0 says 0
10 Sprockell 0 says 0
11 Sprockell 0 says 0
12 Sprockell 0 says 0
13 Sprockell 0 says 0
14 Sprockell 0 says 0
15 Sprockell 0 says 0
```

```
16 Sprockell 0 says 0
17 Sprockell 0 says 0
18 Sprockell 0 says 0
19 Sprockell 0 says 0
20 Sprockell 0 says 0
```

## 3.15 Recursion

### 3.15.1 Source

```
1 procedure rec(int i) {
2     if ((i < 3)) {
3         print(i);
4         i = (i + 1);
5         rec(i);
6     } else {
7         print(i);
8     }
9 }
10
11 int i = 0;
12 rec(i);
```

### 3.15.2 Results

```
1 Sprockell 0 says 0
2 Sprockell 0 says 1
3 Sprockell 0 says 2
4 Sprockell 0 says 3
```

## 3.16 Simple Concurrency

This test has been run on 3 Sprockells.

### 3.16.1 Source

```
1 global int num = 5;
2
3 procedure set_four() {
4     (3+(2*(2*(2*332)))));
5     num = 4;
6     print(num);
7 }
8
9 procedure set_six() {
10    num = 6;
11    print(num);
12 }
```

```
13
14 fork set_four();
15 fork set_six();
16 join;
17 print(num);
```

### 3.16.2 Results

```
1 Sprockell 1 says 4
2 Sprockell 2 says 6
3 Sprockell 0 says 6
```

## 3.17 Simple Procedures

### 3.17.1 Source

```
1 global int a = 3;
2
3 procedure p0(int c) {
4     a = c;
5     c = (c + 2);
6 }
7
8 int b = 2;
9 print(b); // should print 2
10
11 a = 1;
12
13 p0(b);
14 print(b); // should print 4
15 print(0);
16 print(a); // should print 2
```

### 3.17.2 Results

```
1 Sprockell 0 says 2
2 Sprockell 0 says 4
3 Sprockell 0 says 0
4 Sprockell 0 says 2
```

## 3.18 While

### 3.18.1 Source

```
1 int i = 100;
2 while ((i >= 0)) {
3     1;
4     print(i);
```

```
5     i = (i - 1);
6 }
```

### 3.18.2 Results

```
1 Sprockell 0 says 100
2 Sprockell 0 says 99
3 Sprockell 0 says 98
4 Sprockell 0 says 97
5 Sprockell 0 says 96
6 Sprockell 0 says 95
7 Sprockell 0 says 94
8 Sprockell 0 says 93
9 Sprockell 0 says 92
10 Sprockell 0 says 91
11 Sprockell 0 says 90
12 ...
13 Sprockell 0 says 10
14 Sprockell 0 says 9
15 Sprockell 0 says 8
16 Sprockell 0 says 7
17 Sprockell 0 says 6
18 Sprockell 0 says 5
19 Sprockell 0 says 4
20 Sprockell 0 says 3
21 Sprockell 0 says 2
22 Sprockell 0 says 1
23 Sprockell 0 says 0
```

## 3.19 Enumerations

### 3.19.1 Source

```
1 global enum boe = test;
2 global enum boe2 = test;
3 global enum boe3 = bar;
4
5 enum bla = {test, test1, test2};
6 enum foo = {bar,baz};
7
8 procedure prok() {
9     boe = test2;
10 }
11
12 bool b = false;
13
14 bool a = (true);
15 print(a);
```



```
16
17  a = (test == bar);
18
19  print(a);
20  a = (test == test);
21  print(a);
22
23  //test + test;
24  //test - test;
25
26  //test * test;
27  //test < test;
28  //test > test;
29
30  print(test != test);
31  //test && test;
32  //test || test;
33  //test <> test;
34  //test <= test;
35  //test >= test;
```

### 3.19.2 Results

```
1  Sprockell 0 says 1
2  Sprockell 0 says 0
3  Sprockell 0 says 1
4  Sprockell 0 says 0
```