

Note: this code is very very bad, but it illustrates the mechanism very well.

CODE FRAGMENT

```

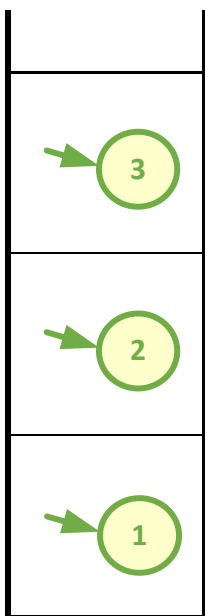
1  int start;
2  int f(int n){
3      int i, result;
4      void g(int n, int start) {
5          if n=start
6              return result+n;
7          else
8              return 1+g(n-1, start);
9      }
10     result = start;
11     for i=1 to n    g(i,0);
12     return result;
13 }
14 start = 10;
15 f(3);
    
```

IDENTIFIER TABLE

	lexeme	Scanned on line
0	"start"	1
1	"f"	2
2	"n"	2
3	"i"	3
4	"result"	3
5	"g"	4

PARSER CONSTRUCTION OF SYMBOL TABLES WITH SCOPE STACK

SCOPE STACK

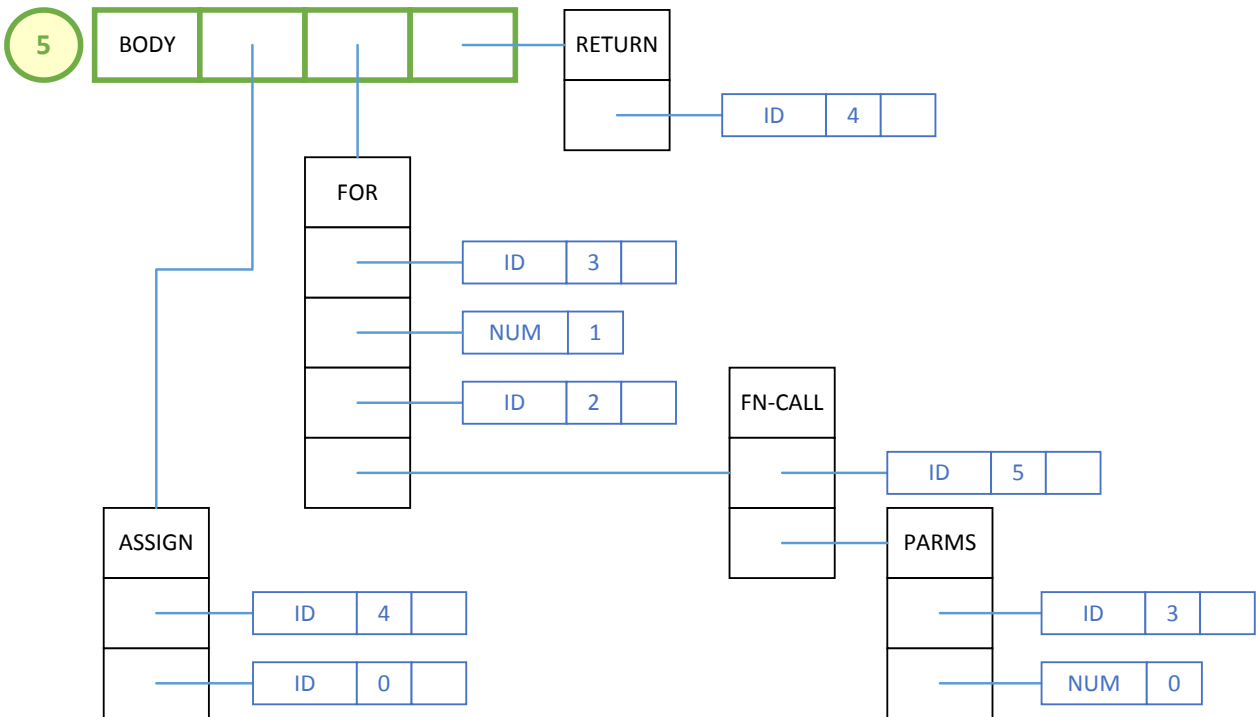
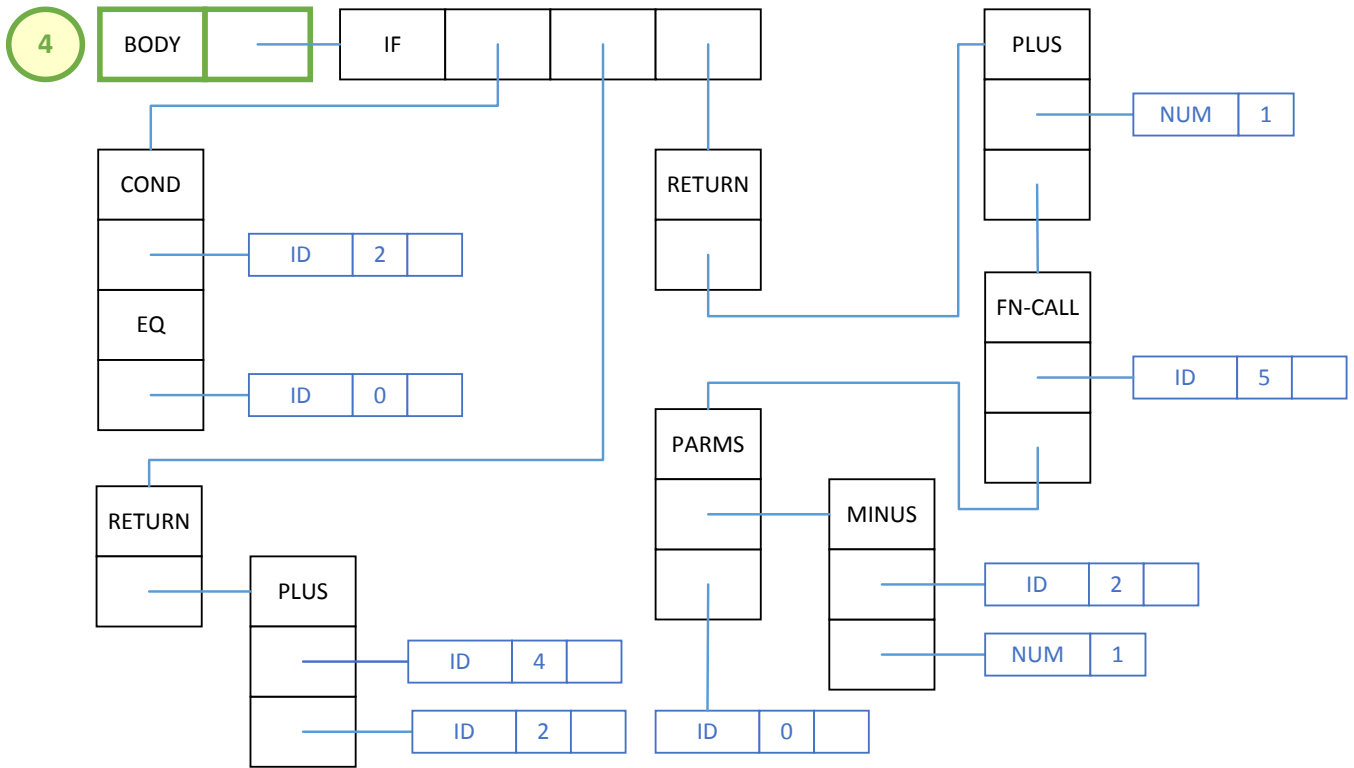


SYMBOL TABLES

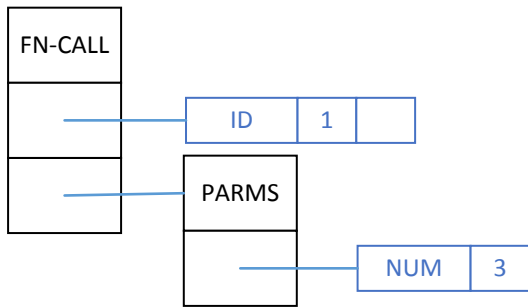
parsed ID Complexity Type Params Scope Body

4	0	2	Primitive	Int				3
	1	0	Primitive	Int				
2	0	2	Primitive	Int				2
	1	3	Primitive	Int				
	2	4	Primitive	Int				
	3	5	Function	void				
1	0	0	Primitive	Int				1
	1	1	Function	Int				

ASTS CONSTRUCTED: Note: body ASTs with only one statement have been simplified.

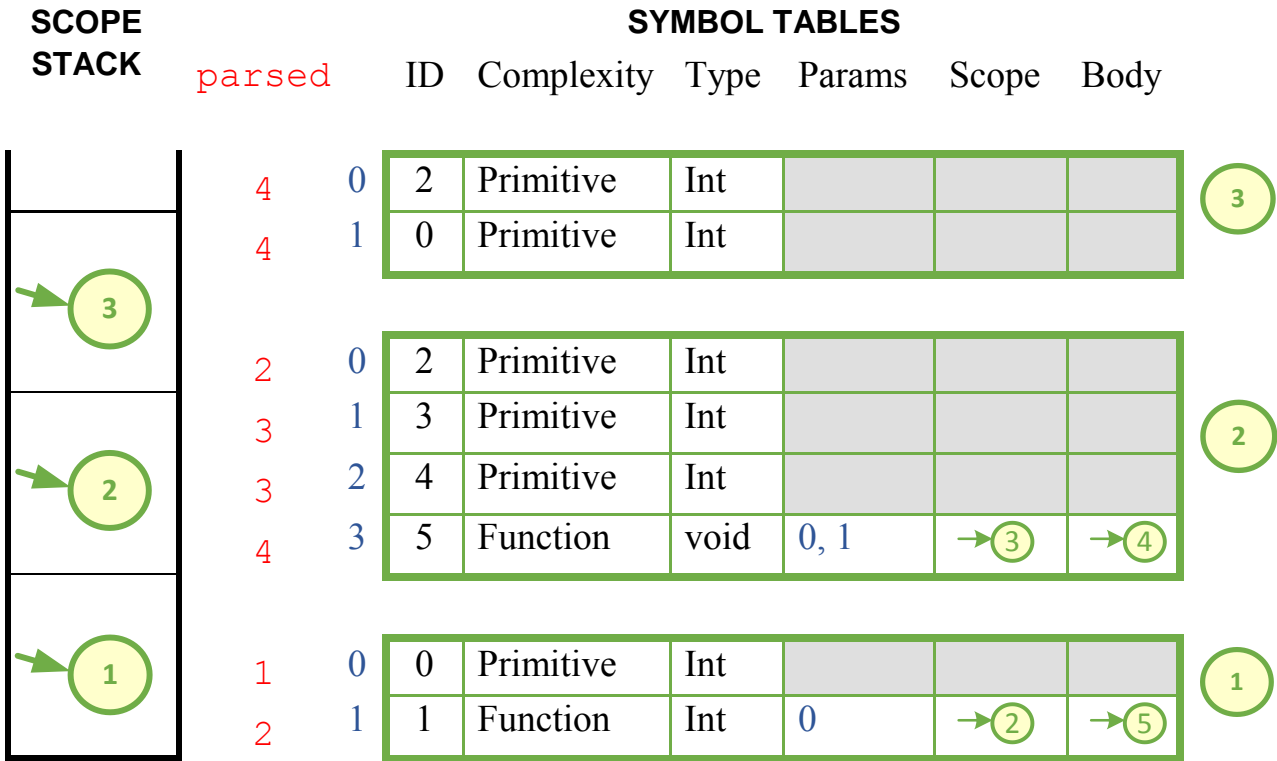


FUNCTION CALL EVALUATION (15)

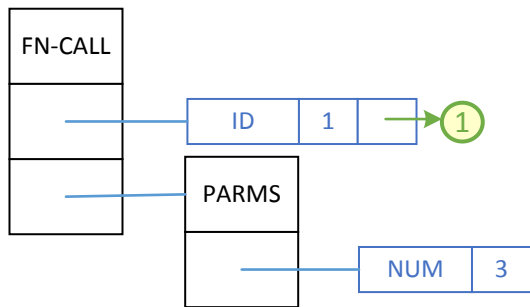


Visit this AST to evaluate it

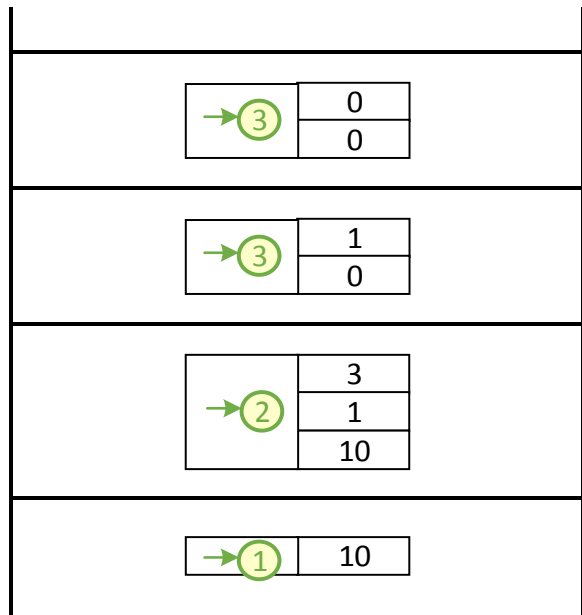
ACTIVATION STACK



FUNCTION CALL EVALUATION (15)



ACTIVATION STACK



ASTS CONSTRUCTED: Note: body ASTs with only one statement have been simplified.

