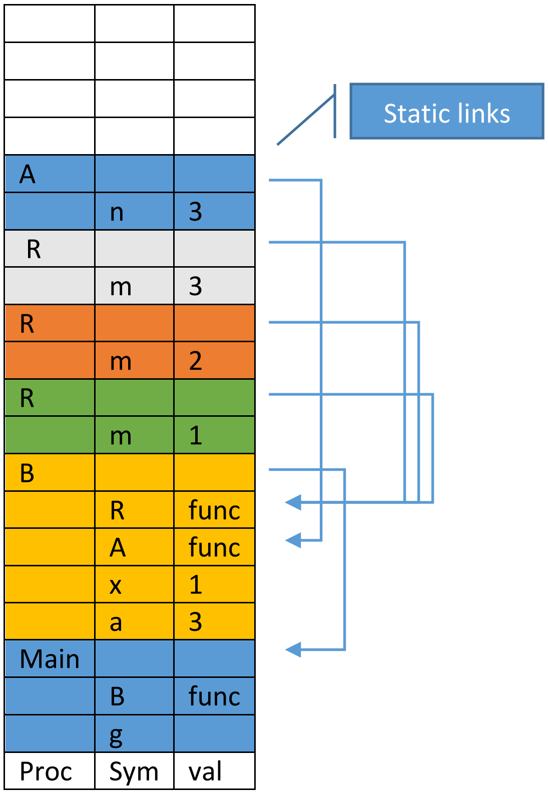
Brent Nix

1) Problem 3.6:

a. The Program prints out:

```
% 9
% 4
% 2
% 3
```

b. The frames on the stack when A has just been called



A		
	n	3
R		
	m	3
R		
	m	2
R		
	m	1
B		
	R	func
	A	func
	x	1
	a	3
Main		
	B	func
	g	
Proc	Sym	val

c. A follows the static links back up the stack to main where it finds 'g' waiting to be given a value

mips assembly for calculating address of g:

```
lw $s0 -4($fp)
lw $s0 -4($s0)
lw $s0 -4($s0)
```

2) Problem 3.14

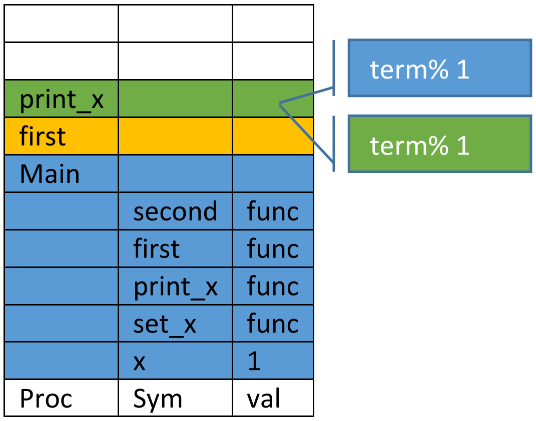
Blue flags are static scope output

Green flags are dynamic scope output

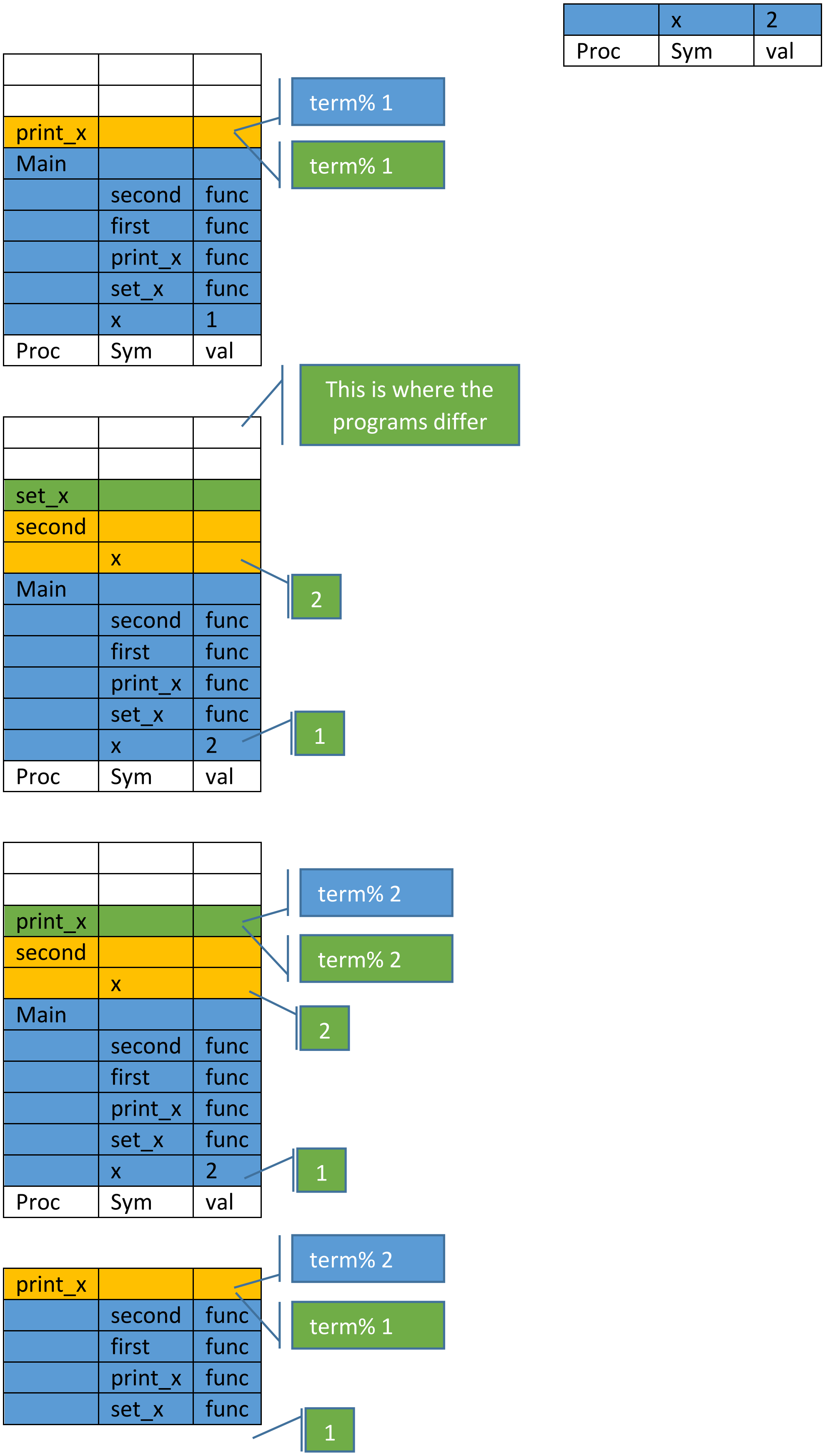
The difference in the two comes from the fact that the dynamic scope language pulls the first available symbol value, whereas static pulls the closest symbol value in its calling tree

set_x		
Main		
	second	func
	first	func
	print_x	func
	set_x	func
	x	0
Proc	Sym	val

set_x		
first		
Main		
	second	func
	first	func
	print_x	func
	set_x	func
	x	1
Proc	Sym	val



print_x		
first		
Main		
	second	func
	first	func
	print_x	func
	set_x	func
	x	1
Proc	Sym	val



	x	2
Proc	Sym	val

print_x		
Main		
	second	func
	first	func
	print_x	func
	set_x	func
	x	1
Proc	Sym	val

term% 1

term% 1

This is where the programs differ

set_x		
second		
	x	
Main		
	second	func
	first	func
	print_x	func
	set_x	func
	x	2
Proc	Sym	val

2

1

print_x		
second		
	x	
Main		
	second	func
	first	func
	print_x	func
	set_x	func
	x	2
Proc	Sym	val

term% 2

term% 2

2

1

print_x		
	second	func
	first	func
	print_x	func
	set_x	func

term% 2

term% 1

1

3) Problem 3.19
    a. both x and y are evaluated to be the global variables, ‘3’ is printed.
    b. It prints ‘4’. at the time x := x + y is evaluated, x is still the global x and y is the y defined in first so 4:=1+3.
    c. It prints ‘1’. At the time x := x + y is evaluated, x is defined in second and y is defined in first. 5 := 2 + 3. but, first returns and both of the locally declared variables are no longer there. because the global value is never changed, it prints the unchanged x as ‘1’.

4) call-by-value:
   y = 6
   z = 18
   x = 42

   call-by-reference:
   y = 42
   z = 18
   x = 210

   call-by-value-result:
   y = 6
   z = 18
   x = 210

   call-by-name:
   y = 42
   z = 18
   x = 42