Caitlyn Goetz

Textbook Assignment 2

February 23, 2016

Edition 3

3.2 Different languages use different allocation techniques for storing data for different reasons. Fortran 77 does not have recursive calls, so the local variables in a given function can't have any more than one instance. Algol and its descendants need to have a stack to have multiple copies of variables. Lisp uses space for local objects with unlimited extent must be allocated on the heap

Pascal Example:

```
x = int
func foo(int x) : integer;
var
   y: int;
begin
   if x==y { return x; }
   y = x;
         return x -foo(x/y);
end;
```

Scheme Example:

```
(define add (lambda(x)
        (lambda (y) (+ xy))))
…
(let ((i (add 2)))
        (i 3))
```

3.4 a) In Java, if a local and global variable have the same name in a program then that variable will be live but not in the scope.

b) In C, a static variable that is declared inside a function is live but cannot be accessed once you are out of that function.

c) In C++, private fields of an object are live but not in scope when we are not inside their method.

3.5 C: There won't be errors and the program should work.

Line 7 will run first:

-> a = b = 1

-> a = 3 and b = 1

-> other variables are out-of-scope

-> a = 1 and b = 2

C#: I believe that this would crash or throw semantic errors. The book states that the scope of name is the entire block, the very first call to print a,b on line 7 calls a before it's declared in that block, and it says in C# that names must be declared before they're used.

Modula-3:

-> a = 3 and b = 3

-> a = 3 and b = 3

-> a = 1 and b = 2

3.7 a) The reverse_list produces a new list and new list nodes. When the program assigns the return value back to L, the program forgets all the work that was just previously done. After some more runs, the program will shut itself down because it runs out of memory.

b) This fixes the previous issue but now the program has dangling references so some data that the program returns to will have changed its value at random times during its running.

3.14 static scoping: 1 1 2 2

dynamic scoping 1 1 2 1

Because the program will either see the global x or the x from the second method.

3.18 static scoping: 3

dynamic scoping with deep binding: 4

dynamic scoping with shallow binding: 1

With the different binding techniques, the program might be accessing the local or global values of x's.