Programming Languages: Homework #3

Due on February 28, 2019 at 9:00am

Erin Keith Section 101

Michael DesRoches

Problem 1

```
(20 pts) Consider the following fragment of code in C:
1.  { int a, b, c;
2.     ...
3.     { int d, e;
4.        ...
5.        { int f;
6.           ...
7.        }
8.        ...
9.     }
10.    ...
11.    { int g, h, i;
12.       ...
13.    }
14.    ...
15. }

Assume that each integer variable occupies four bytes. How much total space
is required for the variables in this code? Justify your answer.
```

Solution

The total space required will be 24 bytes.
If we observe line 1, there are three integers; a, b, and c
3 x 4 = 12. 12 bytes will be allocated. At line 3, two more integers, d and e, are declared. 2 x 4 = 8 bytes will be allocated. Again at line 5(f), 1 x 4 = 4 bytes of memory allocated. 12 + 8 + 4 = 24 bytes.

Memory for (f) is deallocated at line 7. Memory for (d, e) is deallocated at line 9. 4 x 3 = 12 bytes deallocated. 24 - 12 = 12. Line 11 has another 3 variables that require 12 bytes of memory. 12 + 12 = 24 bytes. line 13, scope for (g, h, i) ends. 24 - 12 = 12 bytes. Line 15 deallocates (a, b, c) which brings us back to 0.

Problem 2

(20 pts) Consider the following pseudocode, where procedures Q and R are nested inside procedure P:

```
procedure P (A,B: real)
  X: real

  procedure Q (B,C: real)
    Y: real
     ...
  procedure R (A,C: real)
    Z: real
     ...                // (*)
  ...
```

Assuming static scope, what is the referencing environment (i.e., what names are known, and what do they refer to at the location marked by (*)?

Solution
Z, A (parameter to R), C (parameter to R), B (parameter to P), X, R, Q, P. P's parameter to A and Q's parameters to B, C and Q's local variable Y are not in scope.

Problem 3

```
(30 pts) Consider the following pseudocode:
1. procedure main
2.  a: integer := 1
3.  b: integer := 2
4.  procedure middle
5.    b: integer := a
6.    procedure inner
7.    print a, b
8.    a: integer := 3
9.    // body of middle
10.   inner()
11.   print a, b
12. //body of main
13. middle()
14. print a, b

(a) Suppose this was code for a language with the declaration-order rules of C (but with
nested subroutines) - that is, names must be declared before use, and the scope of a name
extends from its declaration through the end of the block. At each print statement,
indicate which declarations of a and b are in the referencing environment. What does the
program print (or will the compiler identify static semantic errors)?
(b) Repeat the exercise for the declaration-order rules of C# (names must be declared
before use, but the scope of a name is the entire block in which it is declared).
(c) Repeat the exercise for the declaration-order rules Modula-3 (names can be declared
in any order, and their scope is the entire block in which they are declared).
```

Solution
With C rules, line 7 refers to the a and b declared on lines 2 and 5, respectively; line 11 refers to the a and b declared on lines 8 and 5, respectively; and line 14 refers to the a and b declared on lines 2 and 3, respectively.

Line 7 will print 1 and 1
Line 11 will print 3 and 1
Line 14 will print 1 and 2

With C# rules
Line 7 will print 3 and 1
Line 11 will print 3 and 1
Line 14 will print 3 and 1

With Modula-3
Line 7 will print 1 and 1
Line 11 will print 1 and 2
Line 14 will print 1 and 2

Problem 4

```
(30 pts) Consider the following pseudocode:
x: integer := 1
y: integer := 2
procedure add
x := x + y
procedure second (P: procedure)
x: integer := 2
P()
procedure first
y: integer := 3
second(add)
// main program
first()
write integer(x)

(a) What does this program print if the language uses static scoping?
(b) What does it print if the language uses dynamic scoping with deep binding?
(c) What does it print if the language uses dynamic scoping with shallow binding?
```

Solution
(a) 3
(b) 4
(c) 1