**Concepts of Programming Languages, CSCI 305, Fall 2021**
**Homework #9, complete by Nov. 19**

Exercises 3.5, 3.6, 3.7 and 3.13 from the text (pages 167-171)

1. Exercise 3.5 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
```

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)?

1 1 (print at line 7 uses ‘a’ declared in line 2 and ‘b’ declared in line 5)
1 1 (print at line 11 uses ‘a’ declared in line 2 and ‘b’ declared in line 5)
1 2 (print at line 14 uses ‘a’ declared in line 2 and ‘b’ declared in line 3.)

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).

> The ‘a’ declared at line 8 is known throughout the block, so it is known by the print statement in line 7. It is known, but not yet assigned, so an error results from the statement ‘print a, b’ at line 7.

Repeat the exercise for Modula-3 (names can be declared in any order, and their scope is the entire block in which they are declared).

> 3 1 (print at line 7 uses ‘a’ declared in line 2 and ‘b’ declared in line 5)
> 1 1 (print at line 11 uses ‘a’ declared in line 2 and ‘b’ declared in line 5)
> 1 2 (print at line 14 uses ‘a’ declared in line 2 and ‘b’ declared in line 3.)

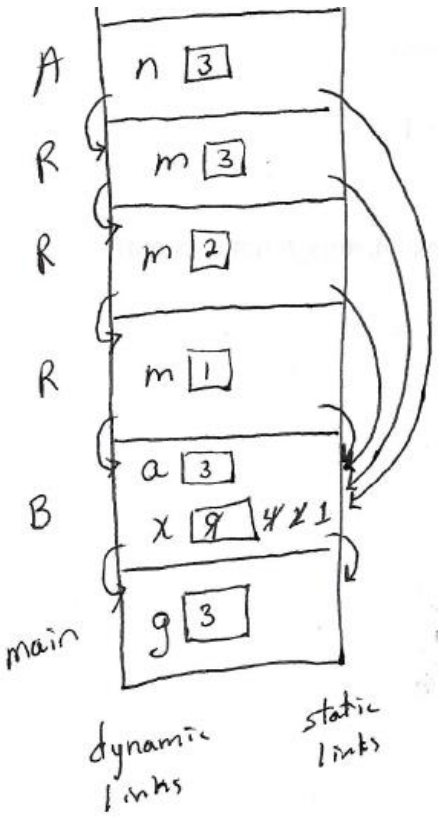
2. Exercise 3.6 Consider the following pseudocode, assuming nested subroutines and static scope:

```
procedure main()
    g : integer

    procedure B( a :integer)
        x : integer

        procedure A( n :integer)
            g := n


        procedure R( m :integer)
            write_integer(x)
            x := 2 --- integer division (truncate fraction)
            if x > 1
                R(m+1)
            else
                A(m)
            end


        ----- body of B
        x := a x a
        R(1)


    ----- body of main
    B(3)
    write_integer(g)
```

a. What does this program print?

9 4 2 3

b. Show the frames on the stack when A has just been called. For each frame, show the static and dynamic links.




c. Explain how A finds g.

It dereferences its static link to find the stack frame of B. Within this frame it finds B's static link, at a statically known offset. It dereferences that to find the stack frame of main. In the main stack frame, at a statically known offset, it finds g.