LECTURE 15
Names, Scopes, and Bindings: Example Problems
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
  if x > 1:
    R(m+1)
  else:
    A(m)
  -- body of B
x := a * a
R(1)
-- body of main
B(3)
write_integer(g)

Assume the language uses nested subroutines and static scoping.

What does this program print?
Assume the language uses nested subroutines and static scoping.

What does this program print?

9 4 2 3
Assume the language uses nested subroutines and static scoping.

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

How does A find g?

*dynamic links reference the caller of a subroutine.*
Assume the language uses nested subroutines and static scoping.

```plaintext
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
        if x > 1:
          R(m+1)
        else:
          A(m)
          -- body of B
      x := a * a
      R(1)
      -- body of main
      B(3)
    write_integer(g)
```
A finds g by first looking to see if it is defined locally. Since it is not, A traverses its static link to B. Since g is not defined in B, it traverses B’s static link to Main, where g is defined.
EXERCISE 2

Consider the following pseudocode. What is the referencing environment at the location marked by (*)?

```
procedure P(A, B : real)
    X:real
procedure Q(B, C : real)
    Y:real
    ...
procedure R(A, C: real)
    Z:real
    ...
    --(*)
    ...
```
EXERCISE 2

Z, A (parameter to R), C (parameter to R), B (parameter to P), X, R, Q, P.
Note: P’s parameter A as well as Q’s parameters B and C and Q’s local variable Y are not in scope.

```plaintext
procedure P(A, B : real)
  X:real
procedure Q(B, C : real)
  Y:real
  ...
procedure R(A, C: real)
  Z:real
  ...
  --(*)
  ...
```
What does the program print if the language uses static scoping? What does it print with dynamic scoping?

```
x:integer     --global variable

procedure set_x(n: integer)
  x := n

procedure print_x
  write_integer(x)

procedure first
  set_x(1)
  print_x

procedure second
  x:integer
  set_x(2)
  print_x

set_x(0)
first()
print_x
second()
print_x
```
EXERCISE 3

What does the program print if the language uses static scoping?

$1\ 1\ 2\ 2$
EXERCISE 3

What does it print with dynamic scoping?

1 1 2 1
Assume the language uses dynamic scoping. What does the program print if the language uses shallow binding? What does it print with deep binding?

```pascal
x:integer     -- global variable

procedure set_x(n: integer)
  x := n

procedure print_x
  write_integer(x)

procedure foo(S, P: function, n: integer)
  x:integer := 5
  if n in {1,3}
    set_x(n)
  else
    S(n)
  if n in {1,2}
    print_x
  else
    P

set_x(0); foo(set_x, print_x, 1); print_x
set_x(0); foo(set_x, print_x, 2); print_x
set_x(0); foo(set_x, print_x, 3); print_x
set_x(0); foo(set_x, print_x, 4); print_x
```
Assume the language uses dynamic scoping.
What does the program print if the language uses shallow binding?

1 0
2 0
3 0
4 0
What does it print with deep binding?

10
52
00
44
EXERCISE 5

What does the program print if the language uses static scoping?

What does the language print if it uses dynamic scoping with deep binding?

What does the language print if it uses dynamic scoping with shallow binding?
What does the program print if the language uses static scoping?

```
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)

first()
write_integer(x)
```
What does the language print if it uses dynamic scoping with deep binding?

```
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)

first()
write_integer(x)
```
What does the language print if it uses dynamic scoping with shallow binding?

```plaintext
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)

first()
write_integer(x)
```