Daniel Rao

CS-354

P167
3.1

Indicate binding time:

(a) The number of built-in functions:
language design time.

(b) The variable declaration that corresponds to a particular variable reference:
Compile time.

(c) The maximum length allowed for a constant (literal) character string
Language implementation time.

(d) The referencing environment for a sub-routine that is passed as parameter:
compile-time.

(e) The address of an particular library routine:

loader time

(f) The total amount of space occupied by a code and data:

Execution time

p167 3.4

Give 3 example: Variable use but not in scope.

Example 1:

```
class test {
    private int x = 50;
    public static void main(String args[]) {
        System.out.println(x);
    }
}
```

x not accessible here.

Example 2:

```
class test {

    public static void main (String args[]) {

        int x = 50;
        System.out.print(x);
        doStuff();
    }

    public static doStuff() {

        System.out.print(x);

    }
}
```

x is accessible.

x is not accessible here.

Example 3:

```
class test {

    public static void main (String args[]) {

        for (int x = 0; x < 10; x++) {

        System.out.print(x);
        }

    System.out.print(x);

    }
}
```

x is accessible.

x is not accessible.

Output : 1 1 3 1 1 2

For C.

For C:

(a) int a = 1
(b) int b = 2
(c) middle()
in main()

↳ int b = a (b = 1)
↳ int a = 3 ← procedure inner : prints a b
↳ Inner inner() → called.
↳ print a b
↳ print a, b

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

(d) print a, b

For C#

the middle() method on line 5 will result in an error because b is assigned a value before a declaration. This type of variable binding is syntactically incorrect and will result in an error. In these programming languages variables can't be used before being declared. So, within the scope of middle variable b has no idea of what variable a is.

For Module-3 :

int a = 1
int b = 2
middle()
- in main()

↳ int b = a (b = 1)
int a = 3
inner() → prints 3 1
print 3 1
prints 1 2

Output
3 1 3 1 1 2

↳ Doesn't care what order variable are declared.

p 169
3.6 a

Consider the following pseudocode, assuming restrict sub-routine and static scope what does the following program print.

procedure main ( )

int g ;

B (a = 3)

int x ;

x = a x a = 9

R (m = 1)

print (x) ✓ : (9)

x = x/2 = 9/2 = 4

(x > 1) = (4 > 1) True

R (m = m+1) = R (2)

print (4) ✓

x = x/2 = 4/2 = 2

(x > 1) = (2 > 1) True

R (m = m+1) = R (3)

print (2)

x = 2/2 = 1

(x > 1) = (1 > 1) False

A (m)

A (n = m = 3)

g = 3

print (3)

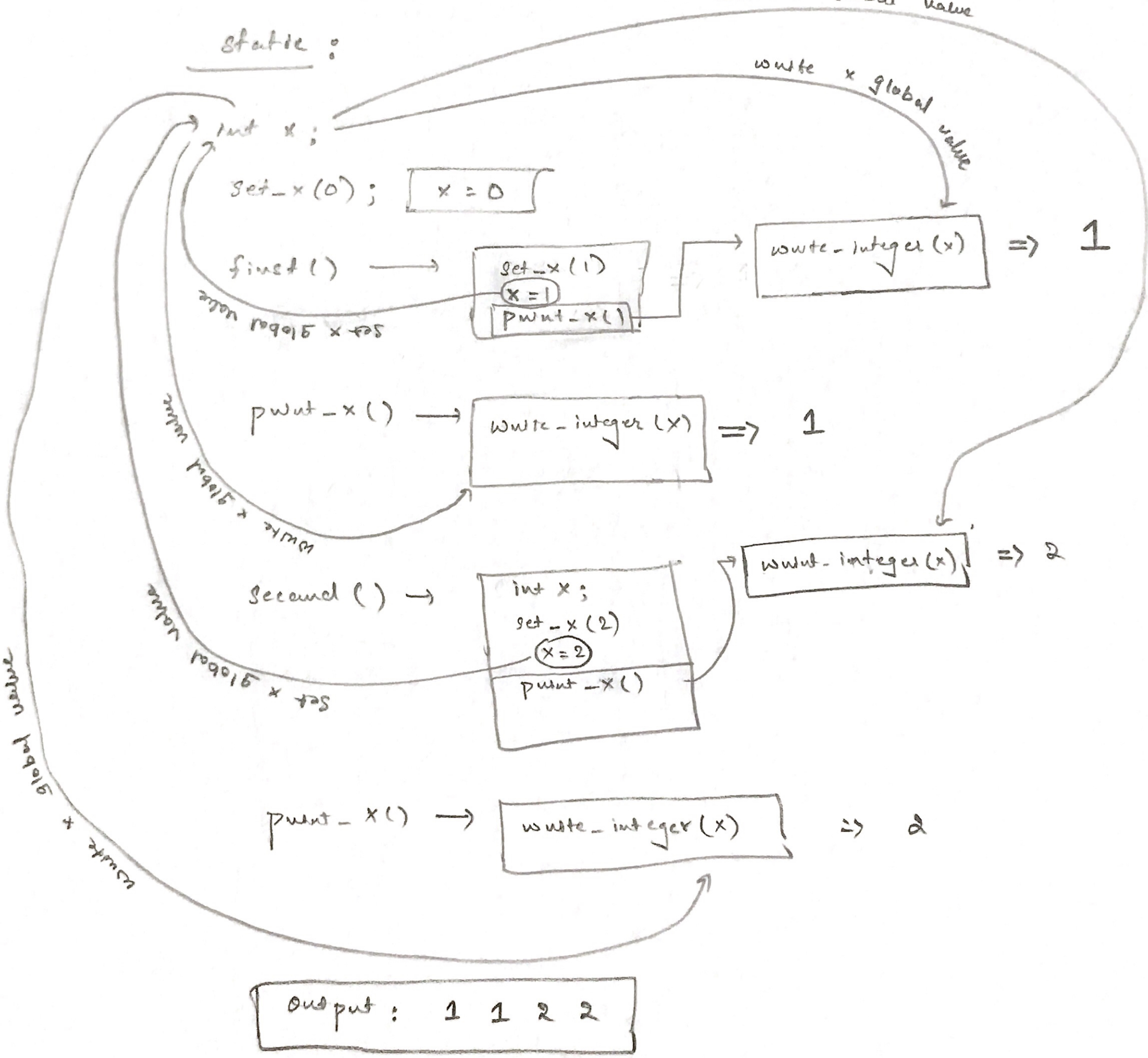
Output :

9 4 2 3

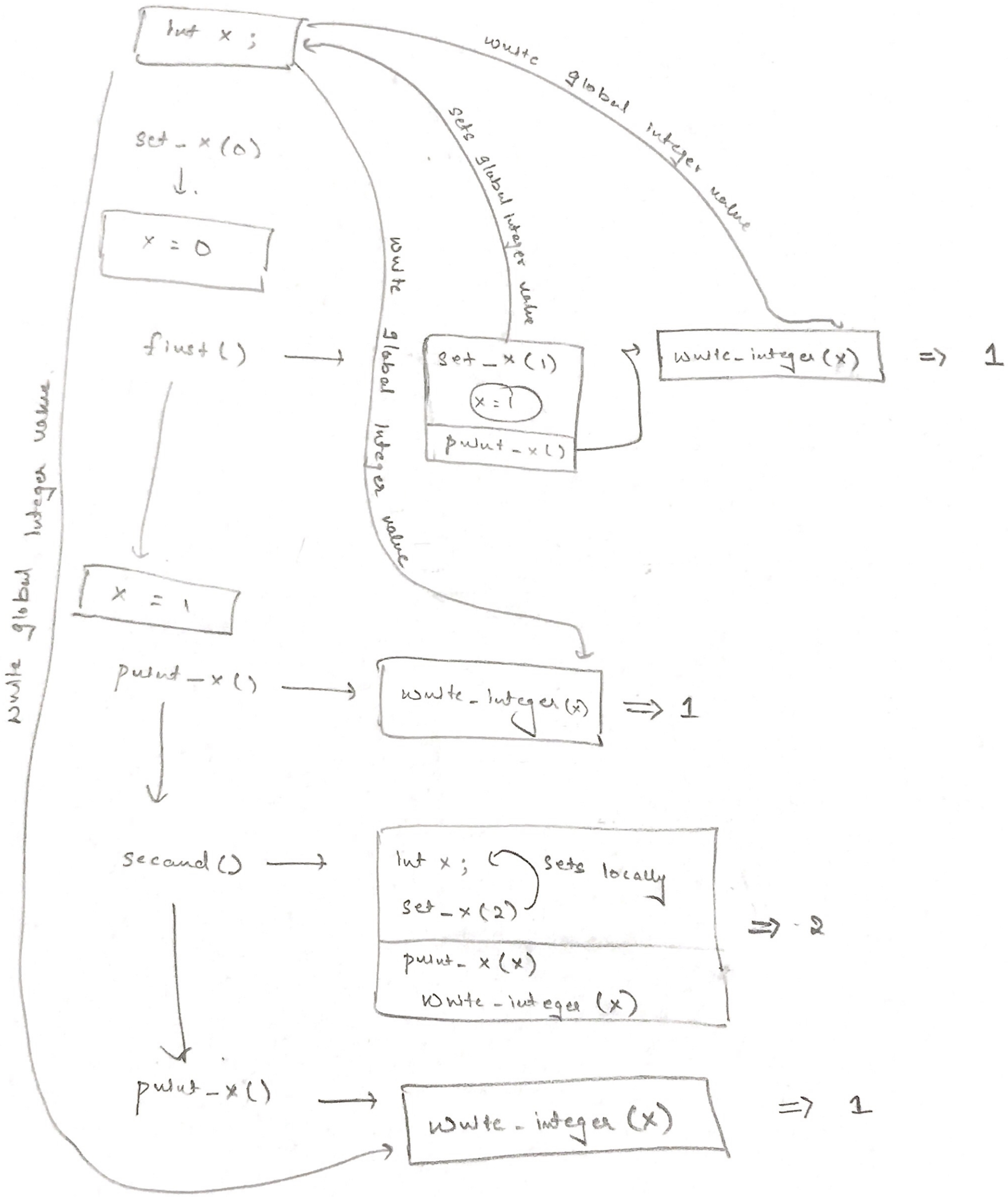
p 171
3.14

Consider the following pseudocode:

What does the program print if the programming language uses static scoping? What does it print with dynamic scoping? Why?



Dynamic Scoping



Explanation:

The procedure 'second' contains a variable 'x'.
When 'second' calls set_x using static scoping,
the 'global' x is set; while in dynamic
scoping 'local' x is set. Thus, in the dynamic
case global x is not changed when 'sd_x(2)'
is called.