First Code Snippet: Dynamic Scope

Hayden Coble
hcoble@my.athens.edu
CS 414
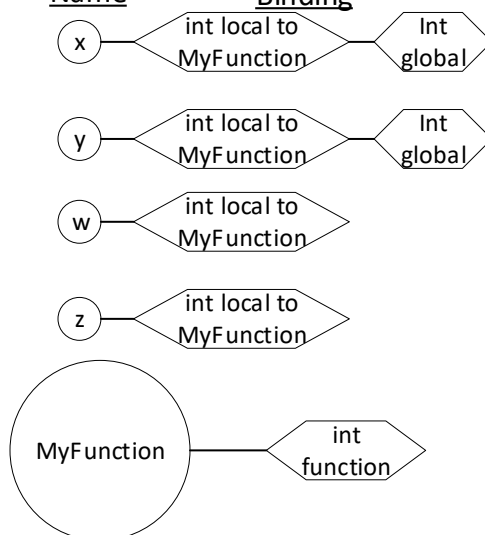
Line 3

Name   **Binding**

- x — Int global
- y — Int global
- MyFunction — int function

Lines 5, 8, 11

Name   **Binding**

- x — int local to MyFunction — Int global
- y — int local to MyFunction — Int global
- w — int local to MyFunction
- z — int local to MyFunction
- MyFunction — int function
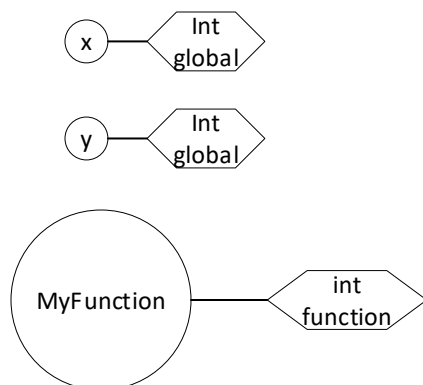
Each table represents the code before executing the line used to label it.

Line 7

Name   **Binding**

- x — int local to MyFunction — Int global
- y — Int local to block nested in MyFunction — int local to MyFunction — Int global
- w — int local to MyFunction
- z — int local to MyFunction
- MyFunction — int function

Line 10

Name   **Binding**

- x — int local to MyFunction — Int global
- w — Int local to block nested in MyFunction — int local to MyFunction — Int global
- y — int local to MyFunction
- z — int local to MyFunction
- MyFunction — int function
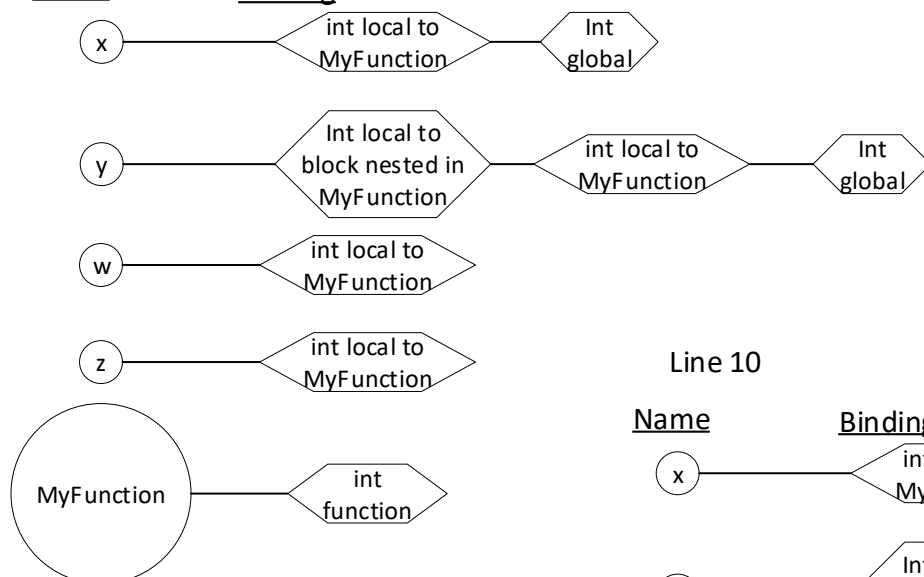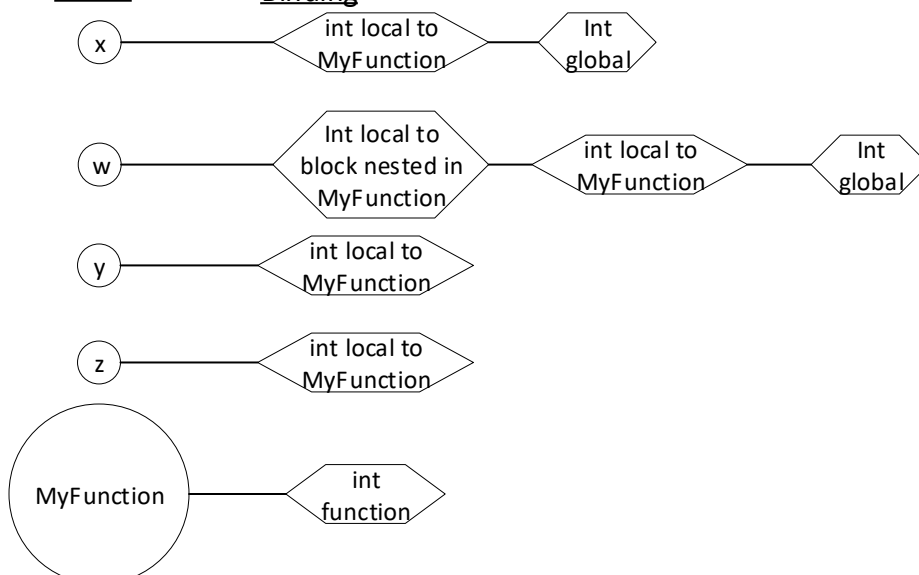
Lexical Scope

First Snippet

Name   **Binding**

x — int local to MyFunction — Int global

y — Int local to block nested in MyFunction — int local to MyFunction — Int global

w — Int local to block nested in MyFunction — int local to MyFunction

z — int local to MyFunction

MyFunction — int function

Second Snippet

Name   **Binding**

x — Int = z (local to nested block in MyFunction)*, local to block nested in MyFunction — Int, argument, local to MyFunction — Int = 137 global

z — Int = y (argument local to MyFunction)* local to block nested in MyFunction — Int = 42 global

y — Int = x (local to block nested in MyFunction)*, local to block nested in block in MyFunction — Int, argument, local to MyFunction

MyFunction — int function
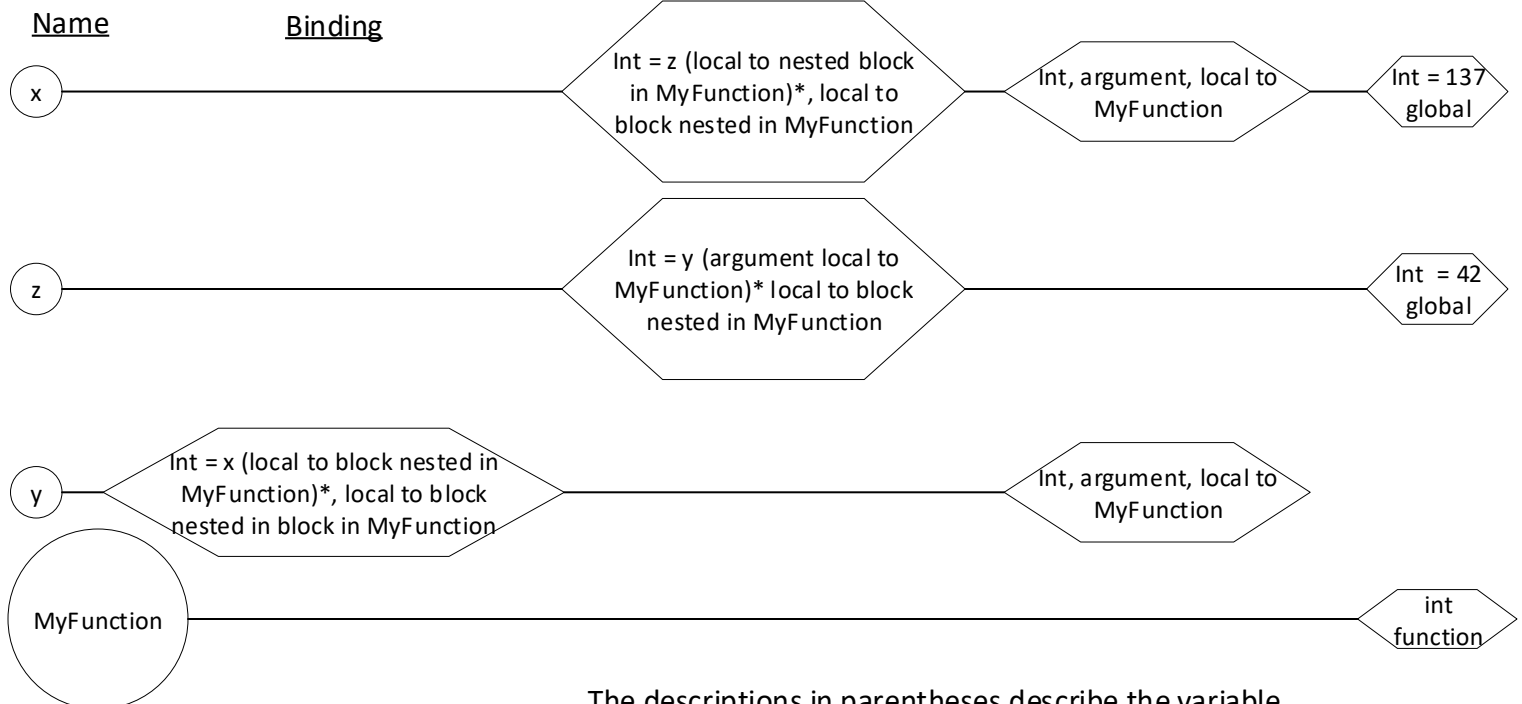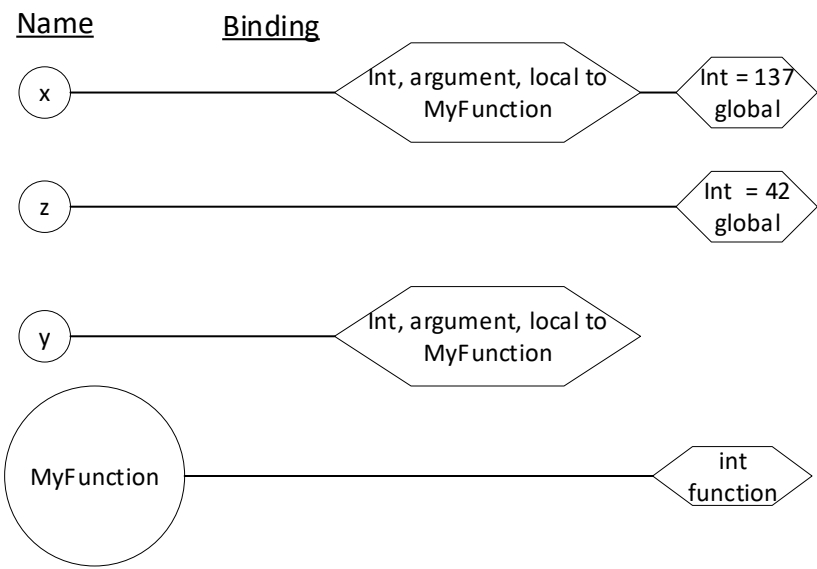
The descriptions in parentheses describe the variable right of the equal sign. The description following and separated by a comma describes the variable its shape bound to.

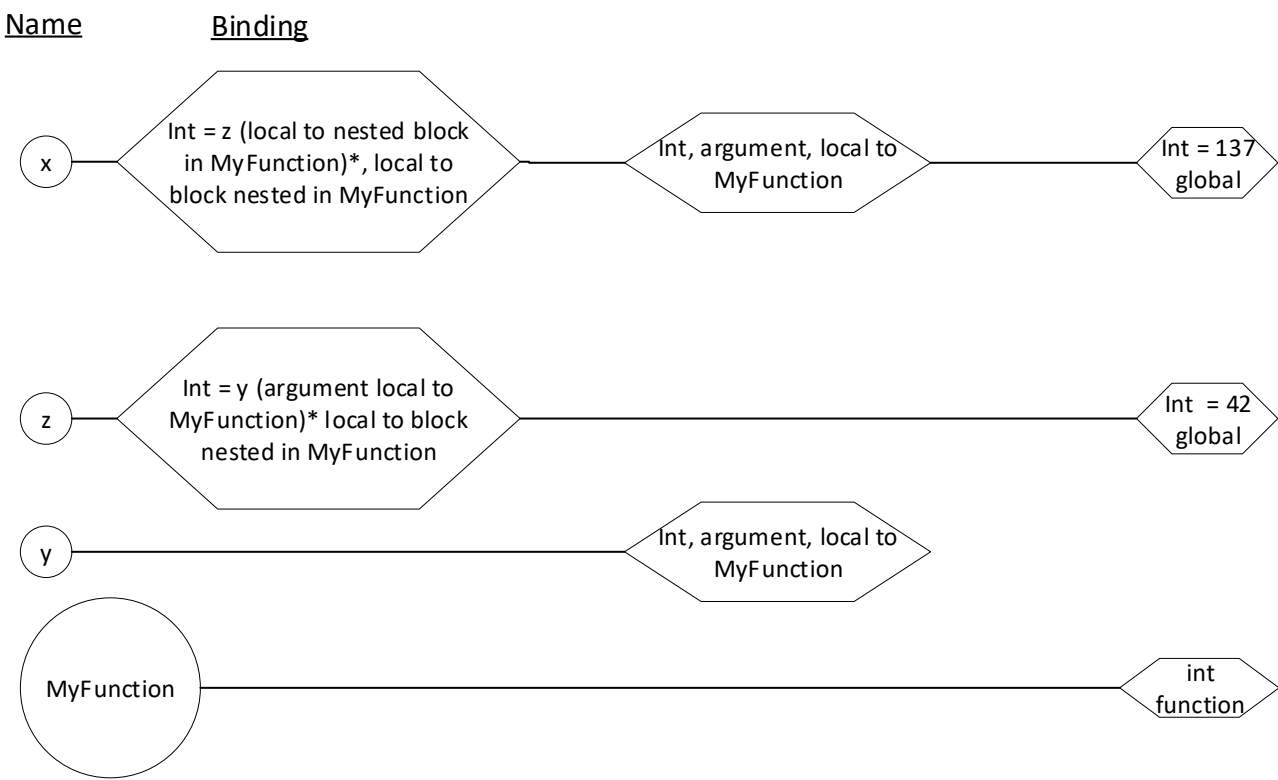*Everything noted with an asterisk is equal to the argument y passed to MyFunction

The following tables also use this notation

Second Code Snippet: Dynamic Scope

Lines 3 and 17

Name   Binding

x ⎯⎯⎯⎯⎯ Int, argument, local to MyFunction ⎯⎯⎯ Int = 137 global

z ⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯ Int = 42 global

y ⎯⎯⎯⎯⎯ Int, argument, local to MyFunction

MyFunction ⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯ int function

Lines 8 and 15

Name   Binding

x ⎯⎯⎯ Int = z (local to nested block in MyFunction)*, local to block nested in MyFunction ⎯⎯⎯ Int, argument, local to MyFunction ⎯⎯⎯ Int = 137 global

z ⎯⎯⎯ Int = y (argument local to MyFunction)* local to block nested in MyFunction ⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯ Int = 42 global

y ⎯⎯⎯⎯⎯ Int, argument, local to MyFunction

MyFunction ⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯ int function

Second Code Snippet: Dynamic Scope

Lines 10

<u>Name</u>   <u>Binding</u>

x ———————————————————— Int = z (local to nested block in MyFunction)*, local to block nested in MyFunction ———————— Int, argument, local to MyFunction ———— Int = 137 global

z ———————————————————— Int = y (argument local to MyFunction)* local to block nested in MyFunction ———————————————————— Int = 42 global

y ———— Int = x (local to block nested in MyFunction)*, local to block nested in block in MyFunction ———————————————— Int, argument, local to MyFunction

MyFunction ———————————————————————————— int function

Final Question

Assuming that dynamic scoping is used, what is output by the following program?

```
void main() {
  int x = 0;
  f1();
  g();
  f2();
}

void f1() {
  int x = 10;
  g();
}

void f2() {
  int x = 20;
  f1();
  g();
}

void g() {
  print(x);
}
```

10
0
10
20