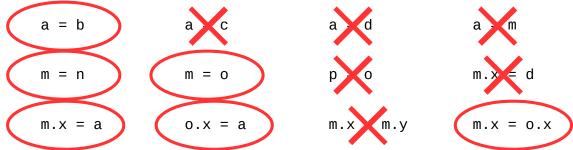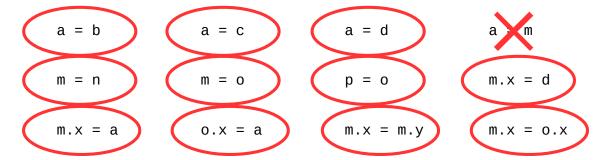# CS 430 Types (Solution to Part 4 only)

Suppose the following declarations have been made in a C-like language. For each context, circle all of the assignments that are valid, and **cross out all that are not valid**.

```
typedef float inches;
typedef float feet;
typedef struct { inches x; feet y; } box;
typedef struct { inches x; feet y; } bin;

inches a, b;
feet c;
float d;
struct { inches x; feet y; } m, n;
box o;
bin p;
```

18. Assume assignments require name equivalence unless at least one type is anonymous, in which case they only require structure equivalence.

(a = b)	a ✗ c	a ✗ d	a ✗ m
(m = n)	(m = o)	p ✗ o	m.x ✗ d
(m.x = a)	(o.x = a)	m.x ✗ m.y	(m.x = o.x)

19. Assume assignments only require structure equivalence, regardless of type aliases.

(a = b)	(a = c)	(a = d)	a ✗ m
(m = n)	(m = o)	(p = o)	(m.x = d)
(m.x = a)	(o.x = a)	(m.x = m.y)	(m.x = o.x)

20. Assume assignments require name equivalence for primitive types and their aliases but permit structure equivalence for non-primitive types.

(a = b)	a ✗ c	a ✗ d	a ✗ m
(m = n)	(m = o)	(p = o)	m.x ✗ d
(m.x = a)	(o.x = a)	m.x ✗ m.y	(m.x = o.x)