

## CS 430 Types

### Parts 1-3 (only those problems that can't be confirmed using PythonTutor)

2. What is the name of the "\*" operator in C (e.g., on the two lines after location A), and what does it do?

**The dereference operator accesses memory through a pointer variable.**

8. What is the program output?

**N/A - the behavior is undefined because the memory pointed to by 'p' has been freed.**

9. Does the program leak memory? Are there any dangling pointers?

**It does not leak memory, but there are two dangling pointers ('a' and 'p') after the call to "free".**

10. Describe how *mark-sweep garbage collection* would help deal with a problem identified in #9.

**It would remove the need to explicitly free memory, keeping it accessible for the dereference. Instead of an explicit free, the garbage collector would periodically reclaim inaccessible allocations.**

11. Describe how *reference counters* would help deal with a problem identified in #9.

**It would remove the need to explicitly free memory, keeping it accessible for the dereference. Each object would maintain a count indicating how many references refer to it. The count is incremented when aliases are made, and decremented when references go out of scope. When a count reaches 0, its resources are freed.**

12. What is a *reference type* and how is it different from a *pointer type*?

**A pointer refers to a specific address in memory (regardless of whether it's a valid object) while a reference refers to a specific object in memory.**

13. Does C have reference types, pointer types, or both? What about C++, Java, and Ruby?

**C: pointers only  
Java: references only**

**C++: pointers and references  
Ruby: references only**

15. What is the minimum number of bytes necessary to store the struct variable ("v") in the above program?

**4 (int) + 1 (char) + 8 (double) = 13 bytes total**

16. Suppose that on a particular system there are X different possible integers, Y different possible characters, and Z different possible double-precision numbers. As a function of X, Y, and Z, what is the total number of different possible values for the struct variable ("v")?

**$X * Y * Z$**

17. Change "struct" to "union" in the first line. How does this change what is stored in memory? What is the minimum number of bytes required to store the union variable?

**Only a single value is stored. The minimum storage is now 8 bytes (the maximum of 4, 1, and 8).**

18. As a function of X, Y, and Z (as defined in #15), what is the total number of different possible values for the union variable ("v")?

**The total number of possible values is  $X + Y + Z$ .**

#### Part 4: Type Equivalence

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;
```

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

<input checked="" type="checkbox"/> a = b	<input checked="" type="checkbox"/> a = c	<input checked="" type="checkbox"/> a = d	<input checked="" type="checkbox"/> a = m
<input checked="" type="checkbox"/> m = n	<input checked="" type="checkbox"/> m = o	<input checked="" type="checkbox"/> p = o	<input checked="" type="checkbox"/> m.x = d
<input checked="" type="checkbox"/> m.x = a	<input checked="" type="checkbox"/> o.x = a	<input checked="" type="checkbox"/> m.x = m.y	<input checked="" type="checkbox"/> m.x = o.x

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

<input checked="" type="checkbox"/> a = b	<input checked="" type="checkbox"/> a = c	<input checked="" type="checkbox"/> a = d	<input checked="" type="checkbox"/> a = m
<input checked="" type="checkbox"/> m = n	<input checked="" type="checkbox"/> m = o	<input checked="" type="checkbox"/> p = o	<input checked="" type="checkbox"/> m.x = d
<input checked="" type="checkbox"/> m.x = a	<input checked="" type="checkbox"/> o.x = a	<input checked="" type="checkbox"/> m.x = m.y	<input checked="" type="checkbox"/> m.x = o.x

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

<input checked="" type="checkbox"/> a = b	<input checked="" type="checkbox"/> a = c	<input checked="" type="checkbox"/> a = d	<input checked="" type="checkbox"/> a = m
<input checked="" type="checkbox"/> m = n	<input checked="" type="checkbox"/> m = o	<input checked="" type="checkbox"/> p = o	<input checked="" type="checkbox"/> m.x = d
<input checked="" type="checkbox"/> m.x = a	<input checked="" type="checkbox"/> o.x = a	<input checked="" type="checkbox"/> m.x = m.y	<input checked="" type="checkbox"/> m.x = o.x