CS 430 Types (full lab located at `https://lam2mo.github.io/cs430-sp22/m07/lab.html`)

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?

8. What is the program output?

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

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

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

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

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

C:	C++:
Java:	Ruby:

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

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")?

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?

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")?

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.

`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 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`

21. 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`