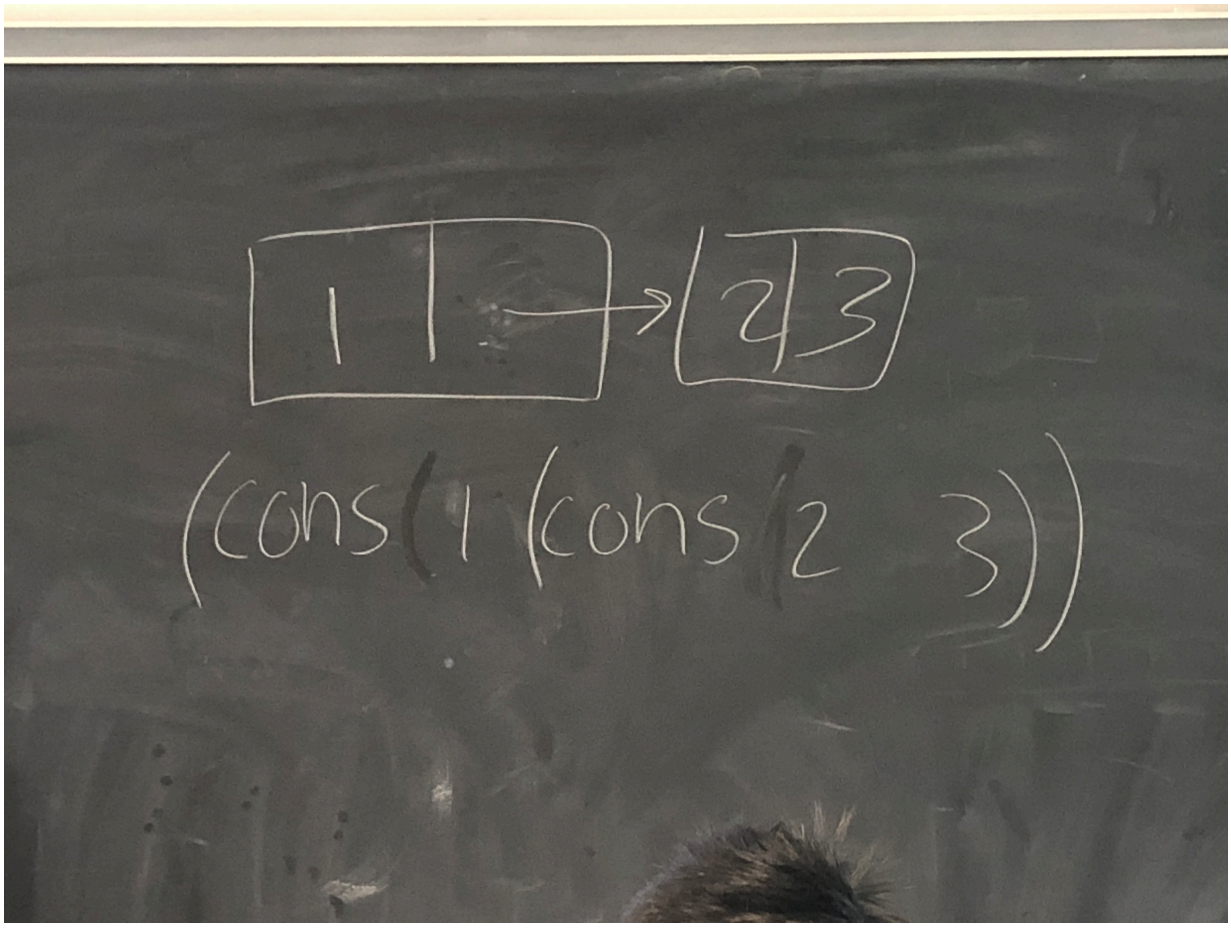
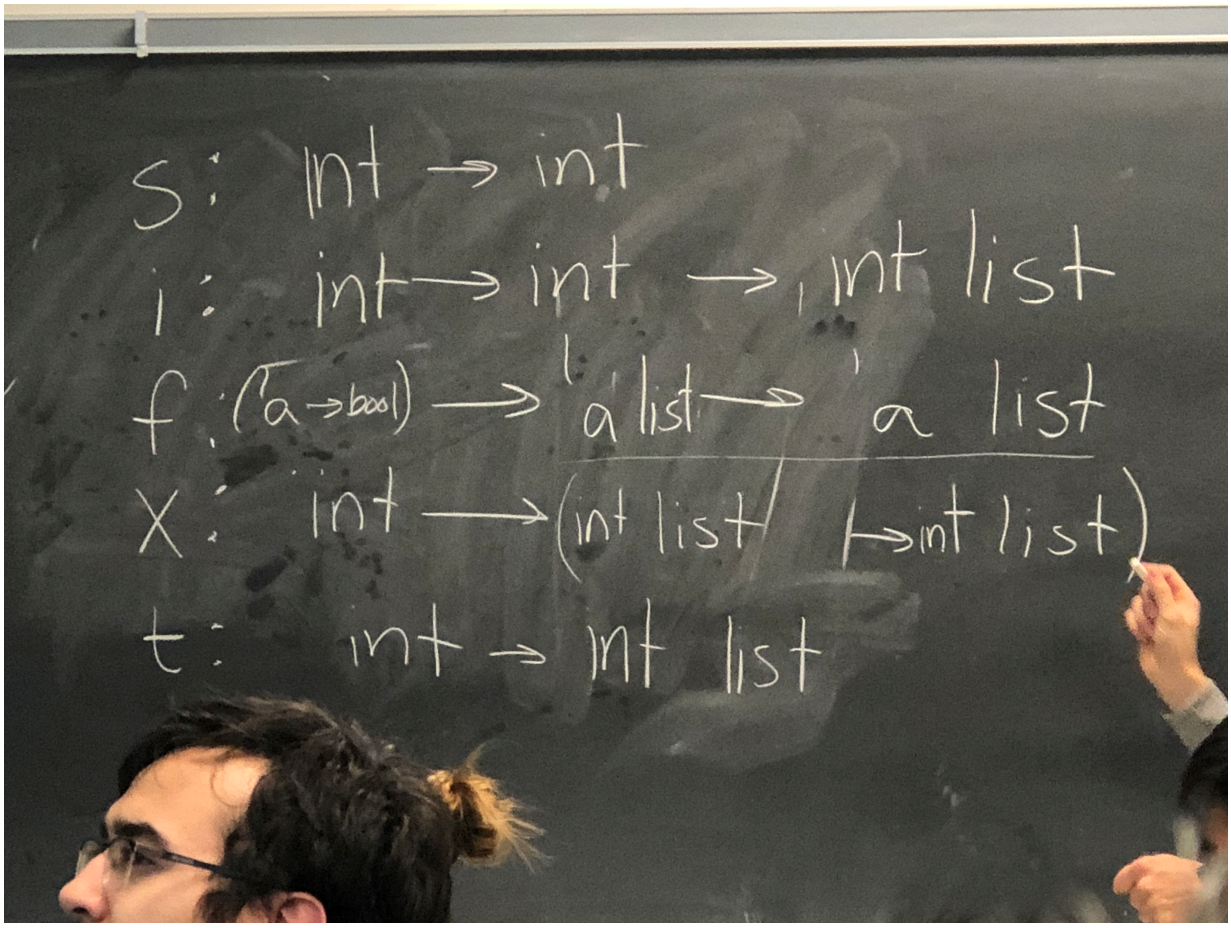
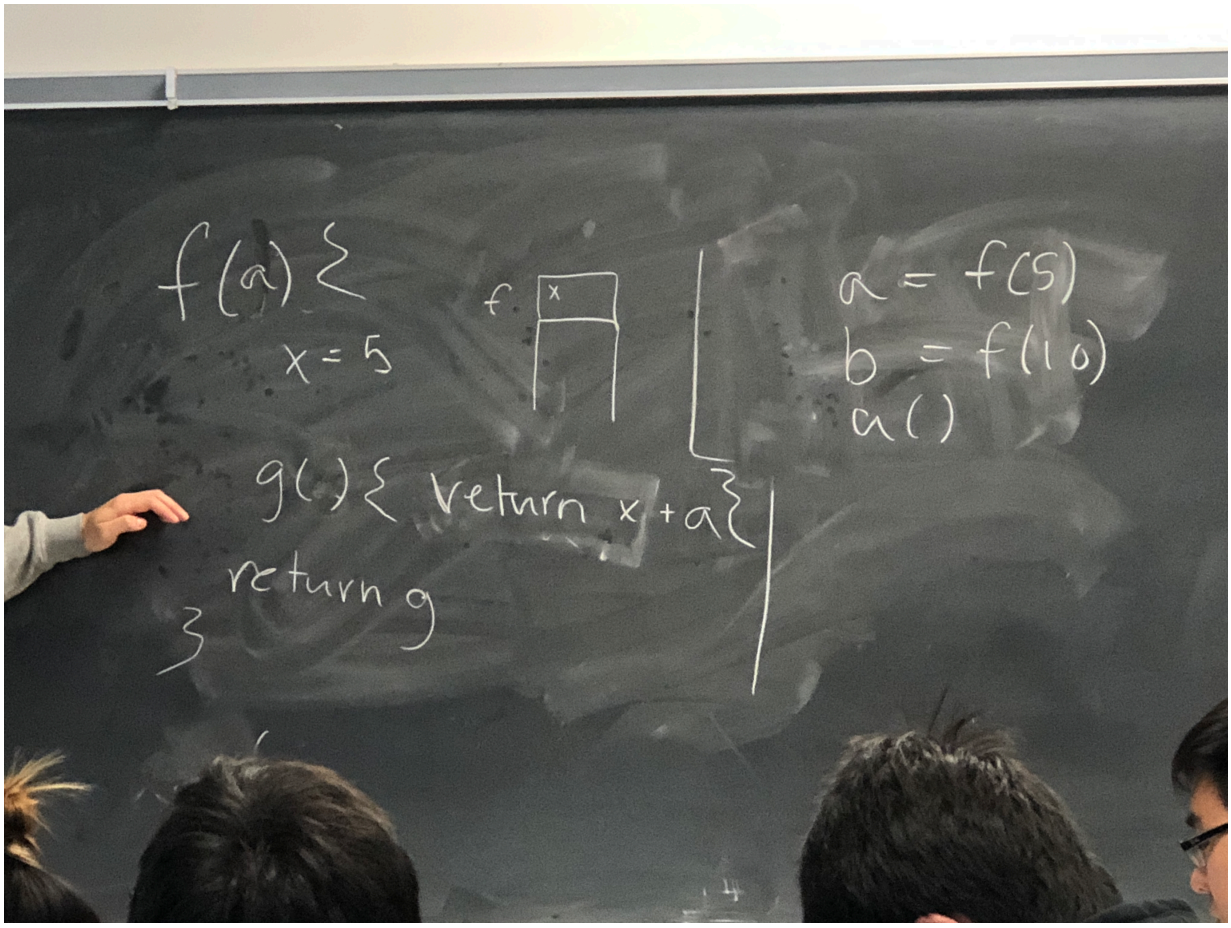
CS131 Last Discussion

if you type check something that has 2 integers, then



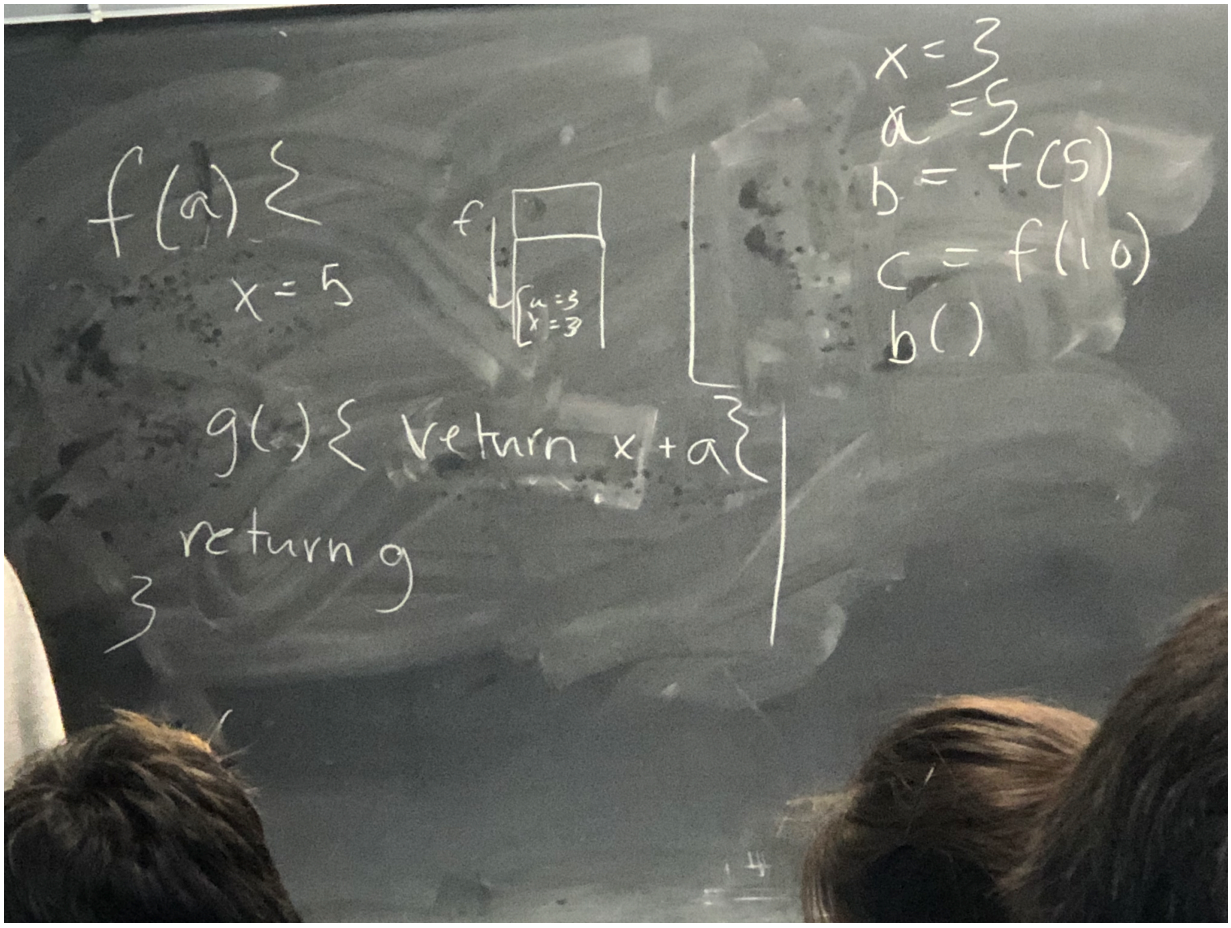


Static scoping semantics need heap because function stack can leave in closure, but need to keep track of variables inside closure even after the function returns

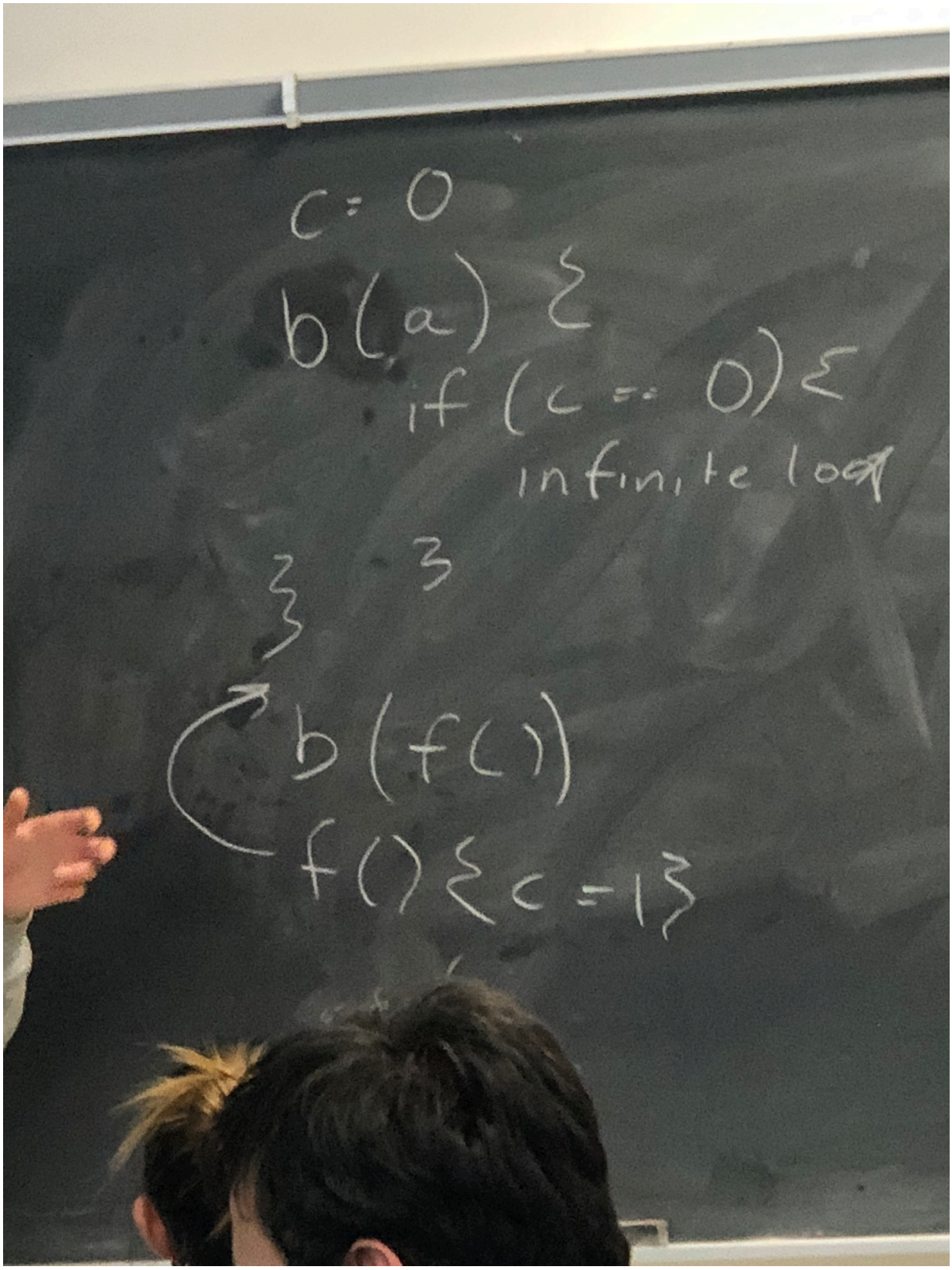


Dynamic scoping
Determines scopes at runtime.
Closure problem is ok in dynamic scoping

- looks up call stack
- downside: if you haven't defined variables at highest level, you might get runtime scoping errors



Call by need: evaluate it when you need it only when it's used in the function, not before you pass it in as an argument (lazy evaluation), store it and then look it up next time
call-by name: evaluate it every single time an argument is used inside the function being used, DOES NOT store and evaluates greedily every single time a value is used
call-by-value evaluates right before call (eager), then pass value which is stored.



Only works for call by value because f needs to be evaluated first, and c will not be 0. but if we use call by name or call by need, the shit in the args wont be evaluated yet and we'll get an infinite loop