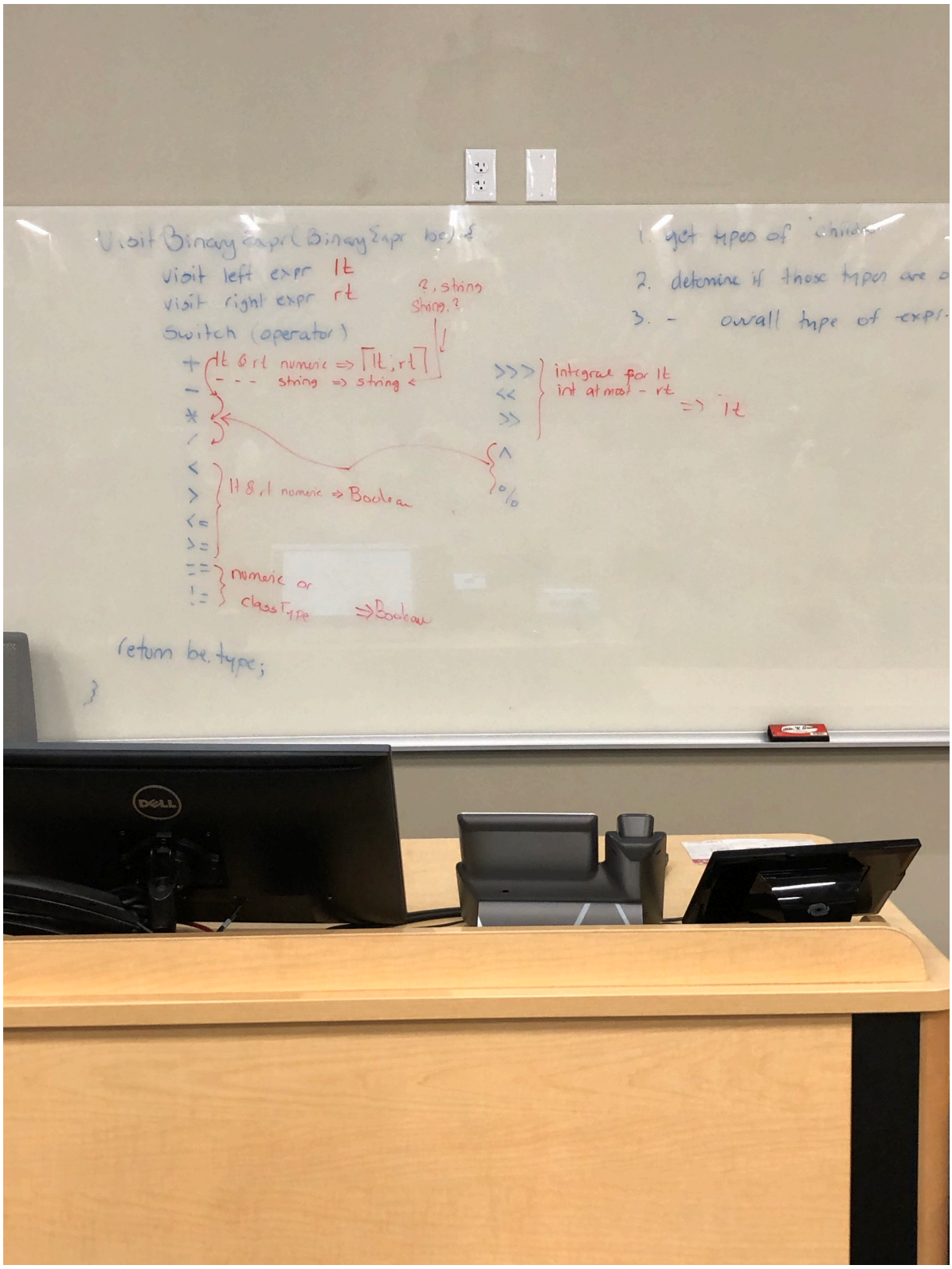
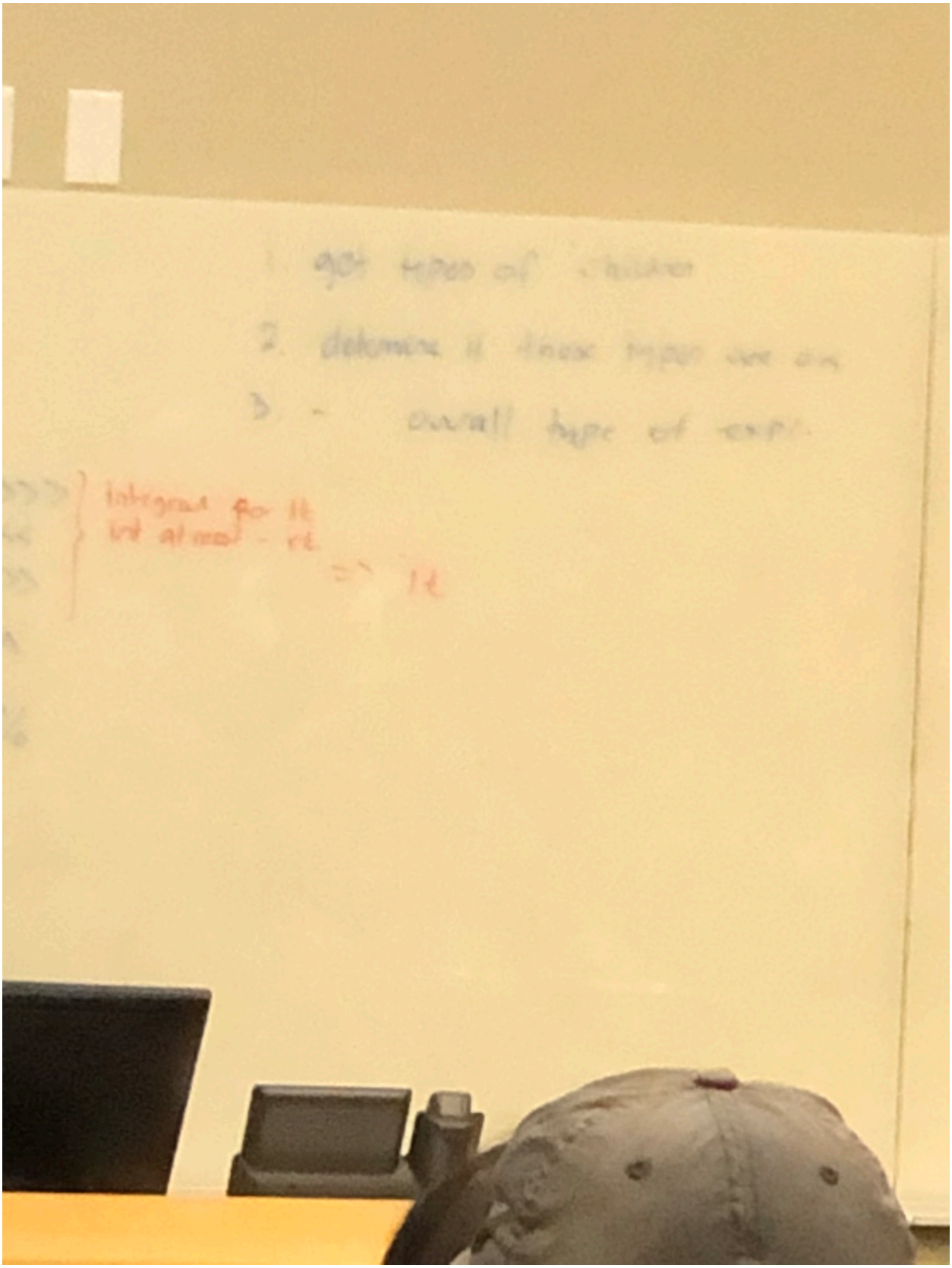
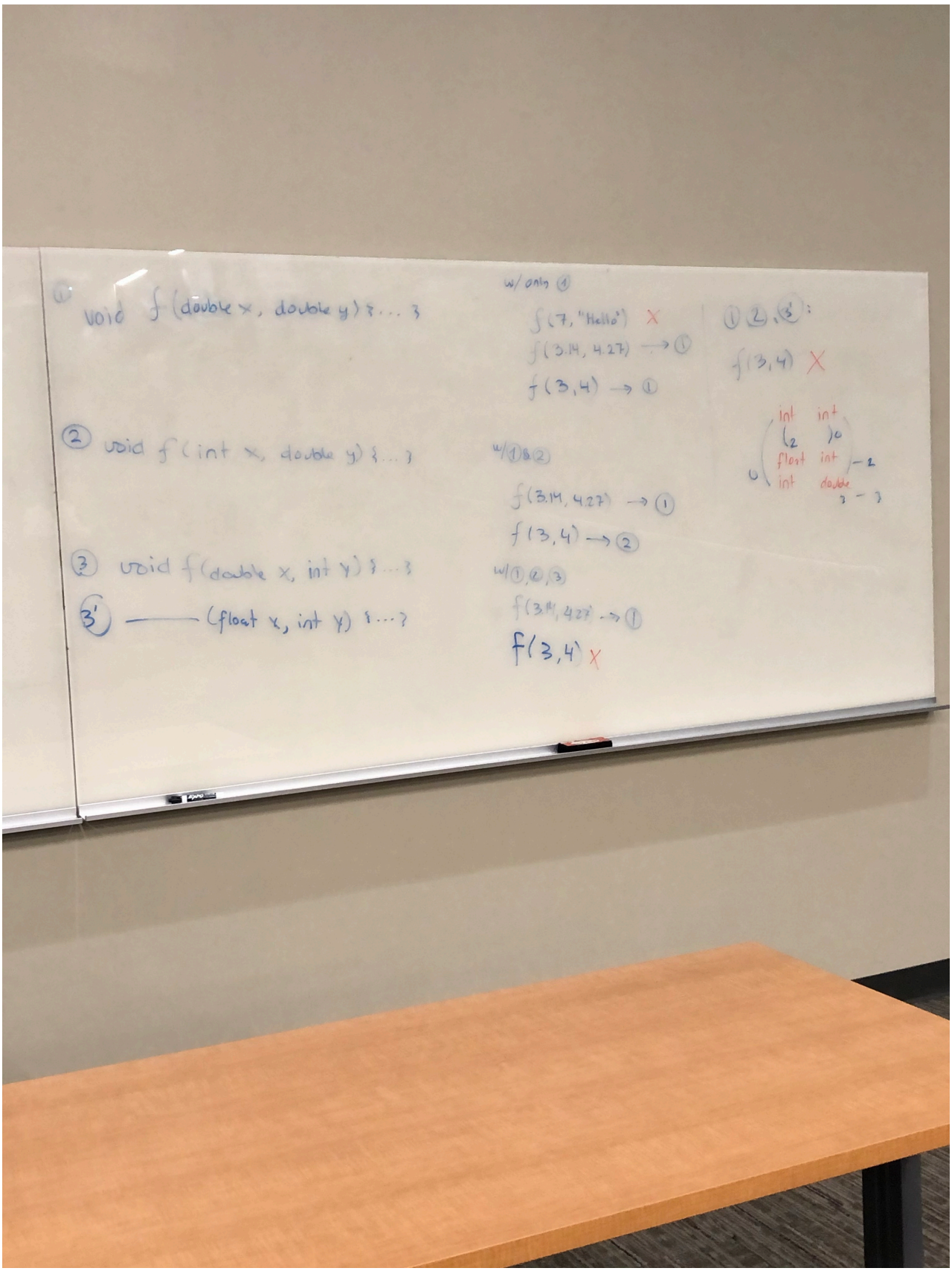

visit BinaryExpr(BinaryExpr be) {
visit left expr lt
visit right expr rt
switch (operator)
+ lt & rt numeric => [lt, rt]
- - - string => string
?, string
String.?
* /
< > <= >=
lt & rt numeric => Boolean
== !=
numeric or
classType => Boolean
>>> << >>
integral for lt
int atmost - rt
=> lt
^ %
return be.type;
}
1. get types of
2. determine if those types are
3. - overall type of expr




① void f(double x, double y) {... }
② void f(int x, double y) {... }
③ void f(double x, int y) {... }
3' ——— (float x, int y) {... }
w/ only ①
f(7, "Hello") X
f(3.14, 4.27) → ①
f(3,4) → ①
w/ ①&②
f(3.14, 4.27) → ①
f(3,4) → ②
w/ ①,②,③
f(3.14, 4.27) → ①
f(3,4) X
①,②,3':
f(3,4) X
int int
(2)0
float int
int double
-2
3 - 3