Homework 2: Type Checking

While I will not require your solution to be typed, your solution should be neat. If your solution is not neat, I might not grade it. If you are not sure if your solution is neat, you should ask. If your solution is typed, it is neat!

You should write your name on your submission.

You should show your work for all problems.

Problem 1. Consider the following type declarations

```
TYPE
  A1 : integer;
  A2 : pointer to float;
  A3 : pointer to integer;
  T1 : structure { x : integer; }
  T2 : structure { x : A1;      next : pointer to integer; }
  T3 : structure { a : integer;     b : float; }
  T4 : structure { b : float;       a : integer; }
  T5 : structure { a : pointer to T5; b : pointer to T6; c : pointer to T7; }
  T6 : structure { a : pointer to T6; b : pointer  to T5; c : pointer to T5; }
  T7 : structure { a : pointer to T6; b : pointer  to T7; c : pointer to T9; }
  T8 : structure { a : pointer to T7; b : pointer  to T6; c : pointer to T10; }

  T9 : array [4][5] of T8;       // array 4 rows 5 columns
 T10 : array [4][5] of T7;
```

Assuming the most permissive definition of structural equivalence, which types are structurally equivalent?

Consider the following variable declarations in conjunction to the above type declarations

```
VAR                          // var declaration section
 s : T9;
 t : T9;
 u : T10;
 v : array [5][4] of T8;
 w, z : struct {
          int a;
          struct T5* next;
        };
 x, y : struct {
          int a;
          struct T5* next;
        };
 f : function of T9 returns int;
 g : function of T9 returns A1;
 m : int;
 n : A1;
```

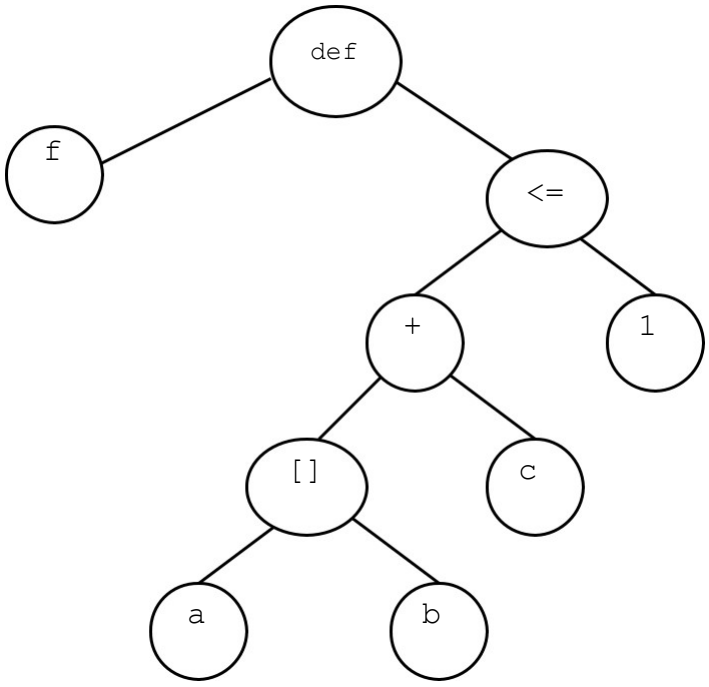
Assume that assignments between variables are allowed if the types of the variables are equivalent. For each of the following, list all type equivalence schemes under which the expression is valid. Consider name equivalence, internal name equivalence, and structural equivalence for each case. Assume that if two variables are equivalent under name equivalence, they are also equivalent under internal name equivalence.

- `s = t;`
- `t = u;`
- `u = v;`
- `v = w;`
- `w = z;`
- `z = x;`
- `m = f(s)`
- `n = f(u)`

Problem 2. Consider the following definition

```
fun f(a, b, c) = a[b] + c <= 1
```

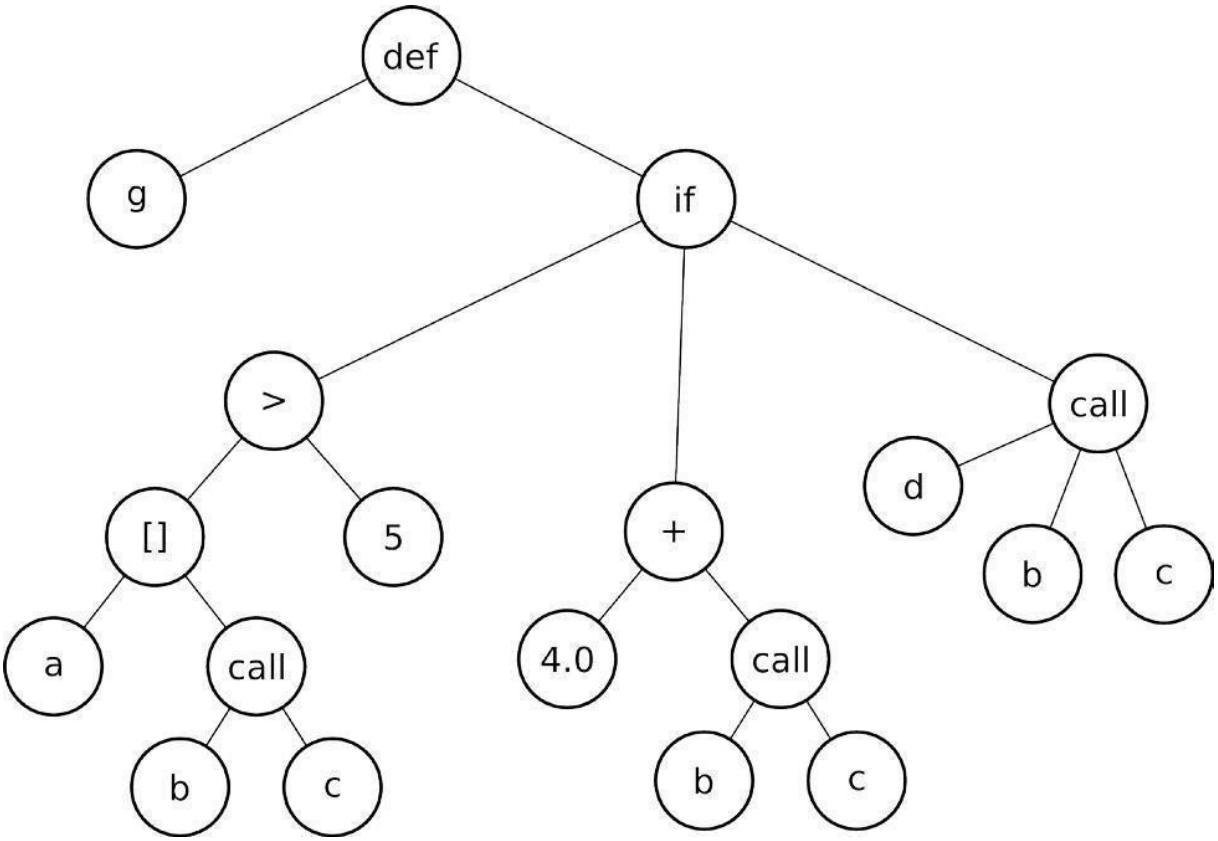
Using Hindley-Milner type inference, determine the type of f.



Problem 3. Consider the following definition:

```
fun g(a, b, c, d) = if a[b(c)] > 5 then
                        4.0 + b(c)
                    else
                        d(b, c)
```

Using Hindley-Milner type inference, determine the type of g.



Problem 4. Consider the following definition:

```
fun h(a, b, c) = if a[b * c](b, c) then
                     1.0
                 else
                     h(a[0], c, b)
```

Using Hindley-Milner type inference, determine the type of h.