Assignment 9.1 Type Checking

This exercise is about checking the types of expressions given in our C-like language. Use a deduction tree to check whether the statements are well-typed. Make sure to only use the rules given in the lecture and specify the rule for each step.

1. Given the declarations \( \Gamma := \{ \text{int } x, \text{int } a[\] \} \), check whether the statement \( \text{int } y = x + a[42] \); is well-typed.

2. Given the declarations \( \Gamma := \{ \text{int } y, \text{double } a[\], \text{struct } \{ \text{double } a[\]; \} g, \text{int } (\ast f)(\text{double}) \} \), check whether the statement \( \text{int } x = f(g.a[y+2]) \); is well-typed.

Assignment 9.2 Subtyping

Consider the following C structs:

```c
struct A {
    A f(B, C);
    C g(C);
};

struct B {
    B f(A, D);
    A g(D);
};

struct C {
    C f(B, B);
    D g(A);
};

struct D {
    D f(B, B);
    D g(B);
    int a;
};
```

We are going to use the non-standard subtyping rules for C structures which have been introduced in the lecture. Let \( \leq \) be the type comparison operator, that is, for two types \( A \) and \( B \) the following holds:

\[
A \leq B \iff A \text{ is a subtype of } B \tag{1}
\]

Now, proof the assertions below either right or wrong:

1. \( A \leq B \)
2. \( A \leq C \)