Problem 1.

For each of the following Cubex statements, give typing contexts Δ and Γ (assume $\Psi = \Psi_0$ and the other contexts are empty) in which it type checks with some return type and outgoing context, or else explain why it can't be done.

(a) `if (x + 2 == 4) return x;`

(b) `x := 0; while (f(x,y)) x := x + f(y,x);`

(c) `return [b, get([3, f], y)];`

Solution

Part (a)

In core cubex the statement is

`if(x.plus<>(2).equals<>(4)) return x; else {}`

The if statement requires that the expression in parenthesis has the `Boolean<>` type, and that both *returnx*; and {} have the same mutable variables available afterwards. The return statement does not make any new variables available so the second requirement holds. The function `equals`, provided in the Integer class returns a boolean as needed. Clearly, 4 is an integer so the type context in the argument of `equals` is correct. The expression `x.plus(2)` has the type Integer, since the function plus provided in the class Integer returns an Integer. This requires that in the type context of the statment `x` is an integer. Therefore, $\Delta = \emptyset$ and $\Gamma =$ `x:Integer<>`

Part (b)

From the assignment `x:=0; x` must have the type `Integer<>`. The while statement requires that `f(x,y)` has the type `Boolean<>`. Since `x` is an integer and the result of `plus(f(y,x)` is assigned to it `f(y,x)` must have the type `Integer<>`. Therefore `f` is overloaded, and to distinguish between these calls to different functions with the same name `y` must have a different type from `x`. Therfore,

$\Delta =$ `f<>(v1:Integer<>,v2:`τ`):Boolean<>,f<>(v1:`τ`,v2:Integer<>):Integer<>`

$\Gamma = x$: `Integer<>`, $y : \tau$

$\tau \neq$ `Integer<>`

Part (c)

The expression `[b, get([3, f], y)]` forms an `Iterable<`τ`>` of some type τ. Each element of the iterable must have the same type, including the return of the function `get`. Since the innermost Iterable contains 3, `f` must also be an integer. Therfore,

$\Delta =$ `get<>(`v_1`:Iterable<Integer<>>,`$v_2 : \tau_1$`):`τ

$\Gamma =$ `f:Integer<>,b:`τ`,y:`τ_1