Part 1: Theoretical Questions

1. Which of the following typing statement is true / false, explain why.

(a) {f : [T1 → T2], g : [T1 → T2], a : T1} ⊢ (f (g a)) : T2

false, f takes T1 not T2 like is the output of g.

(b) {f : [T1 × T2 → T3]} ⊢ (lambda (x) (f x 100)) : [T2 → T3]

false. The reason is that the lambda expression (lambda (x) (f x 100)) has type T2 → T3, but the environment {f : [T1 × T2 → T3]} only has the function f with type T1 × T2 → T3. In order for the lambda expression to have type T2 → T3, the environment would need to have a function f with type T2 → T3.

(c) {f : [T1 → T2]} ⊢ ((lambda (x) (f x))) : [T1 → T2]

true, if the lambda is called with T1 it will return T2 (we don't need to have any assumption about the type of x in the TEnv on the left side because x is not a free variable. The type of the bound variable can be inferred through type inference.)

(d) {f : [T1 × T2 → T3], y : T2} ⊢ (lambda (x) (f x y)) : [T1 → T3]

true, the right side lambda indeed accepts T1 and returns T3, if we assume the left hand side.

2. Perform type inference manually on the following expressions, using the Type Equations method. List all the steps of the procedure:

(a) ((lambda (f x1) (if x1 (f 1 x1) (f 3 x1))) + #t)

Step 1: Assign fresh type variables to all variables and constants:

f:T1
x1: T2
#t: Bool

Step 2: Set up type equations for the lambda expression:

f: T1, x1: T2 |- (if x1 (f 1 x1) (f 3 x1)): T3

Step 3: Set up type equations for the if-then-else expression:

x1: T2 |- x1: Bool
x1:T2, f:T1 |- (f 1 x1):T4
x1:T2, f:T1 |- (f 3 x1):T5
T3 = T4 = T5

Step 4: Set up type equations for the addition expression:

(+): Num -> Num -> Num
T3 = Num

Step 5: Write the type equation for the constant:

#t: Bool
Num = Bool
The type inference fails at because we cannot unify the types Num and Bool, Therefore, the expression is ill-typed.

(b) ((lambda (f1 x1 y1) (f1 x1 y1)) * 1 3)

Step 1: Assign fresh type variables to all variables and constants:

f1:T1
x1: T2
y1:T3
1: Number
3: Number

Step 2: Set up the type equation for the expression:

f1:T1, x1:T2, y1:T3 |- (f1 x1 y1):T4

Step 3: Set up type equations for the multiplication expression:

(*): Num -> Num -> Num
T4 = Num

Step 4: Write the type equation for the constant:

1: Number
3: Number

The type inference is successful, and the expression has a valid type of Num.