**Assignment 3 - Theoretical questions**

**Question 1**

1. This statement is True: {f : [T2 → T3], g : [T1 → T2], a : Number} ⊢ (f (g a)) : T3. Because we’ll get that a: Number so (g a) and we’ll get that so (g a) : and (f (g a)): T3 thus the statement is true.
2. This statement is False. The function f accepts only 1 argument. But in the statement is given 2 arguments.
3. This statement is True: {f : [T1 × T2 → T3], y : T2} ⊢ (lambda (x) (f x y)): [T1 → T3]

The nodes of the AST are:

x: (Introduce new TVar)

y: (By TEnv)

f: (By TEnv)

(f x y): (By AppExp typing rule and equation { })

(lambda (x) (f x y)): (By ProcExp typing rule)

Thus the statement is true.

1. This statement is True. In the case of T1 = T2 f is given an argument of the correct type and returns T1.

**Question 2**

1. The types are :
   1. never
   2. string
   3. any
   4. number
   5. never
   6. boolean
2. The replacements are:

[a] = (union number boolean)

[b] = (union number boolean)

[c] = (if (isBoolean z)

(not z)

(+ z 5))

1. The replacement of [answer] is (union string boolean number) because according to the given code snippet if x is a number (determined by (is\_number? x)), the function will check if x is greater than 0. If it is, it returns the string "positive". If not, it returns the string "negative". Therefore, in this branch, the return type is string.

If x is not a number, it must be a boolean (since x is of type (union number boolean)). The function checks if x is a boolean (determined by (is\_boolean? x)). If it is, it returns x itself, which is of type boolean. If this branch is executed, the return type is boolean.

The function also includes an else case where if x is neither a boolean nor a number (though logically this won’t happen due to the type of x), it returns the number 1. This path ensures that the function always returns a value, although it's redundant in this context.