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 $T_1 = \text{Number}$ so (g a) : T_2 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: T_x (Introduce new TVar)
   y: T_2 (By TEnv)
   f: $[T_1 * T_2 \rightarrow T_3]$ (By TEnv)
   (f x y): T_3 (By AppExp typing rule and equation { $T_x = T_1$ })
   (lambda (x) (f x y)): $[T_1 \rightarrow T_3]$ (By ProcExp typing rule)
   Thus the statement is true.
4. 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 :
   a. never
   b. string
   c. any
   d. number
   e. never
   f. boolean
2. The replacements are:
   [a] = (union number boolean)
   [b] = (union number boolean)
   [c] = (if (isBoolean z)
   (not z)
   (+ z 5))
3. 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.