Question 1: Theoretical Questions [12 points]

Are the following typing statements true or false? Explain why. **[4x3 points]**

a. $\{f : [T2 \to T3], g : [T1 \to T2], a : Number\} \vdash (f\ (g\ a)) : T3$

The statement is True.

Explaination:

- The list of node in the AST are:
  - $f : [T2 \to T3]$ (By TEnv)
  - $g : [T1 \to T2]$ (By TEnv)
  - $a : Tnum$ (introduce new Tvar)
- $(g\ a) : T2$ (By AppExp typing rule and equation $\{Tnum = T1\}$)
  - Since 'g' of type $[T1 \to T2]$ and $a : T1$ (since g expects T1 as the first argument), therefore $(g\ a)$ of type $T2$
- $(f\ (g\ a)) : T3$ (By AppExp typing rule)
  - Since $(g\ a)$ of type $T2$ and 'f' maps from T2 to T3, the result $(f\ (g\ a))$ is of type T3

Thus, the statement is true.

b. $\{f : [T1 \to [T2 \to Boolean]], x : T1, y : T2\} \vdash (f\ x\ y) : Boolean$

the statement is false.

Explaination:

- The list of node in the AST are:
  - $f : [T1 \to [T2 \to Boolean]]$ (by TEnv)
  - $x : T1$ (by TEnv)
  - $y : T2$ (by TEnv)
  - $(f\ x\ y)$: Type mismatch (By AppExp typing rule)

This statement is false because $(f\ x\ y)$ suggest applying 'f' to x and y simultaneously, which would require 'f' to have type $[T1 \times T2 \to Boolean]$, to match the argument structure. However, 'f' type is $[T1 \to [T2 \to Boolean]]$ indicates that 'f' first takes an argument of type 'T1' and returns a function $[T2 \to Boolean]$, not $[T1 \times T2 \to Boolean]$, therefore, the statement is false due to the mismatch in argument structure.

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

The statement is True.

Explanation:

- The list of node in the AST are:
  - f : [T1 × T2 → T3] (By TEnv)
  - y : T2 (By TEnv)
  - x : Tx (Introduce new Tvar)
- (f x y) : T3 (By AppExp typing rule and equation {Tx = T1}) (since 'f' expects a 'T1' as the first argument)
  - ↳ since 'f' is of type [T1 × T2 → T3], with 'x' of type T1 and 'y' of type 'T2' the expression (f x y) has type T3.
- lambda (x) (f x y) : [T1 → T3] (By procExp typing rule)
  - ↳ the lambda takes an argument 'x' of type 'T1' and returns a result of type 'T3', therefore, 'lambda (x) (f x y)' has type [T1 → T3]

Thus, The statement is true.

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

The statement is True.

Explanation:

- The list of node in the AST are:
  - f : [T2 → T1] (By TEnv)
  - x: T1 (by TEnv)
  - y : T3 (by TEnv)
- (f x): T1

To type (f x), x should ideally match the expected type 'T2' of f, By substituting {T2 = T1}, f type becomes [T1 → T1], which aligns with x's type, (f x) correctly returns type T1, Therefore, the statement is true.

2.1 We defined the type constructors `diff`, `inter`, and `union` according to set-theoretic principles. For each of the following TExps, write a simplified TExp denoting the same set of values. **[3 pts]**

a. `(inter number boolean)` → never
b. `(inter any string)` → string
c. `(union any never)` → any
d. `(diff (union number string) string)` → number
e. `(diff string (union number string))` → never
f. `(inter (union boolean number) (union boolean (diff string never)))` → boolean

2.2 Complete the following code snippet (i.e. offer replacements to the **[bracketed gaps]**) such that it will pass type checking in L52 but not in L51. **[8 pts]**

```
;; L52 return type is? boolean
;; L51 return type boolean
(define (isBoolean : (any -> is? boolean))
  (lambda ((x : any)) : is? boolean
    (boolean? x)))

;; Function to complete
(define (good_in_L52 : ((union number boolean) -> [a]))
    (lambda ((z : (union number boolean))) : [b]
        [c]
    ))
```

[a] → is? Boolean

[b] → is? Boolean

[c] → (isBoolean z)

2.3 Complete the return type for f in L52. Explain. **[7 pts]**

answer: (union string boolean)

Explaination:

- the function 'f' takes an argument 'x' that can be either a number or a boolean, and returns either a 'string' "positive" or "negative" if x is a number, or returns 'x' if it's a boolean.
- the return type of 'f' is (union string boolean), indicating it can return either a 'string' or a 'Boolean' based on the input type.