Question 1

a. {f : [T2 → T3], g : [T1 → T2], a : Number} ⊢ (f (g a)) : T3

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

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

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

- s' is applied to the type-expressions of s, i.e., for every variable T' for which s'(T') is defined, occurrences of T' in type expressions in s are replaced by s'(T').
- A variable T' in s', for which s(T) is defined, is removed from the domain of s', i.e., s'(T) is not defined on it anymore.
- The modified s' is added to s.
- Identity bindings, i.e., s(T) = T, are removed.
- If for some variable, (s o s')(T) includes T, the combination fails.

https://bguppl.github.io/interpreters/class_material/3.3TypeInferenceSystem.html

a)
acc = {}
a = number
T1 = a
T2 = T2
f = [T2 -> T3]
g = [T1 -> T2]

—
acc = {a = number}
{T1 = a} o acc => {T1 = number }

acc o {T1 = number} = > acc = {a = number, T1 = number}
T2 = T2
f = [T2 -> T3]
g = [T1 -> T2]

—-

{T2 = T2} o acc => {T2 = T2,...}
acc o {T2 = T2} => acc = {a = number, T1 = number, T2 = T2}

f = [T2 -> T3]
g = [T1 -> T2]

—-
{f = [T2 -> T3]} o acc => { f = [T2 ->T3] }
acc o {f = [T2 -> T3]} => acc = { f = [T2 ->T3] , a = number, T1 = number, T2 = T2 }
g = [T1 -> T2]

–

{g = [T1 -> T2] o acc => {g = [Number => T2]}
acc o {g = [T1 -> T2] => acc = { f = [T2 ->T3] , a = number, T1 = number, T2 = T2, g= [number=> T2] }

–

Finally to calculate the inference:
MGU: acc = { f = [T2 ->T3] , a = number, T1 = number, T2 = T2, g= [number=> T2] }

(f (g a))
(f ([number -> T2] number))
(f T2)
([T2 - > T3] T2)
T3

b) This inference is wrong.

inferred equations:
f: T3 -> boolean
x: T4
y: T5
T3: T4 * T5

This clashes with environment substitutions:

f : [T1 => [T2 => boolean]]
x : T1
y: T2

solving the equations we reach

T4 = T1
T5 = T2
T1 = T1 X T2

which is impossible.

C) step 1 substitute:

(lambda (x) (f x y)) ->_sub lambda (x) ([T1xT2 -> T3] x T2)

stage 2: type variables for every equation:

T0 = (lambda (x) (f x y))
T4 = (f x y)
T5 = x
T6 = y
T7 = f

stage 3: generate type equations

T0 = [T5 -> T4]
T7 = [T5xT6 => T4]
T6 = T2 (from substitution)
T7 = [T1 x T2 -> T3] (from substitution)

step 4: solve
acc = {}
T0 = [[T5 -> T4]
T7 = [T5xT6 => T4]
T6 = T2
T7 = [T1 x T2 -> T3]
—-
acc = { T0 = [[T5 -> T4]}
T7 = [T5xT6 => T4]
T6 = T2
T7 = [T1 x T2 -> T3]
—--
{T7 = [T5xT6 => T4]} o acc => {T7 = [T5xT6 => T4]}
acc o {T7 = [T5xT6 => T4]} => acc = { T0 = [[T5 -> T4] ,T7 = [T5xT6 => T4]}
T6 = T2
T7 = [T1 x T2 -> T3]
—-

{T6 = T2} o acc => {T6 = T2}
acc o {T6 = T2} => acc = { T0 = [[T5 -> T4] ,T7 = [T5xT2 => T4], T6=T2}
T7 = [T1 x T2 -> T3]
—-
{T7 = [T1 x T2 -> T3]} o acc => { [T1 x T2 -> T3] = [T5xT2 -> T4] }
acc o { [T1 x T2 -> T3] = [T5xT6 -> T4] } =>
acc = { T0 = [[T5 -> T4] ,T6=T2, T7 = [T1 x T2 -> T3] }
T1 = T5
T2 = T2
T3 = T4

—
{T1 = T5} o acc => {T1 = T5}
acc o {T1 = T5} =>
acc = { T0 = [[T5 -> T4] , T6=T2, T7 = [T5 x T2 -> T3], T1=T5 }
T2 = T6
T3 = T4
—
{T2 = T6} o acc => {T2 = T2} , not added.
acc = { T0 = [T5 -> T4] , T6=T2, T7 = [T5 x T2 -> T3], T1=T5 }
T3 = T4
–
{T3 = T4} o acc = { T3=T4}
acc o {T3 = T4} =>
acc = { T0 = [T5 -> T4] , T6=T2, T7 = [T5 x T2 -> T4], T1=T5, T3=T4}

MGU: acc = { T0 = [T5 -> T4] , T6=T2, T7 = [T5 x T2 -> T4], T1=T5, T3=T4}

step 5: using the MGU as substitution
T0 = (lambda (x) (f x y))
T4 = (f x y)
T5 = x
T6 = y
T7 = f

(lambda (T5) T4)

(lambda (T1) T3) =
[T1 -> T3]
TRUE

d) running through inference, we will find T2 = T1 so f is [T1=>T1] and (f x) returns T1.

Question 2

2.1

a) never
b) string
c) any
d) number
e) never
f) boolean

2.2

```
(define (isBoolean : (any -> is? boolean))
        (lambda ((x : any)) : is? boolean
                (boolean? x)))

(define (good_in_L52 : ((union number boolean) -> boolean)
        ( lambda ((z: ( union number boolean))) (if (isBoolean z)
         z #f
        ))
```

2.3)

(union string boolean)

The return type can be either a string if x is a number, or a boolean if x is a boolean. The function never reaches the “1” as x is either a number or a boolean, so a number is not returned.