PPL assignment 3

1

a. true, under the substitution {T1 = number}, we can infer that (g a): T2,
   thus (f (g a)) : T3 is true.
b. false, (f x y) means appExp where operator=f, operands= x,y, meaning f should take 2 arguments, but according to the environment f takes only 1 argument, thus it wont pass the type checking.
c. true, in the lambda statement f,y are free and x is bound, first we can write it as: f: [T1 x T2 -> T3] y: T2 , x: ?. we can infer that x should be T1, so under the substitution {x = T1}, the lambda expression takes x: T1 and returns (f x y), which type is T3.
d. true, under the substitution {T2 = T1} , we can infer that f : [T1 -> T1], and since x's type is T1, (f x) type is T1.

2.1

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

2.2

a. boolean
b. boolean
c. (if (isBoolean z) z #f)

2.3

(union string boolean)

possible outcomes:
if x is a number, we'll enter the then, and we can either return "positive" or "negative"
if x is a boolean, we'll enter the alt, we'll always enter the then statement of the third if, and we'll return x (which is boolean in the then body).
assuming the compiler knows in advance that we never enter the third if alt, we never return 1 (since if we entered the alt after we know x is not a number, thus must be boolean) , making the whole if statement type (union string boolean)
** if no such check was made (that we can't enter the third if's alt) , the type of the whole statement should be (union string (union boolean number)).