Q1:

1) **Which of the following typing statement is true / false, explain why?**

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

answer: false , because g get a T1 type and T1 not necessary to be type of number (t1 can be a Boolean or string or …) because that it not true (or it not a legal type function)

***note:(we can assume that 1 is true if and only if T1 was type of number)**

b) {x : T1, y : T2, f : [T2 → T1]} ⊢ (f x) : T1
Answer: false , because function f get a T2 and x is a type T1 because that we gets an error for that (f x):T1 is not true (if we put y instead of x the (f x) : T1 was true)

c) {f : [T1 → T2], x : T1} ⊢ ((lambda () (f x))) : T2
Answer: it is false ,the body of the lambda is true because x is a T1 type and f get a T1 and return a T2 for that (f x) is true but the lambda it is a function that get something and return something (in another words lambda look like this [T1->T2]) but in the Question lambda it is a T2 type (that the type of what it returned),but it has to be a any->T2 type for that Folse

d) {f : [T1 × T2 → T3], y : T2} ⊢ (lambda (x) (f x y)) : [T1 → T3]
Answer: true ,because if lambda is called with T1 it will return T3 (we don't need to have any assumption about the type of x in the TEnv on the left side because x is not a free variable type of the bound variable can be inferred through type inference)

2) **Perform type inference manually on the following expressions, using the Type Equations method. List all the steps of the procedure:**

- **((lambda (f x1) (f 1 x1)) + #t)**
  Stage I: Rename bound variables
  ((lambda (f x1)(f 1 x1))+ #t) turn to ((lambda (f x)(f 1 x)) + #t)

  Stage II: Assign type variables for every sub expression:

Expression	Variable
((lambda (f x)(f 1 x)) + #t)	T_0
(lambda(f x)(f 1 x))	T_1
(f 1 x)	T_2
f	T_f
1	T_{1-num}
x	T_x
+	T_+
#t	$T_{\#t}$

  Stage III: Construct type equations.
  The equations for the sub-expressions are:

Expression	Equation
((lambda (f x)(f 1 x)) + #t)	$T_1 \coloneqq [T_+ * T_{\#t} \rightarrow T_0]$
(lambda(f x)(f 1 x))	$T_1 \coloneqq [T_f * T_x \rightarrow T_2]$
(f 1 x)	$T_f \coloneqq [T_{1-num} * T_x \rightarrow T_2]$

The equations for the primitives are:

Expression	Equation
+	$T_+ \coloneqq [Number * Number \rightarrow Number]$
#t	$Boolean$
1	$Number$

Stage IV: Solve the equations:

Equation	Substitution
$\boldsymbol{T_1 \coloneqq [T_+ * T_{\#t} \rightarrow T_0]}$	{}
$\boldsymbol{T_1 \coloneqq [T_f * T_x \rightarrow T_2]}$	
$\boldsymbol{T_f \coloneqq [T_{1-num} * T_x \rightarrow T_2]}$	
$\boldsymbol{T_+ \coloneqq [Number * Number \rightarrow Number]}$	
$\boldsymbol{T_{\#t} = Boolean}$	
$\boldsymbol{T_{1-num} = Number}$	

Step 1:

$$T_1 \coloneqq [T_+ * T_{\#t} \rightarrow T_0] \circ \text{Substitution} = (T_1 \coloneqq [T_+ * T_{\#t} \rightarrow T_0])$$

Equation	Substitution
$\boldsymbol{T_1 \coloneqq [T_f * T_x \rightarrow T_2]}$	$T_1 \coloneqq [T_+ * T_{\#t} \rightarrow T_0]$
$\boldsymbol{T_f \coloneqq [T_{1-num} * T_x \rightarrow T_2]}$	
$\boldsymbol{T_+ \coloneqq [Number * Number \rightarrow Number]}$	
$\boldsymbol{T_{\#t} = Boolean}$	
$\boldsymbol{T_{1-num} = Number}$	

Step 2:

T1 = $[T_f * T_x \rightarrow T_2]$ ○ Substitution = ($[T_+ * T_{\#t} \rightarrow T_0]$ =$[T_f * T_x \rightarrow T_2]$) There is not type-sub since both sides of the equation are composite we split it into three equations and remove equation $T_1 \coloneqq [T_f * T_{1-num} * T_x \rightarrow T_1]$.

Equation	Substitution
$\boldsymbol{T_f \coloneqq [T_{1-num} * T_x \rightarrow T_2]}$	$T_1 \coloneqq [T_+ * T_{\#t} \rightarrow T_0]$
$\boldsymbol{T_+ \coloneqq [Number * Number \rightarrow Number]}$	
$\boldsymbol{T_{\#t} = Boolean}$	
$\boldsymbol{T_{1-num} = Number}$	
$\boldsymbol{T_f = T_+}$	
$\boldsymbol{T_x = T_{\#t}}$	
$\boldsymbol{T_2 = T_0}$	

Step 3:

(T_f = $[T_{1-num} * T_x \rightarrow T_2]$) ○ Substitution = ($T_f$ = $[T_{1-num} * T_x \rightarrow T_2]$). Substitution = Substitution ○ (T_f = $[T_{1-num} * T_x \rightarrow T_2]$).

Equation	Substitution
$\boldsymbol{T_+ \coloneqq [Number * Number \to Number]}$	$T_1 \coloneqq [T_+ * T_{\#t} \to T_0]$
$\boldsymbol{T_{\#t} = Boolean}$	$T_f \coloneqq [T_{1-num} * T_x \to T_2]$
$\boldsymbol{T_x = Number}$	
$\boldsymbol{T_f = T_+}$	
$\boldsymbol{T_{1-num} = T_{\#t}}$	
$\boldsymbol{T_2 = T_0}$	

Step 4:
(
$T_+ \coloneqq [Number * Number \to Number]$) ○ Substitution = (
$T_+ \coloneqq [Number * Number \to Number]$). Substitution = Substitution ○ (
$T_+ \coloneqq [Number * Number \to Number]$)

Equation	Substitution
$\boldsymbol{T_{\#t} = Boolean}$	$T_1 \coloneqq [[Number * Number \to Number] * T_{\#t} \to T_0]$
$\boldsymbol{T_{1-num} = Number}$	$T_f \coloneqq [T_{1-num} * T_x \to T_2]$
$\boldsymbol{T_f = T_+}$	$T_+ \coloneqq [Number * Number \to Number]$
$\boldsymbol{T_x = T_{\#t}}$	
$\boldsymbol{T_2 = T_0}$	

Step 5:
($T_{\#t}$ = $Boolean$) ○ Substitution = ($T_{\#t}$ = $Boolean$), is a type-sub. Substitution = Substitution ○ ($T_{\#t}$ = $Boolean$).

Equation	Substitution
$\boldsymbol{T_{1-num} = Number}$	$T_1 \coloneqq [[Number * Number \to Number] * Boolean \to T_0]$
$\boldsymbol{T_f = T_+}$	$T_f \coloneqq [T_{1-num} * T_x \to T_2]$
$\boldsymbol{T_x = T_{\#t}}$	$T_+ \coloneqq [Number * Number \to Number]$
$\boldsymbol{T_2 = T_0}$	$T_{\#t} = Boolean$

Step 6:
($T_{1-num} = Number$) ○ Substitution = ($T_{1-num} = Number$), is a type-sub. Substitution = Substitution ○ ($T_{1-num} = Number$).

Equation	Substitution
$T_f = T_+$	$T_1 \coloneqq [[Number * Number \to Number] * Boolean \to T_0]$
$T_x = T_{\#t}$	$T_f \coloneqq [Number * T_x \to T_2]$
$T_2 = T_0$	$T_+ \coloneqq [Number * Number \to Number]$
	$T_{\#t} = Boolean$
	$T_{1-num} = Number$

Step 7
($T_f = T_+$) ○ Substitution = ([$Number * T_x \to T_2$]=[$Number * Number \to Number$]) There is no a sub-type. We split the equation into three equations Tx = Number T2 = Number and

Number=Number(and there is from the same type for that we don't put it in the equation) , we remove the $T_f = T_+$

Equation	Substitution
$\boldsymbol{T_x = T_{\#t}}$	$T_1 \coloneqq [[Number * Number \to Number] * Boolean \to T_0]$
$\boldsymbol{T_2 = T_0}$	$T_f \coloneqq [Number * T_x \to T_2]$
$\boldsymbol{T_x = Number}$	$T_+ \coloneqq [Number * Number \to Number]$
$\boldsymbol{T_2 = Number}$	$T_{\#t} = Boolean$
	$T_{1-num} = Number$

Step 8:

$(T_x = T_{\#t})$ ○ Substitution=$T_x = Boolean$,is a is a type-sub. Substitution = Substitution ○ $(T_x = Boolean)$.

Equation	Substitution
$\boldsymbol{T_2 = T_0}$	$T_1 \coloneqq [[Number * Number \to Number] * Boolean \to T_0]$
$\boldsymbol{T_x = Number}$	$T_f \coloneqq [Number * T_x \to T_2]$
$\boldsymbol{T_2 = Number}$	$T_+ \coloneqq [Number * Number \to Number]$
$\boldsymbol{T_x = Boolean}$	$T_{\#t} = Boolean$
	$T_{1-num} = Number$

Step 9:

$T_2 = T_0$○ Substitution=$(T_2 = T_0)$, type-sub. Substitution = Substitution ○ $(T_2 = T_0)$.

Equation	Substitution
$\boldsymbol{T_x = Number}$	$T_1 \coloneqq [[Number * Number \to Number] * Boolean \to T_0]$
$\boldsymbol{T_2 = Number}$	$T_f \coloneqq [Number * T_x \to T_0]$
$\boldsymbol{T_x = Boolean}$	$T_+ \coloneqq [Number * Number \to Number]$
	$T_{\#t} \coloneqq Boolean$
	$T_{1-num} \coloneqq Number$
	$T_2 \coloneqq T_0$

Step 10:

(T_x = Number) ○ Substitution = (Number = Number) always true.

Equation	Substitution
$\boldsymbol{T_2 = Number}$	$T_1 \coloneqq [[Number * Number \to Number] * Boolean \to T_0]$
$\boldsymbol{T_x = Boolean}$	$T_f \coloneqq [Number * Number \to T_0]$
	$T_+ \coloneqq [Number * Number \to Number]$
	$T_{\#t} \coloneqq Boolean$
	$T_{1-num} \coloneqq Number$
	$T_2 \coloneqq T_0$
	$T_x = Number$

Step 11:

(T_2 = Number) ○ Substitution = (T_0 = Number), type-sub. Substitution = Substitution ○ (T_0 = Number).

Equation	Substitution
$\boldsymbol{T_x = Boolean}$	$T_1 \coloneqq [[Number * Number \to Number] * Boolean \to Number]$
	$T_f \coloneqq [Number * Number \to Number]$
	$T_+ \coloneqq [Number * Number \to Number]$
	$T_{\#t} \coloneqq Boolean$
	$T_{1-num} \coloneqq Number$
	$T_2 \coloneqq Number$
	$T_x \coloneqq Number$
	$T_0 \coloneqq Numebr$

Step 12 :
$T_x = Boolean$○ Substitution=(we know that T_x is Number)Number=Boolean
$we\ get\ an\ error\ becouse\ we\ say\ a\ number\ equals\ boolean$ and they are different types for that we can say the expression not well type(on another words this is wrong)

- **((lambda (f1 x1) (f1 x1 1)) + *)**

Stage I: Rename bound variables
((lambda (f1 x1)(f1 x1 1)) + *) turn to ((lambda (f x)(f x 1)) + *)

Stage II: Assign type variables for every sub expression:

Expression	Variable
((lambda (f x)(f x 1)) + *)	T_0
(lambda(f x)(f x 1))	T_1
(f x 1)	T_2
f	T_f
1	T_{1-num}
x	T_x
+	T_+
*	T_*

Stage III: Construct type equations.
The equations for the sub-expressions are:

Expression	Equation
((lambda (f x)(f x 1)) + *)	$T_1 \coloneqq [T_+ * T_* \to T_0]$
(lambda(f x)(f x 1))	$T_1 \coloneqq [T_f * T_x \to T_2]$
(f x 1)	$T_f \coloneqq [T_x * T_{1-num} \to T_2]$

The equations for the primitives are:

Expression	Equation
+	$T_+ \coloneqq [Number * Number \to Number]$
*	$T_* \coloneqq [Number * Number \to Number]$
1	$Number$

Stage IV: Solve the equations:

Equation	Substitution
$T_1 \coloneqq [T_+ * T_* \to T_0]$	{}
$T_1 \coloneqq [T_f * T_x \to T_2]$	
$T_f \coloneqq [T_x * T_{1-num} \to T_2]$	
$T_+ \coloneqq [Number * Number \to Number]$	
$T_* = [Number * Number \to Number]$	

$T_{1-num} = Number$	

Step 1:

$T_1 \coloneqq [T_+ * T_* \to T_0]$°Substitution $= (T_1 \coloneqq [T_+ * T_* \to T_0])$

Equation	Substitution
$T_1 \coloneqq [T_f * T_x \to T_2]$	$T_1 \coloneqq [T_+ * T_* \to T_0]$
$T_f \coloneqq [T_x * T_{1-num} \to T_2]$	
$T_+ \coloneqq [Number * Number \to Number]$	
$T_* = [Number * Number \to Number]$	
$T_{1-num} = Number$	

Step 2:

T1 = $[T_f * T_x \to T_2]$ ○ Substitution = ($[T_+ * T_* \to T_0]$ =$[T_f * T_x \to T_2]$) There is not type-sub since both sides of the equation are composite we split it into three equations and remove equation $T_1 \coloneqq [T_f * T_x \to T_1]$.

Equation	Substitution
$T_f \coloneqq [T_x * T_{1-num} \to T_2]$	$T_1 \coloneqq [T_+ * T_* \to T_0]$
$T_+ \coloneqq [Number * Number \to Number]$	
$T_* = [Number * Number \to Number]$	
$T_{1-num} = Number$	
$T_f = T_+$	
$T_x = T_*$	
$T_2 = T_0$	

Step 3:

(T_f= $[T_x * T_{1-num} \to T_2]$) ○ Substitution = ($T_f$ = $[T_x * T_{1-num} \to T_2]$). Substitution = Substitution ○ (T_f = $[T_x * T_{1-num} \to T_2]$).

Equation	Substitution
$T_+ \coloneqq [Number * Number \to Number]$	$T_1 \coloneqq [T_+ * T_* \to T_0]$
$T_* = [Number * Number \to Number]$	$T_f \coloneqq [T_x * T_{1-num} \to T_2]$
$T_{1-num} = Number$	
$T_f = T_+$	
$T_x = T_*$	
$T_2 = T_0$	

Step 4:

(

$T_+ \coloneqq [Number * Number \to Number]$) ○ Substitution = (

$T_+ \coloneqq [Number * Number \to Number]$). Substitution = Substitution ○ (

$T_+ \coloneqq [Number * Number \to Number]$)

Equation	Substitution
$T_* = [Number * Number \to Number]$	$T_1 \coloneqq [[Number * Number \to Number] * T_* \to T_0]$
$T_{1-num} = Number$	$T_f \coloneqq [T_x * T_{1-num} \to T_2]$
$T_f = T_+$	$T_+ \coloneqq [Number * Number \to Number]$
$T_x = T_*$	

$T_2 = T_0$	

Step 5:

$T_* \coloneqq [Number * Number \rightarrow Number]$) ○ Substitution = ($T_* \coloneqq [Number * Number \rightarrow Number]$). Substitution = Substitution ○ ($T_* \coloneqq [Number * Number \rightarrow Number]$)

Equation	**Substitution**
$T_{1-num} = Number$	$T_1 \coloneqq [[Number * Number \rightarrow Number] * [Number * Number \rightarrow Number]) \rightarrow T_0]$
$T_f = T_+$	$T_f \coloneqq [T_x * T_{1-num} \rightarrow T_2]$
$T_x = T_*$	$T_+ \coloneqq [Number * Number \rightarrow Number]$
$T_2 = T_0$	$T_* \coloneqq [Number * Number \rightarrow Number])$

Step 6:

($T_{1-num} = Number$) ○ Substitution = ($T_{1-num} = Number$), is a type-sub. Substitution = Substitution ○ ($T_{1-num} = Number$).

Equation	Substitution
$T_f = T_+$	$T_1 \coloneqq [[Number * Number \rightarrow Number] * [Number * Number \rightarrow Number] \rightarrow T_0]$
$T_x = T_*$	$T_f \coloneqq [T_x * Number \rightarrow T_2]$
$T_2 = T_0$	$T_+ \coloneqq [Number * Number \rightarrow Number]$
	$T_* \coloneqq [Number * Number \rightarrow Number])$
	$T_{1-num} = Number$

Step 7:

($T_f = T_+$) ○ Substitution = ([$Number * T_x \rightarrow T_2$]=[$Number * Number \rightarrow Number$]) There is no a sub-type. We split the equation into three equations Tx = Number T2 = Number and Number=Number (we pass it because it allows true), we remove the $T_f = T_+$

Equation	**Substitution**
$T_x = T_*$	$T_1 \coloneqq [[Number * Number \rightarrow Number] * [Number * Number \rightarrow Number] \rightarrow T_0]$
$T_2 = T_0$	$T_f \coloneqq [T_x * Number \rightarrow T_2]$
$T_x = Number$	$T_+ \coloneqq [Number * Number \rightarrow Number]$
$T_2 = Number$	$T_* \coloneqq [Number * Number \rightarrow Number])$
	$T_{1-num} = Number$

Step 8:

($T_x = T_*$) ○ Substitution=(T_x=[$Number * Number \rightarrow Number$]) ○ Substitution =Substitution○ (T_x=[$Number * Number \rightarrow Number$])

Equation	**Substitution**
$T_2 = T_0$	$T_1 \coloneqq [[Number * Number \rightarrow Number] * [Number * Number \rightarrow Number] \rightarrow T_0]$
$T_x = Number$	$T_f \coloneqq [\ Number * Number \rightarrow Number] * Number \rightarrow T_2]$
$T_2 = Number$	$T_+ \coloneqq [Number * Number \rightarrow Number]$
	$T_* \coloneqq [Number * Number \rightarrow Number])$
	$T_{1-num} \coloneqq Number$
	$T_x \coloneqq [\ Number * Number \rightarrow Number]$

Step 9:

$T_2 = T_0$○ Substitution=($T_2 = T_0$), type-sub. Substitution = Substitution ○ ($T_2 = T_0$).

Equation	Substitution
$T_x = Number$	$T_1 \coloneqq [[Number * Number \to Number] * [Number * Number \to Number] \to T_0]$
$T_2 = Number$	$T_f \coloneqq [\ Number * Number \to Number] * Number \to T_0]$
	$T_+ \coloneqq [Number * Number \to Number]$
	$T_* \coloneqq [Number * Number \to Number])$
	$T_{1-num} \coloneqq Number$
	$T_x \coloneqq [\ Number * Number \to Number]$
	$T_2 \coloneqq T_0$

Step 10:

$T_x = Number$○ Substitution= Number=[Number*Number->Number]

$we\ get\ an\ error\ becouse\ we\ say\ a\ number\ equals\ a\ function\ T_x$ and they are different types for that we can say the expression not well type(on another words this is wrong)

Q3:

<u>Write the Typing rule of:</u>

3.1:

1. **Typing rule set!:**

   For every : type environment _Tenv, variable reference _var1 , expressions _exp1 ,

   type expressions _t1:

   if _Tenv |- _exp1:_t1

   and _Tenv |- _var1: _t1

   then _Tenv |- (set! _var1 exp1): void

2. **Typing rule literal: I wrote the answer in to different ways one is specific and the other general**

   1) For every: type environment _Tenv,
      symbol expression _symb1
      compound sexp _sexp1
      number expression _num
      boolean expression _bool
      string expression _str
      clouser expression _cluser

   _Tenv |- _symb1 : Symbol(_symb1)
   _Tenv |- '() : Symbol
   _Tenv |- _sexp1: Pair
   _Tenv |- _num: Number
   _Tenv |- _bool: Boolean
   _Tenv |- _str: String
   _Tenv |-_cluser:Cluser

2) For every environment _Tenv,
Value _val1, and type expression _texp1
If (_Tenv |- _val1:_texp1) then _Tenv |- (quote _val1) : _texp1

3.2.2:

1) T**yping rule define-type:**

For every : type environment

Expressions $\_exp_1, \_r_1, \_\_r_2, , ,,,,,,, \_r_m,$

Type expressions string ,for all 1<=i<=m records

If _Tenv o {$\_exp_1$:string, $\_r_1$:records, $\_r_2$:record,……, $\_r_m$:record}|- $\_exp_1$: $userDefineType$ then _Tenv |-(define-type $\_exp_1$[$\_r_1$, $\_r_2$,,,,,,,, $\_r_m$]):void

2) **Typing rule Type-case:**

For every :type environment _Tenv,

Expressions $\_exp_1$,_ var_1,_ ce_1,……, $\_ce_n$,

Type expressions string ,CExp,

For all i>=1,i<=n caseExp

For each caseExp: expression _caseExp_value and

Expressions $\_valD_1$,_ $valD_2$,…..,_ $valD_j$,k

Type Expressions: _t1 ,_t2 …….,_tk,rt

If _Tenv o{$\_exp_1$:string , var_1:CExp,_ ce_1:caseExp,_ ce_2:caseExp,…..,_ ce_n:caseExp}|- exp_1:typeCase

Then _Tenv |-{Type-Case $\_exp_1$ $\_var_1$ {[,_ ce_1,……, $\_ce_n$]}:[$\_valD_1$,_ $valD_2$,…..,_ $valD_j$=>rt]