Part 1: Theoretical Questions

1.

(a) The statement is false. g gets T1 type and return T2 type so when g activates on the return value is T2 type, f gets T1 type as an argument but g return T2 type, so in our case f gets invalid argument type.

(b) The statement is true. f gets T2 type as an argument and return T1 type. f activates on y that is T2 type, so the return value is T1 type as expected.

(c) The statement is true. The statement receives and returns types like f. f receives ‘x’, that has no type until we will activate the procedure.

(d) The statement is false. We do not know for sure if T2 type is a number, so we do not know if we can activate f on this closure.

2.

(a) $\left((lambda\ (x)(+\ x\ 1))4\right)$

Expression	Var
$\left((lambda\ (x)(+\ x\ 1))4\right)$	T_0
$lambda\ (x)\ (+x\ 1)$	T_1
$(+x\ 1)$	T_2
1	T_{num1}
4	T_{num4}
+	T_+
x	T_x

Expression	Equation
$\left((lambda\ (x)(+\ x\ 1))4\right)$	$T_1 = [T_{num4} \rightarrow T_0]$
$lambda\ (x)\ (+x\ 1)$	$T_1 = [T_x \rightarrow T_2]$
$(+x\ 1)$	$T_2 = [T_x * T_{num1} \rightarrow T_2]$
+	$T_+ = [number * number \rightarrow number]$
1	$T_{num1} = number$
4	$T_{num4} = number$
x	$T_1 = [T_x \rightarrow T_2]$

Equation	Substitution
$T_1 = [T_{num4} \rightarrow T_0]$ $T_1 = [T_x \rightarrow T_2]$ $T_+ = [T_x * T_{num1} \rightarrow T_2]$ $T_+ = [number * number \rightarrow number]$ $T_{num1} = number$ $T_{num4} = number$	

→

Equation	Substitution
$T_1 = [T_x \rightarrow T_2]$ $T_+ = [T_x * T_{num1} \rightarrow T_2]$ $T_+ = [number * number \rightarrow number]$ $T_{num1} = number$ $T_{num4} = number$	$\boldsymbol{T_1 = [T_{num4} \rightarrow T_0]}$

Equation	Substitution
$T_+ = [T_x * T_{num1} \rightarrow T_2]$ $T_+ = [number * number \rightarrow number]$ $T_{num1} = number$ $T_{num4} = number$ $\boldsymbol{T_{num4} = T_x}$ $\boldsymbol{T_0 = T_2}$	$T_1 = [T_{num4} \rightarrow T_0]$

→

Equation	Substitution
$T_+ = [number * number \rightarrow number]$ $T_{num1} = number$ $T_{num4} = number$ $T_{num4} = T_x$ $T_0 = T_2$	$T_1 = [T_{num4} \rightarrow T_0]$ $\boldsymbol{T_+ = [T_x * T_{num1} \rightarrow T_2]}$

Equation	Substitution
$T_{num1} = number$ $T_{num4} = number$ $T_{num4} = T_x$ $T_0 = T_2$ $\boldsymbol{T_x = number}$ $\boldsymbol{T_2 = number}$	$T_1 = [T_{num4} \rightarrow T_0]$ $T_+ = [T_x * T_{num1} \rightarrow T_2]$

→

Equation	Substitution
$\boldsymbol{number} = T_x$ $T_0 = T_2$ $T_x = number$ $T_2 = number$	$T_1 = [\boldsymbol{number} \rightarrow T_0]$ $T_+ = [T_x * \boldsymbol{number} \rightarrow T_2]$ $\boldsymbol{T_{num1} = number}$ $\boldsymbol{T_{num4} = number}$

Equation	Substitution
$T_2 = number$	$T_1 = [number \rightarrow \boldsymbol{T_2}]$ $T_+ = [\boldsymbol{number} * \boldsymbol{number} \rightarrow T_2]$ $T_{num1} = number$ $T_{num4} = number$ $\boldsymbol{T_0 = T_2}$ $\boldsymbol{T_x = number}$

→

Equation	Substitution
	$T_1 = [number \rightarrow \boldsymbol{number}]$ $T_+ = [number * number \rightarrow \boldsymbol{number}]$ $T_{num1} = number$ $T_{num4} = number$ $T_0 = T_2$ $T_x = number$ $T_2 = number$

Eventually we get the following answer: $\big((lambda\ (x)(+\ x\ 1))4\big) : number \rightarrow number$

(b) $\big((lambda\ (f1\ x1)(f1\ x1\ 1))4\ +\big)$

Stage 1: rename bound variables to get the following equation:

$\big((lambda\ (f\ x)(f\ x\ 1))4\ +\big)$

Stage 2: variable type assignment:

Expression	Variable
$\big((lambda\ (f\ x)(f\ x\ 1))4\ +\big)$	$T0$
$(lambda\ (f\ x)(f\ x\ 1))$	$T1$
$(f\ x\ 1)$	$T2$
f	Tf
x	Tx
1	$Tnum1$
4	$Tnum4$
$+$	$Tsum$

Stage 3: equation construction:

Expression	Equation
$\big((lambda\ (f\ x)(f\ x\ 1))4\ +\big)$	$T1\ =\ Tnum4\ *\ Tsum\ \rightarrow\ T2$
$(lambda\ (f\ x)(f\ x\ 1))$	$T1\ =\ Tf\ *\ Tx\ \rightarrow\ T2$
$(f\ x\ 1)$	$Tf\ =\ Tx\ *\ Tnum1\ \rightarrow\ T2$
1	$Tnum1\ =\ number$
4	$Tnum4\ =\ number$
$+$	$Tsum = number * number \rightarrow number$

Stage 4: solving the equations:

Equation	Substitution
$T1 = Tf * Tx \rightarrow T2$	$\{\boldsymbol{T1} = [\boldsymbol{Tnum4} * \boldsymbol{Tsum} \rightarrow \boldsymbol{T2}]\}$
$Tf = Tx * Tnum1 \rightarrow T2$	
$Tnum1 = number$	
$Tnum4 = number$	
$Tsum = number * number \rightarrow number$	

Equation	Substitution
$\boldsymbol{Tnum4} = Tx * Tnum1 \rightarrow T2$	$\{T1 = [Tnum4 * Tsum \rightarrow T2]\}$
$Tnum1 = number$	
$Tnum4 = number$	
$Tsum = number * number \rightarrow number$	
$\boldsymbol{Tnum4} = \boldsymbol{Tf}$	
$\boldsymbol{Tx} = \boldsymbol{Tsum}$	

Equation	Substitution
$Tnum1 = number$	$\{T1 = [Tnum4 * Tsum \rightarrow T2]\}$ $\boldsymbol{Tnum4} = \boldsymbol{Tx} * \boldsymbol{Tnum1} \rightarrow \boldsymbol{T2}$
$Tnum4 = number$	
$Tsum = number * number \rightarrow number$	
$Tnum4 = Tf$	
$Tx = Tsum$	

Equation	Substitution
$Tsum = number * number \rightarrow number$	$\{T1 = [\boldsymbol{number} * Tsum \rightarrow T2]\}$ $\boldsymbol{number} = Tx * \boldsymbol{number} \rightarrow T2$
$\boldsymbol{number} = Tf$	
$Tx = Tsum$	

At this point we encounter an error because of the substitution: $number = Tx * number \rightarrow T2$ which is incorrect. So we return a failure and exit.

Part 2: Async Fun with TypeScript

2.2 (b) - The wrapped function returns Promise<R> because asyncMemo helps to activate f more efficiently by returning the value of the computation if it has been already done. The benefit of Promise<R> is that we can run the program asynchronically and not waiting for the calculation of each closure.

Part 3: Type Inference System

3.1

Typing rule define:
For every: type environment _Tenv,
variable _x1
expressions _e1 and
type expressions _S1, _U1:

If _Tenv |- _x1: _S1, and
_Tenv |- _e1: _U1, and
_Tenv |- _e1: _U1|- _S1 = _U1
Then _Tenv |- (define _x1 _e1): Void Texp

Typing rule set!:
For every: type environment _Tenv,
variable _x1
expressions _e1 and
type expressions _S1, _U1:
If _Tenv |- _x1: _S1, and
_Tenv |- _e1: _U1, and
_Tenv |- _e1: _U1|- _S1 = _U1
Then _Tenv |- (set! _x1 _e1): Void Texp