WE HAVE IMPLEMENTED VALUES AS SPECIAL FORM
NO SUPPORT FOR NESTED TUPLE

1.1 Stage 1: Rename bound variables

((lambda (x1 y1) (if (> x1 y1) #t #f)) 8 3) => **((lambda (x y) (if (> x y) #t #f)) 8 3)**

Stage 2: Assign type variables for every sub expression

Expression	Variable
((lambda (x y) (if (> x y) #t #f)) 8 3)	T_0
(lambda (x y) (if (> x y) #t #f))	T_1
(if (> x y) #t #f)	T_2
(> x y)	T_3
>	$T_>$
x	T_X
y	T_Y
#t	$T_{\#t}$
#f	$T_{\#f}$
8	T_{num8}
3	T_{num3}

Stage 3: Construct type equations. The equations for the sub-expressions are

Expression	Equation
((lambda (x y) (if (> x y) #t #f)) 8 3)	$T_1 = [T_{num8} * T_{num3} \rightarrow T_0]$
(lambda (x y) (if (> x y) #t #f))	$T_1 = [T_x * T_y \rightarrow T_2]$
(if (> x y) #t #f)	$T_2 = T_{\#t}\ and\ T_{\#t} = T_{\#f}$
(> x y)	$T_> = [T_x * T_y \rightarrow T_3]$
>	$T_> = [Number * Number \rightarrow Boolean]$
#t	$T_{\#t} = Boolean$
#f	$T_{\#f} = Boolean$
8	$T_{num8} = Number$
3	$T_{num3} = Number$

Stage 4: Solve the equations

Equation	Substitution
1. $T_1 = [T_{num8} * T_{num3} \rightarrow T_0]$	{}
2. $T_1 = [T_x * T_y \rightarrow T_2]$	
3. $T_2 = T_{\#t}$	
4. $T_{\#t} = T_{\#f}$	
5. $T_> = [T_x * T_y \rightarrow T_3]$	
6. $T_> = [Number * Number \rightarrow Boolean]$	
7. $T_{\#t} = Boolean$	
8. $T_{\#f} = Boolean$	
9. $T_{num8} = Number$	
10. $T_{num3} = Number$	

Step1:

$(T_1 = [T_{num8} * T_{num3} \rightarrow T_0] \circ \text{Substitution} = T_1 = [T_{num8} * T_{num3} \rightarrow T_0])$

$\text{Substitution} = \text{Substitution} \circ T_1 = [T_{num8} * T_{num3} \rightarrow T_0]$

Equation	Substitution
1. $T_1 = [T_x * T_y \rightarrow T_2]$	$\{T_1 := [T_{num8} * T_{num3} \rightarrow T_0]\}$
2. $T_2 = T_{\#t}$	
3. $T_{\#t} = T_{\#f}$	
4. $T_> = [T_x * T_y \rightarrow T_3]$	
5. $T_> = [Number * Number \rightarrow Boolean]$	
6. $T_{\#t} = Boolean$	
7. $T_{\#f} = Boolean$	
8. $T_{num8} = Number$	
9. $T_{num3} = Number$	

Step2:

$$(T_1 = [T_x * T_y \rightarrow T_2]) \circ \text{Substitution} =$$
$$([T_{num8} * T_{num3} \rightarrow T_0] = [T_x * T_y \rightarrow T_2])$$

Equation	Substitution
1. $T_2 = T_{\#t}$	$\{T_1 := [T_{num8} * T_{num3} \rightarrow T_0]\}$
2. $T_{\#t} = T_{\#f}$	
3. $T_> = [T_x * T_y \rightarrow T_3]$	
4. $T_> = [Number * Number \rightarrow Boolean]$	
5. $T_{\#t} = Boolean$	
6. $T_{\#f} = Boolean$	
7. $T_{num8} = Number$	

8. $T_{num3} = Number$	
9. $T_{num8} = T_x$	
10. $T_{num3} = T_y$	
11. $T_0 = T_2$	

Step3:

$$(T_2 = T_{\#t}) \circ \text{ Substitution} = (T_2 = T_{\#t})$$
$$\text{Substitution} = \text{Substitution} \circ (T_2 = T_{\#t})$$

Equation	**Substitution**
1. $T_{\#t} = T_{\#f}$	$\{T_1 := [T_{num8} * T_{num3} \rightarrow T_0]\}$
2. $T_> = [T_x * T_y \rightarrow T_3]$	$\{T_2 := T_{\#t}\}$
3. $T_> = [Number * Number \rightarrow Boolean]$	
4. $T_{\#t} = Boolean$	
5. $T_{\#f} = Boolean$	
6. $T_{num8} = Number$	
7. $T_{num3} = Number$	
8. $T_{num8} = T_x$	
9. $T_{num3} = T_y$	
10. $T_0 = T_2$	

Step4:

$$(T_{\#t} = T_{\#f}) \circ \text{ Substitution} = (T_{\#t} = T_{\#f})$$
$$\text{Substitution} = \text{Substitution} \circ (T_{\#t} = T_{\#f})$$

Equation	**Substitution**
1. $T_> = [T_x * T_y \rightarrow T_3]$	$\{T_1 := [T_{num8} * T_{num3} \rightarrow T_0]\}$
2. $T_> = [Number * Number \rightarrow Boolean]$	$\{T_2 := T_{\#t}\}$
3. $T_{\#t} = Boolean$	$\{T_{\#t} := T_{\#f}\}$
4. $T_{\#f} = Boolean$	
5. $T_{num8} = Number$	
6. $T_{num3} = Number$	
7. $T_{num8} = T_x$	
8. $T_{num3} = T_y$	
9. $T_0 = T_2$	

Step 5:

$$T_{>} = [T_x * T_y \rightarrow T_3] \circ \text{Substitution} = (T_{>} = [T_x * T_y \rightarrow T_3]),$$
$$\text{Substitution} = \text{Substitution} \circ (T_{>} = [T_x * T_y \rightarrow T_3])$$

Equation	**Substitution**
1. $T_{>} = [Number * Number \rightarrow Boolean]$	$\{T_1 := [T_{num8} * T_{num3} \rightarrow T_0]\}$
2. $T_{\#t} = Boolean$	$\{T_2 := T_{\#t}\}$
3. $T_{\#f} = Boolean$	$\{T_{\#t} := T_{\#f}\}$
4. $T_{num8} = Number$	$\{T_{>} = [T_x * T_y \rightarrow T_3]\}$
5. $T_{num3} = Number$	
6. $T_{num8} = T_x$	
7. $T_{num3} = T_y$	
8. $T_0 = T_2$	

Step 6:

$$(T_{>} = [Number * Number \rightarrow Boolean]) \circ \text{Substitution} = T_{>}$$
$$= [T_x * T_y \rightarrow T_3] = [Number * Number \rightarrow Boolean]$$

Equation	**Substitution**
1. $T_{\#t} = Boolean$	$\{T_1 := [T_{num8} * T_{num3} \rightarrow T_0]\}$
2. $T_{\#f} = Boolean$	$\{T_2 := T_{\#t}\}$
3. $T_{num8} = Number$	$\{T_{\#t} := T_{\#f}\}$
4. $T_{num3} = Number$	$\{T_{>} = [T_x * T_y \rightarrow T_3]\}$
5. $T_{num8} = T_x$	
6. $T_{num3} = T_y$	
7. $T_0 = T_2$	
8. $T_x = Number$	
9. $T_y = Number$	
10. $T_3 = Boolean$	

Step 7:

$$T_{\#t} = Boolean \circ \text{Substitution} = (T_{\#t} = Boolean)$$
$$\text{Substitution} = \text{Substitution} \circ (T_{\#t} = Boolean)$$

Equation	Substitution
1. $T_{\#f} = Boolean$	$\{T_1 := [T_{num8} * T_{num3} \rightarrow T_0]\}$
2. $T_{num8} = Number$	$\{T_2 := Boolean\}$
3. $T_{num3} = Number$	$\{T_{\#t} := T_{\#f}\}$
4. $T_{num8} = T_x$	$\{T_> = [T_x * T_y \rightarrow T_3]\}$
5. $T_{num3} = T_y$	$\{T_{\#t} = Boolean\}$
6. $T_0 = T_2$	$\{T_{\#f} = Boolean\}$
7. $T_x = Number$	
8. $T_y = Number$	
9. $T_3 = Boolean$	

Step 8:

$$\left(T_{\#f} = Boolean \circ \text{Substitution} = (\text{Boolean} = \text{Boolean})\right)$$

Equation	Substitution
1. $T_{num8} = Number$	$\{T_1 := [T_{num8} * T_{num3} \rightarrow T_0]\}$
2. $T_{num3} = Number$	$\{T_2 := Boolean\}$
3. $T_{num8} = T_x$	$\{T_{\#t} := T_{\#f}\}$
4. $T_{num3} = T_y$	$\{T_> = [T_x * T_y \rightarrow T_3]\}$
5. $T_0 = T_2$	$\{T_{\#f} = Boolean\}$
6. $T_x = Number$	$T_{\#t} = Boolean\}$
7. $T_y = Number$	
8. $T_3 = Boolean$	

Step 9:

$(T_{num8} = Number) \circ$ Substitution $= (\mathrm{T}_{\mathrm{num8}} = Number)$

Substitution = Substitution $\circ$ $(T_{num8} = Number)$

$(T_{num3} = Number) \circ$ Substitution $= (\mathrm{T}_{\mathrm{num3}} = Number)$

Substitution = Substitution $\circ$ $(T_{num3} = Number)$

Equation	Substitution
1. $T_{num8} = T_x$	$\{T_1 := [Number * Number \rightarrow T_0]\}$
2. $T_{num3} = T_y$	$\{T_2 := Boolean\}$
3. $T_0 = T_2$	$\{T_{\#t} := T_{\#f}\}$
4. $T_x = Number$	$\{T_> = [T_x * T_y \rightarrow T_3]\}$
5. $T_y = Number$	$\{T_{\#f} = Boolean\}$
6. $T_3 = Boolean$	$\{T_{\#t} = Boolean\}$
7.	$\{T_{num8} = Number\}$
8.	$\{T_{num3} = Number\}$

Step 10:

$(T_{num8} = T_x) \circ$ Substitution $= (T_x = Number)$

Substitution = Substitution $\circ$ $(T_x = Number)$

$(T_{num3} = T_y) \circ$ Substitution $= (T_y = Number)$

Substitution = Substitution $\circ$ $(T_y = Number)$

Equation	Substitution
1. $T_0 = T_2$	$\{T_1 := [T_{num8} * T_{num3} \rightarrow T_0]\}$
2. $T_x = Number$	$\{T_2 := Boolean\}$
3. $T_y = Number$	$\{T_{\#t} := T_{\#f}\}$
4. $T_3 = Boolean$	$\{T_> = [Number * Number \rightarrow T_3]\}$
5.	$\{T_{\#t} = Boolean\}$
6.	$\{T_{\#f} = Boolean\}$
7.	$\{T_{num8} = Number\}$
8.	$\{T_{num3} = Number\}$
9.	$\{T_x = Number\}$
10.	$\{T_y = Number\}$

Step 13:

$(T_0 = T_2) \circ$ Substitution $= (\mathbf{T_0} = \boldsymbol{Boolean})$

1.2 Are these typing statements true? Explain.

a. {f:[T1->T2], x: T1} |- (f x): T2
The statement is **true**.
⇒ f is function from T_1 to T_2 and x ∈ T_1 so f(x) ∈ T_2

b. {f:[T1->T2],g: [T2->T3]}, x: T2 }|- (f g x): T3
The statement if **false**.
⟹ f expected **one** argument of type T_1 but received **two** arguments.

c. {f:[T2->T1],g: [T1->T2], x: T1 }|- (f (g x)): T1
The statement is **true**.
⟹ g is function from $T_1 to\ T_2$, and $x \in T_1$.
In addition f is function from $T_2\ to\ T_1$ as necessary.

d. {f:[T2->Number], x: Number}|- (f x x): Number
The statement is **false**;
⟹ f expected **one** argument of type T_2 but received **two** arguments of type Number.

1.3 What is the type of the following primitive operators:

a. cons = $[T_1 * T_2 \rightarrow Pair(T_1, T_2)]$

b. car = $[Pair(T_1, T_2) \rightarrow T_1]$

c. cdr = $[Pair(T_1, T_2) \rightarrow T_2]$

1.4 Write the type of the following function: (Define f (lambda (x) (values x x x))) (see question 2 for the definition of values).

⟹ $[T_1 \rightarrow [T_1 * T_1 * T_1]$

1.5 Write the MGU of the following expressions, or state that there is no such MGU.

Substitution	**MGU**
$\{x: T_1\}, \{x: T_2\}$	$\{T_1 = T_2\}$
{x: Number}, {x: Number}	{}
$\{x: [T_1 * [T_1 \rightarrow T_2] \rightarrow Number]\}$ $\{x: [[T_3 \rightarrow Number] * [T_4 \rightarrow Number] \rightarrow Number]\}$	$\{T_1 = T_4 = [T_3 \rightarrow Number],$ $T2 = Number\}$
$\{x: [T_1 \rightarrow T_1]\}$ $\{x: [T_1 \rightarrow [Number * Number]\}$	$\{T1 = [Number * Number]\}$

2.3 Write the fully type-annotated version of this function:

a. (define f (lambda (x) (values x (+ x 1))))

```
(lambda ((x : number)) : (number * number) (values x (+ x 1))
```

b. (define g (lambda (x) (values “x” x)))

```
(lambda ((x : T)) : (string * T) (values “x” x))
```

4.2 What are the benefits of the promise interface compared to the callback interface

a. Dealing with errors is simpler with the use of "catch" mechanism instead of using different cases with callbacks.

b. The code is more readable because of the ability to concat promises instead of doing so in a nested way when working with callbacks.

Part 3 code:

```
function* braid(gen1: Generator, gen2: Generator) {

    let iter1 = gen1.next();
    let iter2 = gen2.next();

    while(iter1.value!=undefined || iter2.value!=undefined){
        if(iter1.value!=undefined){
            yield iter1.value;
            iter1 = gen1.next();
        }
        if(iter2.value!=undefined){
            yield iter2.value;
            iter2 = gen2.next();
        }
    }
}

function* biased(gen1: Generator, gen2: Generator) {
    let iter1a = gen1.next();
    let iter1b = gen1.next();
    let iter2 = gen2.next();

    while(iter1a.value!=undefined || iter2.value!=undefined){
        if(iter1a.value!=undefined){
            yield iter1a.value;
            iter1a = gen1.next();
        }
        if(iter1b.value!=undefined){
            yield iter1b.value;
            iter1b = gen1.next();
        }
        if(iter2.value!=undefined){
            yield iter2.value;
            iter2 = gen2.next();
        }
    }
}
```

Part 4 code:

```
export const f = (x: number): Promise<number> => {
    return new Promise (
        (resolve, reject) => {
        if (x!= 0) resolve(1/x);
        else reject("CAN NOT DIVIDE BY ZERO!");
        });
}

export const g = (x: number): Promise<number> => {
    return new Promise (
        (resolve, reject) => {
        if(true) resolve(x*x);
        else reject("NOT POSSIBLE!");
        });
}

export const h = (x: number): Promise<number> => {
    return g(x)
            .then((c) => {return f(c)})
            .catch();
}

export const slower = <T1, T2>(arr: [Promise<T1>, Promise<T2>]): Promise<strin
g> => {
    return new Promise(
      (resolve, reject) => {
        const f=(i:number)=>(value: T1 | T2)=>{
          done++;
          if(done===2){
            resolve(`(${i} , '${value}')`)
          }
        }
        let done = 0;
        arr[0].then(f(0)).catch(() => reject("0 FAILED!"))
        arr[1].then(f(1)).catch(() => reject("1 FAILED!"))
      });
}
```