חלק תיאורטי

- בחלק 2 מימשנו את "values" כ- primop.
- לא מימשנו nested tuples.

חלק 1

נבצע type inference על הביטוי: $\left(\left(lambda\ (x\ y)(if\ (>\ x\ y)\ \#t\ \#f)\right)8\ 3\right)$.

שלב 1: לא צריך לבצע renaming ל- bound variables.

שלב 2: Assign type variables for every sub expression and primitives:

Expression	Variable
$\left(\left(lambda\ (x\ y)(if\ (>\ x\ y)\ \#t\ \#f)\right)\ 8\ 3\right)$	T_0
$\left(lambda\ (x\ y)(if\ (>\ x\ y)\ \#t\ \#f)\right)$	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}

שלב 3: Construct type equations:

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

שלב 4: Solve the equations:

Equation	Substitution
$T_1 = [T_{num8} * T_{num3} \rightarrow T_0]$	{}
$T_1 = [T_x * T_y \rightarrow T_2]$	
$T_2 = T_{\#t}$ $T_{\#t} = T_{\#f}$	
$T_> = [T_x * T_y \rightarrow T_3]$	
$T_> = [Number * Number \rightarrow Boolean]$	
$T_{\#t} = Boolean$	
$T_{\#f} = Boolean$	
$T_{num8} = Number$	
$T_{num3} = Number$	
Equation	Substitution
$\cancel{T_1 = [T_x * T_y \rightarrow T_2]}$	$T_1 = [T_{num8} * T_{num3} \rightarrow T_0]$
$T_2 = T_{\#t}$ $T_{\#t} = T_{\#f}$	
$T_> = [T_x * T_y \rightarrow T_3]$	
$T_> = [Number * Number \rightarrow Boolean]$	
$T_{\#t} = Boolean$	
$T_{\#f} = Boolean$	
$T_{num8} = Number$	
$T_{num3} = Number$	
$T_x = T_{num8}$	
$T_y = T_{num3}$	
$T_2 = T_0$	
Equation	Substitution
$T_2 = T_{\#t}$ $T_{\#t} = T_{\#f}$	$T_1 = [T_{num8} * T_{num3} \rightarrow T_0]$
$T_> = [T_x * T_y \rightarrow T_3]$	
$T_> = [Number * Number \rightarrow Boolean]$	
$T_{\#t} = Boolean$	
$T_{\#f} = Boolean$	
$T_{num8} = Number$	
$T_{num3} = Number$	
$T_x = T_{num8}$	
$T_y = T_{num3}$	
$T_2 = T_0$	
Equation	Substitution

$T_> = [T_x * T_y \rightarrow T_3]$	$T_1 = [T_{num8} * T_{num3} \rightarrow T_0]$ $T_2 = T_{\#t}$ $T_2 = t_{\#f}$
$T_> = [Number * Number \rightarrow Boolean]$	
$T_{\#t} = Boolean$	
$T_{\#f} = Boolean$	
$T_{num8} = Number$	
$T_{num3} = Number$	
$T_x = T_{num8}$	
$T_y = T_{num3}$	
$T_2 = T_0$	
Equation	Substitution
$T_> = [Number * Number \rightarrow Boolean]$	$T_1 = [T_{num8} * T_{num3} \rightarrow T_0]$ $T_2 = T_{\#t}$ $T_2 = t_{\#f}$ $T_> = [T_x * T_y \rightarrow T_3]$
$T_{\#t} = Boolean$	
$T_{\#f} = Boolean$	
$T_{num8} = Number$	
$T_{num3} = Number$	
$T_x = T_{num8}$	
$T_y = T_{num3}$	
$T_2 = T_0$	
Equation	Substitution
$T_{\#t} = Boolean$	$T_1 = [T_{num8} * T_{num3} \rightarrow T_0]$ $T_2 = T_{\#t}$ $T_2 = t_{\#f}$ $T_> = [T_x * T_y \rightarrow T_3]$
$T_{\#f} = Boolean$	
$T_{num8} = Number$	
$T_{num3} = Number$	
$T_x = T_{num8}$	
$T_y = T_{num3}$	
$T_2 = T_0$	
$T_x = Number$	
$T_y = Number$	
$T_3 = Boolean$	
Equation	Substitution
$T_x = T_{num8}$	$T_1 = [Number * Number \rightarrow T_0]$ $T_{\#t} = Boolean$ $T_{\#f} = Boolean$ $T_> = [T_x * T_y \rightarrow T_3]$ $T_2 = Boolean$ $T_{num8} = Number$ $T_{num3} = Number$
$T_y = T_{num3}$	
$T_2 = T_0$	
$T_x = Number$	
$T_y = Number$	
$T_3 = Boolean$	
Equation	Substitution
$T_2 = T_0$	$T_1 = [Number * Number \rightarrow T_0]$ $T_{\#t} = Boolean$ $T_{\#f} = Boolean$ $T_> = [Number * Number \rightarrow T_3]$ $T_2 = Boolean$ $T_{num8} = Number$
$T_x = Number$	
$T_y = Number$	
$T_3 = Boolean$	

	$T_{num3} = Number$
	$T_x = Number$
	$T_y = Number$
Equation	Substitution
	$T_1 = [Number * Number \rightarrow Boolean]$
	$T_{\#t} = Boolean$
	$T_{\#f} = Boolean$
	$T_> = [Number * Number \rightarrow Boolean]$
	$T_2 = Boolean$
	$T_{num8} = Number$
	$T_{num3} = Number$
	$T_x = Number$
	$T_y = Number$
	$T_0 = Boolean$
	$T_3 = Boolean$

שאלה 2

a. כן. משום ש- x מטיפוס T_1 ו- f מקבלת T_1 כפרמטר ומחזירה T_2, ולכן הביטוי $(f\ x)$ מטיפוס T_2.

b. לא, כי f לא מקבלת שני פרמטרים, אלא היא מקבלת רק פרמטר אחד.

c. כן. כי g מקבלת T_1 ומחזירה T_2 ו-x מטיפוס T_1 ולכן g אכן תחזיר T_2. כמו כן, f מקבלת T_2 ומחזירה T_1 ואכן g מטיפוס T_2 ⇐ הטיפוס של הביטוי $\left(f\ (g\ x)\right)$ הוא T_1.

d. לא, כי f מקבלת פרמטר אחד מטיפוס T_2 ובביטוי $(f\ x\ x)$ היא מקבלת שני פרמטרים ולכן לא ניתן להסיק את הנדרש.

שאלה 3

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

b. $car: [Pair(T_1 * T_2) \rightarrow T_1]$

c. $cdr: [Pair(T_1 * T_2) \rightarrow T_2]$

שאלה 4

$$f: [T \rightarrow T * T * T]$$

שאלה 5

a. $MGU\ is\ \{T_1 = T_2\}$

b. $MGU\ is\ \{\}$

c. $MGU\ is\ \{T_1 = [T_3 \rightarrow Number], T_4 = [T_3 \rightarrow Number], T_2 = Number\}$

d. $MGU\ is\ \{T_1 = [Number \rightarrow Number]\}$

חלק 2

שאלה 3

```
(define f : [number -> number * number]
    (lambda (x : number) : number * number
        (values x (+ x 1))))

(define g : [T -> string * T]
    (lambda (x : T) : string * T
        (values "x" x)))
```

חלק 3

הקוד:

```
function* braid(generator1: Generator, generator2: Generator): Generator
{
    let first = generator1.next();
    let sec = generator2.next();

    while (!first.done && !sec.done) {
        yield first.value;
        yield sec.value;
        first = generator1.next();
        sec = generator2.next();
    }
    while (first.done && !sec.done) {
        yield sec.value;
        sec = generator2.next();
    }
    while (sec.done && !first.done) {
        yield first.value;
        first = generator1.next();
    }
```

```
}

function* biased(generator1: Generator, generator2: Generator): Generato
r {
    let first = generator1.next();
    let sec = generator2.next();

    while (!first.done && !sec.done) {
        yield first.value;
        if(!first.done){
            first = generator1.next();
            yield first.value
        }
        yield sec.value;
        first = generator1.next();
        sec = generator2.next();
    }
    while (first.done && !sec.done) {
        yield sec.value;
        sec = generator2.next();
    }
    while (sec.done && !first.done) {
        yield first.value;
        first = generator1.next();
    }
}
```

חלק 4

שאלה 1ב

- הרכבה של פונקציות אסינכרוניות בממשק ה promise יותר פשוט ואינטואיטיבי מאשר הממשק callback.
- הטיפוס של פונקציות אסינכרוניות בממשק promise דומה יותר לדרך שבה נגדיר טיפוס של פונקציות סינכרוניות, מאשר בממשק callback.

- טיפול בשגיאות בממשק promise ניתן לבצע במקום אחד. לעומת זאת, בממשק ה callback נצטרך להשתמש בהרבה if-else בשביל להתמודד עם שגיאות, מה שיוצר קוד מבלבל.

הקוד:

```
function f(x: number): Promise<number> {
    return new Promise((resolve, reject) => {
        try {
            resolve(1 / x);
        }
        catch (err) {
            console.log(err);
            reject(err);
        }
    })
}

function g(x: number): Promise<number> {
    return new Promise((resolve, reject) => {
        try {
            resolve(x * x);
        }
        catch (err) {
            console.log(err);
            reject(err);
        }
    })
}

function h(x: number): Promise<number> {
    return new Promise(async (resolve, reject) => {
        try {
            resolve(await f(await g(x)));
        }
        catch (err) {
            console.log(err);
```

```
            reject(err);
        }
    })
}

function slower<T1, T2>(ps: Promise<T1 | T2>[]): Promise<[number, T1 | T
2]> {
    return new Promise(async (resolve, reject) => {
        try {
            let arr: [number, T1 | T2][] = [];
            ps[0].then(async _ => resolve([1, await ps[1]]))
                .catch((e) => { throw e })
            ps[1].then(async _ => resolve([0, await ps[0]]))
                .catch((e) => { throw e })
        } catch (e) {
            reject(e);
        }
    })
}
```