Part 1: Theoretical questions

1. שלב א' – שינוי שמות של משתנים שהם bound: ((lambda (x y) (if (> x y) #t #f)) 8 3)  
   שלב ב' – השמת כל ביטוי למשנה טיפוס:

|  |  |
| --- | --- |
| Var | Expression |
|  | ((lambda (x y) (if (> x y) #t #f)) 8 3) |
|  | (lambda (x y) (if (> x y) #t #f)) |
|  | x |
|  | y |
|  | (if (> x y) #t #f) |
|  | (> x y) |
|  | > |
|  | #t |
|  | #f |
|  | 8 |
|  | 3 |

שלב ג' – בניית משוואות:

|  |  |
| --- | --- |
| Equations | Expression |
|  | ((lambda (x y) (if (> x y) #t #f)) 8 3) |
|  | (lambda (x y) (if (> x y) #t #f)) |
|  | (if (> x y) #t #f) |
|  | (> x y) |
|  | > |
|  | #t |
|  | #f |
|  | 8 |
|  | 3 |

שלב ד' – פתירת המשוואות:  
אתחול:

Substitution (empty)

|  |
| --- |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

צעד 1:  
משוואה:

*נפעיל את* substitution *על המשוואה:*אחד מצדי המשוואה המתקבלת הוא משתנה טיפוס, ולכן נפעיל את המשוואה על substitution:

Substitution

|  |
| --- |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

צעד 2:  
משוואה:

*נפעיל את* substitution *על המשוואה:*שני צידי המשוואה הם ביטויים טיפוסים מורכבים בעלי אותו בנאי, ולכן נפרק למשוואות של תתי הביטויים, נוסיף אותן לקבוצת המשוואות ונמחק את המשוואה המקורית של הצעד הזה.

Substitution

|  |
| --- |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

*צעד 3:  
משוואה:*

*נפעיל את* substitution *על המשוואה:*אחד מצדי המשוואה המתקבלת הוא משתנה טיפוס, ולכן נפעיל את המשוואה על substitution:

Substitution

|  |
| --- |
|  |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

*צעד 4:  
משוואה:*

*נפעיל את* substitution *על המשוואה:*אחד מצדי המשוואה המתקבלת הוא משתנה טיפוס, ולכן נפעיל את המשוואה על substitution:

Substitution

|  |
| --- |
|  |
|  |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

*צעד 5:  
משוואה:*

*נפעיל את* substitution *על המשוואה:*אחד מצדי המשוואה המתקבלת הוא משתנה טיפוס, ולכן נפעיל את המשוואה על substitution:

Substitution

|  |
| --- |
|  |
|  |
|  |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

*צעד 6:  
משוואה:*

*נפעיל את* substitution *על המשוואה:*שני צידי המשוואה הם ביטויים טיפוסים מורכבים בעלי אותו בנאי, ולכן נפרק למשוואות של תתי הביטויים, נוסיף אותן לקבוצת המשוואות ונמחק את המשוואה המקורית של הצעד הזה.

Substitution

|  |
| --- |
|  |
|  |
|  |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

*צעד 7:  
משוואה:*

*נפעיל את* substitution *על המשוואה:*אחד מצדי המשוואה המתקבלת הוא משתנה טיפוס, ולכן נפעיל את המשוואה על substitution:

Substitution

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

*צעד 8:  
משוואה:*

שני הצדדים הם טיפוסים אטומיים שווים, ולכן נמשיך לצעד הבא לאחר שנמחק את המשוואה הראשונה.

Substitution

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

צעד 9:  
משוואה:

*נפעיל את* substitution *על המשוואה:*אחד מצדי המשוואה המתקבלת הוא משתנה טיפוס, ולכן נפעיל את המשוואה על substitution:

Substitution

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

צעד 10:  
משוואה:

*נפעיל את* substitution *על המשוואה:*אחד מצדי המשוואה המתקבלת הוא משתנה טיפוס, ולכן נפעיל את המשוואה על substitution:

Substitution

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

צעד 11:  
משוואה:

*נפעיל את* substitution *על המשוואה:*אחד מצדי המשוואה המתקבלת הוא משתנה טיפוס, ולכן נפעיל את המשוואה על substitution:

Substitution

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

צעד 12:  
משוואה:

*נפעיל את* substitution *על המשוואה:*אחד מצדי המשוואה המתקבלת הוא משתנה טיפוס, ולכן נפעיל את המשוואה על substitution:

Substitution

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

צעד 13:  
משוואה:

*נפעיל את* substitution *על המשוואה:*אחד מצדי המשוואה המתקבלת הוא משתנה טיפוס, ולכן נפעיל את המשוואה על substitution:

Substitution

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

צעד 14:  
משוואה:

*נפעיל את* substitution *על המשוואה:*שני הצדדים הם טיפוסים אטומיים שווים, ולכן נמשיך לצעד הבא לאחר שנמחק את המשוואה הראשונה.

Substitution

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Equations

|  |  |
| --- | --- |
|  |  |
|  |  |

צעד 15:  
משוואה:

*נפעיל את* substitution *על המשוואה:*שני הצדדים הם טיפוסים אטומיים שווים, ולכן נמשיך לצעד הבא לאחר שנמחק את המשוואה הראשונה.

Substitution

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Equations

|  |  |
| --- | --- |
|  |  |

צעד 16:  
משוואה:

*נפעיל את* substitution *על המשוואה:*אחד מצדי המשוואה המתקבלת הוא משתנה טיפוס, ולכן נפעיל את המשוואה על substitution:

Substitution

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Equations (empty)

|  |  |
| --- | --- |
|  |  |

אלגוריתם הסקת הטיפוס הסתיים בהצלחה מאחר ולא נותרו משוואות לעבד (ולא נתקלו בשגיאה במשוואה האחרונה). לכן הסקנו שטיפוס הביטוי שבשאלה הוא Boolean.

* 1. הטענה נכונה כי מפעילים את f על x, מספר וסוג הפרמטרים מתאימים, ולכן טיפוס הביטוי הוא דרך ההחזרה של f שהוא T1
  2. הטענה אינה נכונה כי f מצפה לקבל פרמטר אחד בלבד, אולם היא מקבלת שניים.
  3. הטענה נכונה. מאותה סיבה בסעיף a, טיפוס ההחזרה של הביטוי (g x) הוא טיפוס ההחזרה של g שהוא T2. מאותה סיבה בדיוק, נקבל שטיפוס הביטוי כולו הוא טיפוס ההחזרה של f שהוא T1.
  4. הטענה שגויה מאותה סיבה בסעיף b
  5. cons: [T1\*T2->Pair(T1, T2)]  
     הסבר: cons מקבל שני ארגומנטים ויוצא מהם זוג סדור (כך שהארגומנט הראשון הוא הראשון בזוג והשני שני בזוג)
  6. car: [Pair(T1, T2)->T1]  
     הסבר: car מקבל זוג ומחזיר את הפריט הראשון שלו
  7. cdr: [Pair(T1, T2)->T2]  
     הסבר: cdr מקבל זוג ומחזיר את הפריט השני שלו

1. [T1->(T1 \* T1 \* T1)]
   1. {T1=T2}
   2. {}
   3. {T1=[T3->Number],T2=Number,T4=[T3->Number]}
   4. {T1=[Number->Number]}

Part 2: Type Checking

הערות לגבי המימוש:  
מיממשנו את ‘values’ בתור special form ואנו תומכים ב-tuple-ים מקוננים.

(define (f : (number -> (number \* number)))  
 (lambda ((x : number))  
 (values x (+ x 1)))  
  
(define (g : (T1 -> (string \* T1)))  
 (lambda ((x : T1))  
 (values “x” x))

Part 3: Generators

|  |
| --- |
| export function\* braid<T1, TReturn1, T2, TReturn2>(g1: Generator<T1, TReturn1>, g2: Generator<T2, TReturn2>): Generator<T1 | T2, void> {      let tailGen: Generator<T1, TReturn1> | Generator<T2, TReturn2>;      while (true) {          let r1: IteratorResult<T1, TReturn1> = g1.next();          if (r1.done) {              tailGen = g2;              break;          }          yield r1.value;          let r2: IteratorResult<T2, TReturn2> = g2.next();          if (r2.done) {              tailGen = g1;              break;          }          yield r2.value      }      let tailRes: IteratorResult<T1, TReturn1> | IteratorResult<T2, TReturn2>;      while (true) {          tailRes = tailGen.next();          if (tailRes.done) {              break;          }          yield tailRes.value;      }  }  export function\* biased<T1, TReturn1, T2, TReturn2>(g1: Generator<T1, TReturn1>, g2: Generator<T2, TReturn2>): Generator<T1 | T2, void> {      let tailGen: Generator<T1, TReturn1> | Generator<T2, TReturn2>;      while (true) {          let r1: IteratorResult<T1, TReturn1>;          r1 = g1.next();          if (r1.done) {              tailGen = g2;              break;          }          yield r1.value;          r1 = g1.next();          if (r1.done) {              tailGen = g2;              break;          }          yield r1.value;          let r2: IteratorResult<T2, TReturn2> = g2.next();          if (r2.done) {              tailGen = g1;              break;          }          yield r2.value;      }      let tailRes: IteratorResult<T1, TReturn1> | IteratorResult<T2, TReturn2>;      while (true) {          tailRes = tailGen.next();          if (tailRes.done) {              break;          }          yield tailRes.value;      }  } |

Part 4: Promises

1. b.

* יותר ברור מה טיפוס הערך שיחושב ע"י הפעולה.
* קל יותר בהרבה לשרשר/לבצע הרכבה של שתי פעולות (ב-Promises על ידי op(…).then((res) => op2(…)) ולעומת זאת עם callback-ים זה יהיה משהו כמו:  
  op1(…, (err, res) => {op2(…, (err, res) => {…}})
* קוד שמטפל בשגיאות של פעולות יהיו במקומות מרוכזים יותר כאשר השגיאות יפעפעו עד למעלה (כמו ב-bind)

קוד:

|  |
| --- |
| export function f(x: number): Promise<number> {      return new Promise<number>((resolve, reject) => {          try {              resolve(1 / x);          }          catch (e) {              reject(e);          }      });  }  export function g(x: number): Promise<number> {      return new Promise<number>((resolve, reject) => {          try {              resolve(x \* x);          }          catch (e) {              reject(e);          }      });  }  export function h(x: number): Promise<number> {      return g(x)          .then((x: number) => f(x))          .catch((e) => {              console.log(e);              return e;          });  }  export function slower(ps: [Promise<any>, Promise<any>]): Promise<string> {      let p1 = ps[0], p2 = ps[1];      return new Promise<string>((resolve, reject) => {          // NOTE: This implementation does not prevent race conditions.          // This doesn't seem to be the point of this question nor did we          // learn the tools typescript has to offer for protection against asynchronous code.          let whosDone: number = 0;          p1.then(x => {              if (!whosDone) {                  whosDone = 1;              }              // 1 -> 1, -1 -> 0              p2.then(y => resolve(whosDone == 1 ? `(1, ${y})` : `(0, ${x})`))                .catch(e => reject(e));          })          .catch(e => reject(e));          p2.then(y => {              if (!whosDone) {                  whosDone = -1;              }          });      });  } |