שאלה 1: Lazy Lists

1. תהיינה list1 ו-list2 שתי רשימות (לא עצלות) כלשהן.  
   נגדיר ש- list1 שקולה ל-list2 (וגם ההפך, זהו יחס שקילות) אם"ם מתקיים אחד מהדברים הבאים:
   1. list1=list2='()
   2. (car list1)=(car list2)  
      וגם (cdr list2) שקול ל-(cdr list1)

ונסמן list1=list2.  
  
נגדיר ש-lzl1 שקולה ל-lzl2 (וגם ההפך, זהו יחס שקילות) אם"ם לכל מתקיים  
ש-(take lzl1 n) שקול ל-(take lzl2 n)ונסמן lzl1=lzl2.

1. טענה: even-square-2=even-square-1.  
   כלומר לכל מתקיים כי  
   (take even-square-1 n)=(take even-square-2 n).

אבחנה: לכל רשימה עצלה אינסופית lzl ו- כך ש- מתקיים  
(take lzl n)=(cons (take lzl (- n 1)) (nth lzl n))

הוכחה:  
נשים לב שלכל מתקיים  
ש-(nth even-sqaure-1 n)=(nth even-square-2 n)=  
  
הוכחה באינדוקציה על .  
בסיס: : אז מתקיים

(take even-square-1 0) ==>\* '()  
(take even-square-2 0) ==>\* '()

הנחה: לכל הטענה נכונה.  
צעד: אז .  
מהאבחנה:

(take even-square-1 n)=  
(cons  
 (take even-square-1 (- n 1))  
 (nth even-square-1 n))

מהנחת האינדוקציה ומכך שערכי האבירים ה--ים ברשימות שווים נקבל ש:

(cons  
 (take even-square-1 (- n 1))  
 (nth even-square-1 n))=  
(cons  
 (take even-square-2 (- n 1))  
 (nth even-square-2 n))

מהאבחנה:

(cons  
 (take even-square-2 (- n 1))  
 (nth even-square-2 n))=  
(take even-square-2 n)

בסה"כ קיבלנו ש-(take even-square-1 n)=(take even-square-2 n).

שאלה 2: CPS

1. תהי f פרוצדורה מטיפוס [T1\*...\*Tn->T1|T2] כך ש-f מחזירה ערך מטיפוס T1 במקרה "הצלחה" וערך מסוג T2 במקרה של "כישלון".  
   נגדיר ש-f שקולה לגרסה ה-Success-Fail-Continuations שלה המסומנת f$ אם לכל קלט x1,...,xn ולכל פרוצדורה success-cont (מטיפוס [T1->Tsc] כך ש-Tsc יכול להיות כל טיפוס) מתקיים:  
   אם (f x1 ... xn) מניב "כישלון", אז קיימת פרוצדורה fail-cont כך שקיימים ערכים y1,...,ym כך ש-

(f$ x1 ... xn success-cont fail-cont)  
=(fail-cont y1 ... ym)=(f x1 ... xn)

כאשר הטיפוס של fail-cont הוא [T1\*...\*Tm->T2]  
או שקיימת פרוצדורה fail-cont מטיפוס [Empty->T2] כך ש-

(f$ x1 ... xn success-cont fail-cont)=(fail-cont)  
=(f x1 ... xn).

אחרת, במקרה של "הצלחה", לכל פרוצדורה fail-cont מתקיים:

(f$ x1 ... xn success-cont fail-cont)  
=(success-cont (f x1 ... xn)).

הערות:

* + המקרים "הצלחה" ו-"כישלון" של f אינם מוגדרים. על מי שמגדיר את הפונציה f, להגדיר מהם עבור f.
  + על fail-cont להיות פרוצדורה המחזירה T2. מספר הארגומנטים וטיפוסם אינו מוגדר. על מי שמגדיר את הפונקציה f, להגדיר את ממשק הפונקציה fail-cont.

1. במקרה שלנו, מקרה ההצלחה הוא כאשר קיים זוג ברשימה אשר הערך הראשון שלו שווה ל-key, ומקרה הכישלון הוא אחרת.

טיפוס הפרוצדורה get-values$ הוא  
[List<Pair<Symbol,T>>\*Symbol\*[T->T1]\*[Empty->T2] -> T1|T2].  
(מכאן שטיפוס הפונקציה fail-cont הוא [Empty->T2]).  
  
טענה: הפרוצדורה get-value$ שקולה לפרוצדורה get-value.  
כלומר לכל פרוצדורה success-cont מטיפוס [T->T1], לכל list מהטיפוס List<Pair<Symbol,T>> ולכל key מטיפוס Symbol:  
אם (get-value list key) מניב 'fail, אז קיימת פונקציה fail-cont (מהטיפוס [Empty->'fail] כך ש-

(get-value list key)  
=(fail-cont)=(get-value$ list key success-cont fail-cont).

אחרת (במקרה של הצלחה), אז לכל פרוצדורה fail-cont מטיפוס [Empty->T2] מתקיים:

(get-value$ list key success-cont fail-cont)  
=(success-cont (get-value list key)).

הוכחה: תהיו list רשימה מטיפוס List<Pair<Symbol,T>>, מפתח key מטיפוס Symbol ו-success-cont פרוצדורה מטיפוס [T->T1].  
הוכחה האינדוקציה על אורך הרשימה list: נסמן .  
בסיס:   
במקרה זה, הרשימה ריקה, כלומר list='().  
לכן, עבור הפרוצדורה fail-cont המוגדרת להיות (lambda () 'fail) נקבל:

(get-value$ list key success-cont fail-cont) ==>\*  
(fail-cont)='fail=(get-value list key) .

הנחה: עבור כל , הטענה נכונה.

צעד: אז , כלומר הרשימה אינה ריקה.  
*יהי* '(k . v) *הזוג הראשון ברשימה כלומר* '(k . v) = (car list)*.  
נחלק למקרים:*

מקרה א' – : אז (get-value list key)=v ולכל פרוצדורה  
fail-cont מטיפוס [Empty->T2]:

(get-values list key) ==>\* (cdr (car list))  
=(car '(k . v))=v

וגם:

(get-value$ list key success-cont fail-cont)  
==>\* (success-cont (cdr (car list)))  
= (success-cont (cdr '(k . v))) = (success-cont v)  
= (success-cont (get-value list key))

מקרה ב' – :

(get-value$ list key success-cont fail-cont)  
==>\* (get-value$ (cdr list) key success-cont fail-cont))

וגם:

(get-values list key) ==>\* (get-values (cdr list) key)

נשים לב שאורך הרשימה (cdr list) קטן ממש מ-. ולכן מהנחת האינדוקציה נקבל:

אם (get-values (cdr list) key)='fail, אז נקבל שקיימת פרוצדורה  
fail-cont מטיפוס [Empty->T2] כך ש:

(get-value$ list key success-cont fail-cont)  
==>\* (get-values$ (cdr list) key success-cont fail-cont)  
=(fail-cont)=(get-values (cdr list) key)=(get-values list key).

אחרת, (get-values (cdr list) key) הניב הצלחה.  
תהי fail-cont פרוצדורה מטיפוס [Empty->T2] כלשהי.  
אז נקבל (מהנחת האינדוקציה) ש:

(get-value$ list key success-cont fail-cont)  
==>\* (get-values$ (cdr list) key success-cont fail-cont)  
=(success-cont (get-values (cdr list) key))  
=(success-cont (get-values list key))

Part 3: Logic Programming

**סעיף 3.1**

1. Unify [ t ( s ( s ) , G , H , p, t ( E ) , s ) ,   
   t ( s ( H ) , G , p , p, t ( E ) , K ) ]

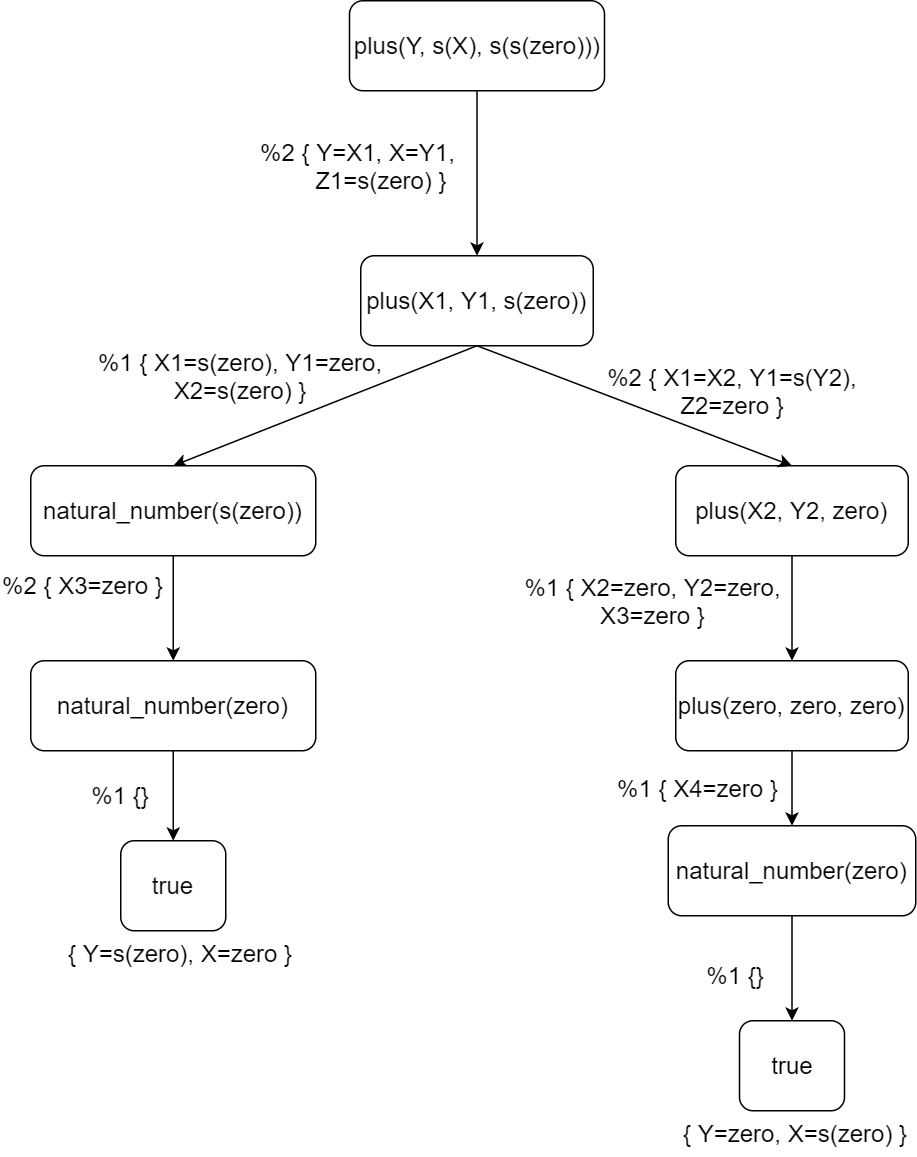
אתחול:  
substitution = {},  
equations = { t ( s ( s ) , G , H , p, t ( E ) , s ) = t ( s ( H ) , G , p , p, t ( E ) , K )}  
צעד 1:  
substitution={}  
equations={ s(s)=s(H), H=p, t(E)=t(E), K=s }  
צעד 2:  
substitution={ H=s }  
equations={ H=p, t(E)=t(E), K=s }  
צעד 3:  
המשוואה בצעד זה היא H=p. מכיוון שיש ב-substitution הצבה אחרת ל-H, הגענו לכישלון.

1. Unify [ g ( c , v ( U ) , g , G , U , E , v ( M ) ) ,   
   g ( c , M , g ,v ( M ) , v ( G ) , g , v ( M ) ]

אתחול:  
substitution={}  
equations={ g ( c , v ( U ) , g , G , U , E , v ( M ) ) =  
g ( c , M , g ,v ( M ) , v ( G ) , g , v ( M ) }  
צעד 1:  
substitution={}  
equations={ M=v(U) }  
equations={ M=v(U), G=v(v(U)) }  
equations={ M=v(v(G)),G=v(v(v(G))), U=v(G) }  
נתקלנו בשמה מעגלית, ולכן ה-unification נכשל.

1. Unify [ s ( [ v | [ [ v | V ] | A ] ] ) ,  
    s ( [ v | [ v | A ] ] ) ]  
   אתחול:  
   substitution={}  
   equations={ s ( [ v | [ [ v | V ] | A ] ] ) = s ( [ v | [ v | A ] ] )}  
   צעד 1:  
   substitution={}  
   equations={ [v|A] = [ [ v | V ] | A ] }  
   צעד 2:  
   substitution={}  
   equataions={ v=[v|A] }  
   לא יתכן כי אטום יהיה משהו מורכב, ולכן ה-unification נכשל.

**סעיף 3.3**

1. 
2. Y=s(zero), X=zero ;  
   Y=zero, X=s(zero) ;  
   false
3. זהו עץ הצלחה משום שקיים מסלול הצלחה בעץ (אפילו שניים).
4. זהו עץ סופי מכיוון שכל ענף הוא סופי, כולל הענפים שלא הראנו בשרטוט (כי ה-unification נכשל בניסיון ירידה ישיר אליהם).