עקרונות שפות תכנות : עבודה 4

1.1

a) **false.** הפונקציה g מקבלת את a שהוא מסוג NUMBER אבל היא מוגדרת לקבל משתנה מטיפוס אחר- 1T . לא הוגדר לנו ש T1=number
לכן שני הצדדים בביטוי זה שונים אז יחזיר false.

b) **false**. הפונקציה f מקבלת את X שמוגדר על ידי הטיפוס 1T
לעומת זאת, כמוגדרת לקבל פרמטר מטיפוס 2T .

c) **true.** הפונקציה f מקבלת את המשתנה X מטיפוס 1T בדיוק כמו שהיא מצפה.
כמו כן f מחזירה משתנה מטיפוס 2T כפי שהיא אמורה להחזיר.

d) **True.** הפונקציה f אמורה לקבל שני טיפוסים 1T ו2T. לא הוגדר הטיפוס של X אז ניתן להסיק שטיפוסו הוא 1T והטיפוס של Y מוגדר להיות 2T כנדרש כלומר TRUE

שאלה 2

(a)

שלב 1 : ((lambda (f x1) (f 1 x1)) + #t)=>((lambda(f x)(f1 1 x)) + #t))

שלב 2 :

Expression	Variable
((lambda(f x)(f1 1 x)) + #t)	**T0**
(lambda(f x)(f1 1 x))	**T1**
+	**T+**
#t	**Ttrue**
(f x)	**T2**
(f1 1 x)	**T3**
f	**Tf**
f1	**Tf1**
x	**Tx**
1	**Tnum1**

שלב 3:

Expression	Equation
((lambda(f x)(f1 1 x)) + #t)	**T1 = [T+*Ttrue->T0]**
(lambda(f x)(f1 1 x))	**T1 =[Tf*Tx->T3]**
(f x)	**Tf = [Tx -> T2]**
(f1 1 x)	**Tf1 = [Tnum1*Tx->T3]**

Expression	Equation
1	**Tnum1= Number**
+	**T+= [Number *Number ->Number]**
#t	**Ttrue = Boolean**

שלב 4: נפתור את מערכת המשוואות

Equation	**Subtitution**
T1 = [T+*Ttrue->T0]	**{}**
T1 =[Tf*Tx->T3]	
Tf = [Tx -> T2]	
Tf1 = [Tnum1*Tx->T3]	
Tnum1= Number	
T+= [Number *Number ->Number]	
Ttrue = Boolean	

Equation	**Subtitution**
T1 =[Tf*Tx->T3]	**{ T1 = [T+*Ttrue->T0]}**
Tf = [Tx -> T2]	
Tf1 = [Tnum1*Tx->T3]	
Tnum1= Number	
T+= [Number *Number ->Number]	
Ttrue = Boolean	

Equation	**Subtitution**
Tf = [Tx -> T2]	**{ T1 = [T+*Ttrue->T0]}**
Tf1 = [Tnum1*Tx->T3]	
Tnum1= Number	
T+= [Number *Number ->Number]	
Ttrue = Boolean	
T0 = T3	
Ttrue = Tx	
T+ = Tf	

Equation	**Subtitution**
Tf1 = [Tnum1*Tx->T3]	**{T1 = [T+*Ttrue->T0],** **Tf = [Tx -> T2] }**
Tnum1= Number	
T+= [Number *Number ->Number]	
Ttrue = Boolean	
T0 = T3	
Ttrue = Tx	
T+ = Tf	

Equation	**Subtitution**
Tnum1= Number	**{T1 = [T+*Ttrue->T0],** **Tf = [Tx -> T2],** **Tf1 = [Tnum1*Tx->T3]}**
T+= [Number *Number ->Number]	
Ttrue = Boolean	
T0 = T3	

T+= [Number *Number ->Number]
Ttrue = Tx
T+ = Tf

Equation	Subtitution
T+= [Number *Number ->Number]	**{T1 = [T+*Ttrue->T0], Tf = [Tx -> T2], Tf1 = [Number*Tx->T3], Tnum1= Number }**
Ttrue = Boolean	
T0 = T3	
Ttrue = Tx	
T+ = Tf	

Equation	Subtitution
Ttrue = Boolean	**{T1 = [[Number *Number ->Number]*Ttrue->T0], Tf = [Tx -> T2], Tf1 = [Number*Tx->T3], Tnum1= Number, T+= [Number *Number ->Number]}**
T0 = T3	
Ttrue = Tx	
Tf = [Number *Number ->Number]	

Equation	Subtitution
T0 = T3	**{T1 = [[Number *Number ->Number]* Boolean ->T0], Tf = [Tx -> T2], Tf1 = [Number*Tx->T3], Tnum1= Number, T+= [Number *Number ->Number], Ttrue = Boolean }**
Tx= Boolean	
Tf = [Number *Number ->Number]	

Equation	Subtitution
Tx = Boolean	**{T1 = [[Number *Number ->Number]* Boolean ->T0], Tf = [Tx -> T2], Tf1 = [Number*Tx->T3], Tnum1= Number, T+= [Number *Number ->Number], Ttrue = Boolean T0 = T3 }**
Tf = [Number *Number ->Number]	

Equation	**Subtitution**
Tf = [Number *Number ->Number],	**{T1 = [[Number *Number ->Number]* Boolean ->T0],** **Tf = [Boolean-> T2],** **Tf1 = [Number*Tx->T3],** **Tnum1= Number,** **T+= [Number *Number ->Number],** **Ttrue = Boolean,** **T0 = T3,** **Tx = Boolean ,** **}**

אז סתירה מפני שTf מוגדר לקבל Boolean ובנוסף מוגדר לקבל number *number כמובן שהדבר לא יתכן.

(b)

<u>שלב 1:</u>

((lambda (f1 x1) (f1 x1 1)) + *)=> ((lambda(f x)(f1 x 1)) + *)

שלב 2:

Expression	**Variable**
((lambda(f x)(f1 x 1)) + *)	**T0**
(lambda(f x)(f1 x 1))	**T1**
(f x)	**T2**
(f1 x 1)	**T3**
f1	**Tf1**
f	**Tf**
X	**Tx**
1	**Tnum1**
+	**T+**
*	**T***

<u>שלב 3:</u>

Expression	Equation
((lambda(f x)(f1 x 1)) + *)	**T1 =[T+*T*->T0]**
(lambda(f x)(f1 x 1))	**T1 = [Tf*Tx->T3]**
(f x)	**Tf= [Tx->.T2]**
(f1 x 1)	**Tf1 = [Tx*Tnum1->T3]**
1	**Tnum1 = [Number]**
+	**T+ =[Number *Number ->Number]**
*	**T* =[Number *Number ->Number]**

שלב 4:

Equation	**Substitution**
T1 =[T+*T*->T0]	**{}**
T1 = [Tf*Tx->T3]	
Tf= [Tx->T2]	
Tf1 = [Tx*Tnum1->T3]	
Tnum1 = [Number]	
T+ =[Number *Number ->Number]	
T* =[Number *Number ->Number]	

Equation	**Substitution**
T1 = [Tf*Tx->T3]	**{ T1 =[T+*T*->T0]}**
Tf= [Tx->T2]	
Tf1 = [Tx*Tnum1->T3]	
Tnum1 = [Number]	
T+ =[Number *Number ->Number]	
T* =[Number *Number ->Number]	

Equation	**Substitution**
Tf= [Tx->T2]	**{T1 =[T+*T*->T0]}**
Tf1 = [Tx*Tnum1->T3]	
Tnum1 = [Number]	
T+ =[Number *Number ->Number]	
T* =[Number *Number ->Number]	
T0=T3	
Tf=T+	
Tx= T*	

Equation	**Substitution**
Tf1 = [Tx*Tnum1->T3]	**{T1 =[T+*T*->T0], Tf= [Tx->T2] }**
Tnum1 = [Number]	
T+ =[Number *Number ->Number]	
T* =[Number *Number ->Number]	
T0=T3	
Tf=T+ = Tx->T2	
Tx= T*	

Equation	**Substitution**
Tnum1 = [Number]	**{T1 =[T+*T*->T0], Tf= [Tx->T2],** **Tf1 = [Tx*Tnum1->T3] }**

T+ =[Number *Number ->Number]
T* =[Number *Number ->Number]
T0=T3
Tf=T+ = Tx->T2
Tx= T*

Equation	**Substitution**
T+ =[Number *Number ->Number]	**{T1 =[T+*T*->T0], Tf= [Tx->T2], Tf1 = [Tx*Number->T3] , Tnum1 = [Number]}**
T* =[Number *Number ->Number]	
T0=T3	
Tf=T+ = Tx->T2	
Tx= T*	

Equation	**Substitution**
T* =[Number *Number ->Number]	**{T1 =[[Number *Number ->Number]*T*->T0], Tf= [Tx->T2], Tf1 = [Tx*Number->T3] , Tnum1 = [Number], T+ =[Number *Number ->Number] }**
T0=T3	
Tf=T+ = Tx->T2 = Number *Number ->Number	
Tx= T*	

Equation	**Substitution**
T0=T3	**{T1 =[[Number *Number ->Number]* [Number *Number ->Number] ->T0], Tf= [Tx->T2], Tf1 = [Tx*Number->T3] , Tnum1 = [Number], T+ =[Number *Number ->Number], T* =[Number *Number ->Number] }**
Tf=T+ = Tx->T2 = Number *Number ->Number	
Tx= T* = [Number *Number ->Number]	

Equation	**Substitution**
Tf=T+ = Tx->T2 = Number *Number ->Number	**{T1 =[[Number *Number ->Number]* [Number *Number ->Number] ->T0], Tf= [Tx->T2], Tf1 = [Tx*Number->T3] , Tnum1 = [Number], T+ =[Number *Number ->Number], T* =[Number *Number ->Number] , T0 =T3 }**

Tx= T* = [Number *Number ->Number]

Equation	**Substitution**
Tf=T+ = Tx->T2 = Number *Number ->Number	**{T1 =[[Number *Number ->Number]* [Number *Number ->Number] ->T0], Tf= [Tx->T2], Tf1 = [Tx*Number->T3] , Tnum1 = [Number], T+ =[Number *Number ->Number], T* =[Number *Number ->Number] , T0 =T3 }**
Tx= T* = [Number *Number ->Number]	

יש שגיאה בהגדרת הטיפוס Tx מפני שהגדרתו בשתי השורות בטבלה שונות.

3.1.1 Typing rules :

Set-Exp :

For every: type environment _Tenv,

Variable reference _r,

Expression _e

Type expressions _S :

If _Tenv |- e : _S and

_Tenv |- _r : _S

Then _Tenv |- (set! _r _e) : void

Lit-Exp :

For every: type environment _Tenv

Expressions e

If _Tenv |- e : T

Then ('e) : T

3.2.2 Typing rules :

Define-type :

For every: type environment _Tenv

Define_type _dt

_Tenv |- (_dt) : void

Type-case :

Expressions : rec1, … , recn (record -> for each record in the UDT)

Expressions : f11 , … , f1t , … , fnn (record.field -> for each record, f_{ij} is record i in the UDT at field j)

Expressions: _t, u (TypeCase, UDT)

Expressions: c1 , … , cn (cases.case -> for each case)

Expressions: c11,…,c1t,…,cnn (case.vardecl -> for each case, cij is case i in cases at vardecl j)

Type expressions _S11 , … , Snn , Sa1, … San

If _Tenv $_o$ { f11 : S11, … , fnn : Snn, … , c1.body : Sa1, … , cn.body : San }

_Tenv |- c11 : S11 , … , cnn : Snn

Then _Tenv |- (TypeCase u _t (c1(c11,…,c1j) c1.body) … (cn(cn1,…,cnn) cn.body)) :

checkCoverType(Sa1 , … , San).value