שפות תכנות – תרגיל 3

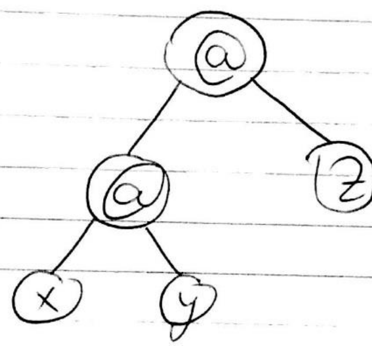
שאלה 1

1.

a. $x\ y\ z$

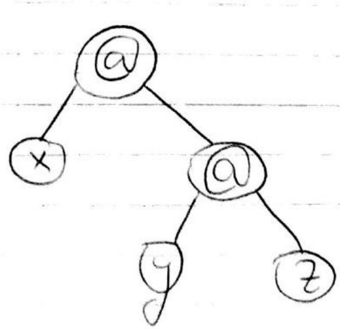
הביטוי ~~תקין~~ חוקי ומפרש לפי אסוציאטיביות שמאלית לכן

$$x\ y\ z \equiv (x\ y)\ z$$



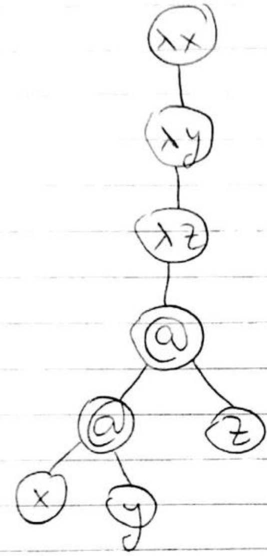
b. $x\ (y z)$

הביטוי חוקי.



c. $\lambda x \lambda y \lambda z. xyz = \lambda x.(\lambda y.(\lambda z. xyz))$

הביטוי תקין.

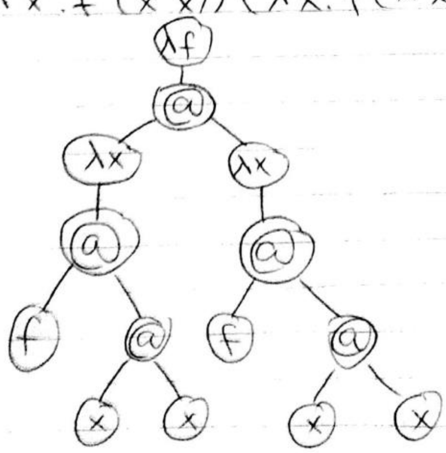


d. $\lambda x. y \lambda z.$

ביטוי לא חוקי לפי חוקי הדקדוק שלמדנו בכיתה,
שכן אין חוק שיוצר $.\lambda z$

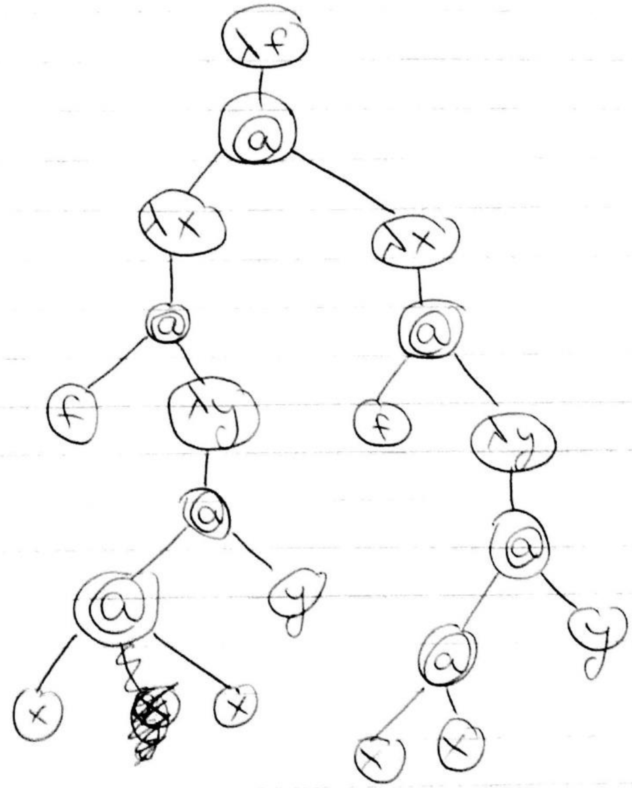
e. $\lambda f.(\lambda x.f(x x))(\lambda x.f(x x))$

ביטוי חוקי.



f. $\lambda f.(\lambda x.f(\lambda y.\,xxy))(\lambda x.\,f(\lambda y.xxy))$

ביטוי חוקי.



שאלה 2

$(\lambda x.\ \lambda x.\ (\lambda x. x)\ x)\ ((\lambda x.\ x\ x)\ \lambda x. x)$

a. ביטוי שמתקבל מהביטוי הנ"ל ע"י alpha-renaming, כך שבביטוי החדש אין משתנה שמופיע בשתי אבסטרקציות שונות:

$(\lambda x.\ \lambda x.\ (\lambda x. x)\ x)\ ((\lambda x.\ x\ x)\ \lambda x. x)$

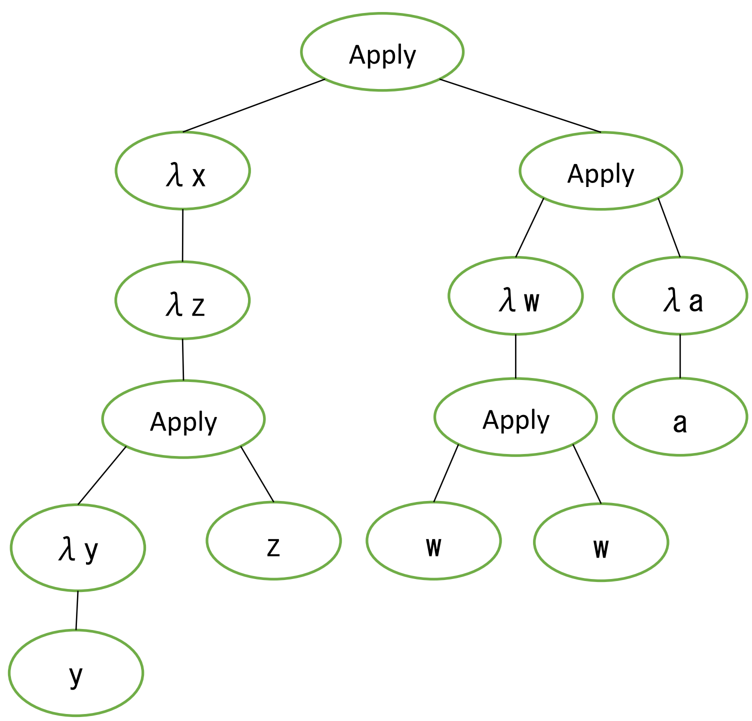
$=>_\alpha (\lambda x.\ \lambda x.\ (\lambda y. y)\ x)\ ((\lambda x.\ x\ x)\ \lambda x. x)$

$=>_\alpha (\lambda x.\ \lambda z.\ (\lambda y. y)\ z)\ ((\lambda x.\ x\ x)\ \lambda x. x)$

$=>_\alpha (\lambda x.\ \lambda z.\ (\lambda y. y)\ z)\ ((\lambda w.\ w\ w)\ \lambda x. x)$

$=>_\alpha (\lambda x.\ \lambda z.\ (\lambda y. y)\ z)\ ((\lambda w.\ w\ w)\ \lambda a. a)$

b. AST המתאים לביטוי מסעיף a:



c. סדרת חישוב על פי call by value:

$(\lambda x.\ \lambda z.\ (\lambda y. y)\ z)\ ((\lambda w.\ w\ w)\ \lambda a. a)$ $=>^{E\text{-}APPL2}$

$(\lambda x.\ \lambda z.\ (\lambda y. y)\ z)\ ((\lambda a. a)(\lambda a. a))$ $=>^{E\text{-}APLL2}$

$(\lambda x.\ \lambda z.\ (\lambda y. y)\ z)\ (\lambda a. a)$ $=>^{E\text{-}AppAbs}$

$\lambda z.\ (\lambda y. y)\ z$ $\not\Rightarrow$

d. סדרת חישוב על פי lazy evaluation:

$(\lambda x.\ \lambda z.\ (\lambda y.y)\ z)\ ((\lambda w.\ w\ w)\ \lambda a.a)$ $\Rightarrow^{\text{E-AppAbs}}$

$\lambda z.\ (\lambda y.y)\ z$ $\not\Rightarrow$

e. סדרת חישוב על פי normal order:

$(\lambda x.\ \lambda z.\ (\lambda y.y)\ z)\ ((\lambda w.\ w\ w)\ \lambda a.a)$ $\Rightarrow^{\text{E-AppAbs}}$

$\lambda z.\ (\lambda y.y)\ z$ $\Rightarrow^{\text{E-Abs}}$

$\lambda z.z$ $\not\Rightarrow$

שאלה 3

A

נראה סדרת חישוב לפי call by value (כאשר a,b הם value)

test (and tru fls) a b =

(λl. λm. λn. l m n) ((λb. λc. b c (λt. λf. f)) (λt. λf. t) (λt. λf. f)) a b =>

(λl. λm. λn. l m n) ((λc. (λt. λf. t) c (λt. λf. f)) (λt. λf. f)) a b =>

(λl. λm. λn. l m n) ((λt. λf. t) (λt. λf. f)) (λt. λf. f)) a b =>

(λl. λm. λn. l m n) ((λf. (λt. λf. f)) (λt. λf. f)) a b =>

(λl. λm. λn. l m n) (λt. λf. f) a b =>

(λm. λn. (λt. λf. f) m n) a b =>

(λn. (λt. λf. f) a n) b =>

(λt. λf. f) a b =>

(λf. f) b =>

b

B

OR

λb. λc. b tru c

<u>אם b הוא tru נקבל:</u>

λc. (λt. λf. t) tru c => (λt. λf. t) tru c => (λf. tru) c => tru

לכן נחזיר tru, לא משנה מה c. (כך גם OR עובד)

<u>אם b הוא fls נקבל:</u>

λc (λt. λf. f) tru c => (λt. λf. f) tru c => (λf. f) c => c

נחזיר את c.
זה נכון, כי כעת אם c הוא tru נקבל tru, ואם c הוא als נקבל fls – בדיוק כמו שOR היה מחזיר.

NOT

λb. b fls tru

<u>אם b הוא tru נקבל:</u> fls [(λt. λf. t) fls tru => (λf. fls) tru => fls]

<u>אם b הוא fls נקבל:</u>tru [(λt. λf. f) fls tru => (λf. f) tru => tru]

XOR

λb. λc. not (and (b c tru) (c b tru))

נראה שכאשר הערכים של b ו-c שונים נקבל tru, וכאשר עם זהים נקבל fls:
(הערה: מיכוון שפעולת הפקודות fls, tru, and, not כבר ברורה לנו, [מטרת הסעיף היא לא בניית סדרת חישוב] הריצה אליהן בחישובים למטה היא מידית מבלי לרשום את החישוב המלא צעד אחר צעד. לכן הסימן החץ מופיע עם כוכבית)

אם b,c הם tru נקבל:

λb. λc. not (and (b c tru) (c b tru)) =>* not (and (tru c tru) (tru b tru)) =>*

not (and c b) =>* not tru =>*

fls

אם b,c הם fls נקבל:

λb. λc. not (and (b c tru) (c b tru)) =>* not (and (fls c tru) (fls b tru)) =>*

not (and tru tru) =>* not tru =>*

fls

אם b הוא tru וc הוא fls, נקבל:

λb. λc. not (and (b c tru) (c b tru)) =>* not (and (tru c tru) (fls b tru)) =>*

not (and c tru) =>* not fls =>*

tru

אם b הוא fls וc הוא tru, נקבל:

λb. λc. not (and (b c tru) (c b tru)) =>* not (and (fls c tru) (tru b tru)) =>*

not (and tru b) =>* not fls =>*

tru

שאלה 4

A

כן, התוצאה שווה ל-c1.

$scc = \lambda n.\ \lambda s.\ \lambda z.\ s\ (n\ s\ z)$

$c0 = \lambda s.\ \lambda z.\ z$

scc c0 = $(\lambda n.\ \lambda s.\ \lambda z.\ s\ (n\ s\ z))\ \lambda s.\ \lambda z.\ z \Rightarrow$ (E-AppAbs)

$\lambda s.\ \lambda z.\ s\ ((\lambda s.\ \lambda z.\ z)\ s\ z) \Rightarrow$ (E-Abs)

$\lambda s.\ \lambda z.\ s\ z = c1$

B

לא, התוצאה לא שווה ל-c1

$(\lambda n.\ \lambda s.\ \lambda z.\ s\ (n\ s\ z))\ \lambda s.\ \lambda z.\ z \Rightarrow$ (E-AppAbs)

$\lambda s.\ \lambda z.\ s\ ((\lambda s.\ \lambda z.\ z)\ s\ z) \not\Rightarrow$

התוצאה שקולה ל-c1 במובן שהפעלה של הפונקציה הנ"ל על ערכים x,y כלשהם, תוביל לתוצאה זהה להפעלתם על הפונקציה c1:

$\lambda s.\ \lambda z.\ s\ ((\lambda s.\ \lambda z.\ z)\ s\ z)\ x\ y \Rightarrow \lambda z.\ x\ ((\lambda s.\ \lambda z.\ z)\ x\ z)\ y \Rightarrow$

$x\ ((\lambda s.\ \lambda z.\ z)\ x\ y) \Rightarrow x\ ((\lambda z.\ z)\ y) \Rightarrow x\ y$.

$c1 = (\lambda s.\ \lambda z.\ s\ z)\ x\ y \Rightarrow (\lambda z.\ x\ z)\ y \Rightarrow x\ y$

קיבלנו אותה תוצאה לכל x,y , לכן הפונקציות שקולות.

C

newScc = λn. λs. λz. c1 s (n s z)

כאשר הערך של n הינו Church number – אותו נקדם באחד.

השתמשנו בהגדרה של plus ($= \lambda m.\ \lambda n.\ \lambda s.\ \lambda z.\ m\ s\ (n\ s\ z)$)- אשר מקבל שני מספרי church ומחבר אותם, וקבענו בו את m (המחובר הראשון) להיות c1, כלומר נחבר את מספר הקלט (n) תמיד עם c1 – קיבלנו scc.

D

$times = \lambda m.\ \lambda n.\ m\ (plus\ n)\ c0$

power = λm. λn. n (times m) c1

כאשר הפרמטר הראשון (m) הוא המעריך, והפרמטר השני (n) הוא החזקה.

<u>בנינו זאת בצורה דומה לבניית פונקצית הכפל:</u> פעולת חזקה היא הכפלה של המעריך בעצמו כמספר הפעמים של החזקה.

לדוגמא, נבחר מעריך c2 וחזקה c3, נקבל:

(λm. λn. n (times m) c1) c2 c3 =>

(λn. n (times c2) c1) c3 =>

c3 (times c2) c1 =

(λs. λz. s s s z) (times c2) c1 =>

(λz. (times c2) (times c2) (times c2) z) c1 =>

ניתן לראות שאכן קיבלנו שהפרמטר m מייצג את מספר ההכפלות של הפרמטר n בעצמו, כאשר ההכפלה הראשונה היא ב-c1 (כלומר הכפלה שלא משנה את התוצאה) – נמשיך בדוגמא:

(times c2) (times c2) (times c2) c1 =>

(times c2) (times c2) c2 =>

(times c2) c4 =>

c8

E

isZero = λm. m (λb. fls) tru

<u>הגדרה:</u> הפונקציה isZero מקבל מספר church (שהוא בפועל פונקציה שמקבלת 2 פרמטרים) ומעבירה לו כפרמטר ראשון פונקציה שמקבלת פרמטר 1 ומחזירה false **(λb. fls)**, וכפרמטר שני את הערך tru.

<u>הסבר:</u> נבחר מספר church כלשהו ck.
המקרה היחידי בו אין שימוש בפרמטר הראשון **(λb. fls)** במספר ck, הוא כאשר ck=c0.
במקרה הזה פשוט חוזר הערך השני: לכן בהפעלת פונקציה isZero על c0 נקבל tru.
בכל מקרה אחר (ck>c0), נעשה גם שימוש בפרמטר הראשון שהעברנו, ולכן תמיד נקבל fls. (נריץ את הפונקציה **λb. fls** כמספר הפעמים של המספר ck – כל פעם על תוצאת הפונקציה בריצה הקודמת [התוצאה היא תמיד fls] – ולכן התוצאה הסופית היא fls).

שאלה 5

a. הגדרת sum-l שתעבוד ב-lazy evaluation semantics:

g = λf. λk. test (iszero k) c_0 (plus k (f(prd k)))

Y = λf. (λx. f(x x))(λx. f(x x))

sum-l = Y g

b. סדרת חישוב לביטוי sum-l c_1 תחת lazy evaluation semantics:

sum-l c_1 = Y g c_1 =

(λf. (λx. f(x x))(λx. f(x x))) g c_1 =>

(λx. g(x x))(λx. g(x x)) c_1 =>

g((λx. g(x x)) (λx. g(x x))) c_1 =

(λf. λk. test (iszero k) c_0 (plus k (f(prd k)))) ((λx. g(x x)) (λx. g(x x))) c_1 =>

(λk. test (iszero k) c_0 (plus k (((λx. g(x x)) (λx. g(x x))) (prd k)))) c_1 =>

test (iszero c_1) c_0 (plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1))) =

(λl. λm. λn. l m n) (iszero c_1) c_0 (plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1))) =>

(λm. λn. (iszero c_1) m n) c_0 (plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1))) =>

(λn. (iszero c_1) c_0 n) (plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1))) =>

(iszero c_1) c_0 (plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1))) =

תזכורת: בשאלה 4 הפונקציה iszero הוגדרה כך: iszero = λm. m (λb. fls) tru

ולכן המשך סדרת החישוב הוא:

((λm. m (λb. fls) tru) c_1) c_0 (plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1))) =>

(c_1 (λb. fls) tru) c_0 (plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1))) =

((λs. λz. s z) (λb. fls) tru) c_0 (plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1))) =>

((λz. (λb. fls) z) tru) c_0 (plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1))) =>

((λb. fls) tru) c_0 (plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1))) =>

fls c_0 (plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1))) =

(λt. λf. f) c_0 (plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1))) =>

(λf. f) (plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1))) =>

plus c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1)) =

(λm. λn. λs. λz. m s (n s z)) c_1 (((λx. g(x x)) (λx. g(x x))) (prd c_1)) =>

$(\lambda n.\ \lambda s.\ \lambda z.\ c_1\ s\ (n\ s\ z))\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ c_1))$ =>

$(\lambda n.\ \lambda s.\ \lambda z.\ c_1\ s\ (n\ s\ z))\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ c_1))$ =>

$\lambda s.\ \lambda z.\ c_1\ s\ ((((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ c_1))\ s\ z)$ =

ביטוי זה לא מתקדם יותר ב-lazy evaluation, לכן נמשיך לפתח תחת normal order semantics:

$\lambda s.\ \lambda z.\ (\lambda s.\ \lambda z.\ s\ z)\ s\ ((((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ c_1))\ s\ z)$ =>

$\lambda s.\ \lambda z.\ (\lambda z.\ s\ z)\ ((((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ c_1))\ s\ z)$ =>

$\lambda s.\ \lambda z.\ s\ ((((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ c_1))\ s\ z)$ =>

$\lambda s.\ \lambda z.\ s\ ((g((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ c_1))\ s\ z)$ =

$\lambda s.\ \lambda z.\ s\ (((\lambda f.\ \lambda k.\ \text{test}\ (\text{iszero}\ k)\ c_0\ (\text{plus}\ k\ (f(\text{prd}\ k))))\ ((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ c_1))\ s\ z)$ =>

$\lambda s.\ \lambda z.\ s\ (((\lambda k.\ \text{test}\ (\text{iszero}\ k)\ c_0\ (\text{plus}\ k\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ k))))\ (\text{prd}\ c_1))\ s\ z)$ =>

$\lambda s.\ \lambda z.\ s\ ((\text{test}\ (\text{iszero}\ (\text{prd}\ c_1))\ c_0\ (\text{plus}\ (\text{prd}\ c_1)\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ (\text{prd}\ c_1)))))\ s\ z)$ =

$\lambda s.\ \lambda z.\ s\ (((\lambda l.\ \lambda m.\ \lambda n.\ l\ m\ n)\ (\text{iszero}\ (\text{prd}\ c_1))\ c_0\ (\text{plus}\ (\text{prd}\ c_1)\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ (\text{prd}\ c_1)))))\ s\ z)$=>

$\lambda s.\ \lambda z.\ s\ (((\lambda m.\ \lambda n.\ (\text{iszero}\ (\text{prd}\ c_1))\ m\ n)\ c_0\ (\text{plus}\ (\text{prd}\ c_1)\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ (\text{prd}\ c_1)))))\ s\ z)$ =>

$\lambda s.\ \lambda z.\ s\ (((\lambda n.\ (\text{iszero}\ (\text{prd}\ c_1))\ c_0\ n)\ (\text{plus}\ (\text{prd}\ c_1)\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ (\text{prd}\ c_1)))))\ s\ z)$ =>

$\lambda s.\ \lambda z.\ s\ (((\text{iszero}\ (\text{prd}\ c_1))\ c_0\ (\text{plus}\ (\text{prd}\ c_1)\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ (\text{prd}\ c_1)))))\ s\ z)$ =

$\lambda s.\ \lambda z.\ s\ ((((\lambda m.\ m\ (\lambda b.\ \text{fls})\ \text{tru})\ (\text{prd}\ c_1))\ c_0\ (\text{plus}\ (\text{prd}\ c_1)\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ (\text{prd}\ c_1)))))\ s\ z)$ =>

$\lambda s.\ \lambda z.\ s\ ((((\text{prd}\ c_1)\ (\lambda b.\ \text{fls})\ \text{tru})\ c_0\ (\text{plus}\ (\text{prd}\ c_1)\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ (\text{prd}\ c_1)))))\ s\ z)$ =>

$\lambda s.\ \lambda z.\ s\ (((c_0\ (\lambda b.\ \text{fls})\ \text{tru})\ c_0\ (\text{plus}\ (\text{prd}\ c_1)\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ (\text{prd}\ c_1)))))\ s\ z)$ =

$\lambda s.\ \lambda z.\ s\ ((((\lambda s.\ \lambda z.\ z)\ (\lambda b.\ \text{fls})\ \text{tru})\ c_0\ (\text{plus}\ (\text{prd}\ c_1)\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ (\text{prd}\ c_1)))))\ s\ z)$ =>

$\lambda s.\ \lambda z.\ s\ ((((\lambda z.\ z)\ \text{tru})\ c_0\ (\text{plus}\ (\text{prd}\ c_1)\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ (\text{prd}\ c_1)))))\ s\ z)$ =>

$\lambda s.\ \lambda z.\ s\ ((\text{tru}\ c_0\ (\text{plus}\ (\text{prd}\ c_1)\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ (\text{prd}\ c_1)))))\ s\ z)$ =

$\lambda s.\ \lambda z.\ s\ (((\lambda t.\ \lambda f.\ t)\ c_0\ (\text{plus}\ (\text{prd}\ c_1)\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ (\text{prd}\ c_1)))))\ s\ z)$ =>

$\lambda s.\ \lambda z.\ s\ (((\lambda f.\ c_0)\ (\text{plus}\ (\text{prd}\ c_1)\ (((\lambda x.\ g(x\ x))\ (\lambda x.\ g(x\ x)))\ (\text{prd}\ (\text{prd}\ c_1)))))\ s\ z)$ =>

$\lambda s.\ \lambda z.\ s\ (c_0\ s\ z)$ =

$\lambda s.\ \lambda z.\ s\ ((\lambda s.\ \lambda z.\ z)\ s\ z)$ =>

$\lambda s.\ \lambda z.\ s\ ((\lambda z.\ z)\ z)$ =>

$\lambda s.\ \lambda z.\ s\ z$ =

c_1

כלומר קיבלנו שהחישוב של c_1 sum-l מסתיים ב-c_1, כצפוי על פי ההגדרה.

c. אם היינו מחשבים את הביטוי תחת call by value semantics, היינו נכנסים ללולאה אינסופית. תחילת החישוב הייתה נראית זהה לחישוב בסעיף b, ואז מהשורה הרביעית (צבועה בכחול) והלאה כל צעד היה מוסיף הפעלה של g על הביטוי, אך משאיר את הארגומנט בצורה שניתן לקדם, ולכן היינו ממשיכים לקדם אותו עד אינסוף:

g ((λx. g(x x)) (λx. g(x x))) c_1 =>

מכיוון ש-g היא אבסטרקציה, הכלל היחיד בו ניתן להשתמש כעת על פי call by value semantics הוא E-APPL2, כלומר הכלל שבו מקדמים את הארגומנט. זה היה מוביל ל:

g(g((λx. g(x x)) (λx. g(x x)))) c_1 =>

g(g(g((λx. g(x x)) (λx. g(x x))))) c_1 =>

...

ניתן לראות שהביטוי הכחול נשאר זהה, ובכל צעד הוא נעטף בעוד קריאה ל-g. כך סדרת החישוב תמשיך לנצח ולא תסתיים לעולם.

d. הגדרת sum-s שתעבוד ב-call by value semantics:

g = λf. λk. test (iszero k) c_0 (plus k (λa. f(prd k) a))

Z = λf. (λx. f(λy. x x y))(λx. f(λy. x x y))

sum-s = Z g

e. סדרת חישוב לביטוי sum-s c_1 תחת call by value semantics:

sum-s c_1 = Z g c_1 =

(λf. (λx. f(λy. x x y))(λx. f(λy. x x y))) g c_1 =>

(λx. g(λy. x x y))(λx. g(λy. x x y)) c_1 =>

g(λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) c_1 =

(λf. λk. test (iszero k) c_0 (plus k (λa. f(prd k) a))) (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) c_1 =>

(λk. test (iszero k) c_0 (plus k (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd k) a))) c_1 =

(λk. test (iszero k) c_0 (plus k (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd k) a))) (λs. λz. s z) =>

test (iszero (λs. λz. s z)) c_0 (plus (λs. λz. s z) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a)) =

(λl. λm. λn. l m n) ((λm. m (λb. fls) tru) (λs. λz. s z)) c_0 (plus (λs. λz. s z) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a)) =>

(λl. λm. λn. l m n) ((λs. λz. s z) (λb. fls) tru) c_0 (plus (λs. λz. s z) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a)) =>

(λl. λm. λn. l m n) ((λz. (λb. fls) z) tru) c_0 (plus (λs. λz. s z) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a)) =

(λl. λm. λn. l m n) ((λz. (λb. fls) z) (λt. λf. t)) c_0 (plus (λs. λz. s z) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a)) =>

(λl. λm. λn. l m n) ((λb. fls) (λt. λf. t)) c_0 (plus (λs. λz. s z) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a)) =>

(λl. λm. λn. l m n) (fls) c_0 (plus (λs. λz. s z) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a)) =

(λl. λm. λn. l m n) (λt. λf. f) c_0 (plus (λs. λz. s z) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a)) =>

(λm. λn. (λt. λf. f) m n) c_0 (plus (λs. λz. s z) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a)) =

(λm. λn. (λt. λf. f) m n) (λs. λz. z) (plus (λs. λz. s z) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a)) =>

(λn. (λt. λf. f) (λs. λz. z) n) (plus (λs. λz. s z) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a)) =

(λn. (λt. λf. f) (λs. λz. z) n) ((λm. λn. λs. λz. m s (n s z)) (λs. λz. s z) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a)) =>

(λn. (λt. λf. f) (λs. λz. z) n) ((λn. λs. λz. (λs. λz. s z) s (n s z)) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a)) =>

(λn. (λt. λf. f) (λs. λz. z) n) (λs. λz. (λs. λz. s z) s ((λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a) s z)) =>

(λt. λf. f) (λs. λz. z) (λs. λz. (λs. λz. s z) s ((λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a) s z)) =>

(λf. f) (λs. λz. (λs. λz. s z) s ((λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a) s z)) =>

λs. λz. (λs. λz. s z) s ((λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a) s z) =>

ביטוי זה לא מתקדם יותר ב-lazy evaluation, לכן נמשיך לפתח תחת normal order semantics:

λs. λz. (λz. s z) ((λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a) s z) =>

λs. λz. s ((λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) a) s z) =>

λs. λz. s ((λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (λs. λz. s z)) s z) =>

λs. λz. s ((λx. g(λy. x x y)) (λx. g(λy. x x y)) (prd (λs. λz. s z)) s z) =>

λs. λz. s ((g (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y)) (prd (λs. λz. s z)) s z) =

λs. λz. s (((λf. λk. test (iszero k) c_0 (plus k (λa. f(prd k) a))) (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y)) (prd (λs. λz. s z)) s z) =>

λs. λz. s ((λk. test (iszero k) c_0 (plus k (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd k) a))) (prd (λs. λz. s z)) s z) =>

λs. λz. s ((test (iszero (prd (λs. λz. s z))) c_0 (plus (prd (λs. λz. s z)) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (prd (λs. λz. s z))) a))) s z) =

λs. λz. s (((λl. λm. λn. l m n) (iszero (prd (λs. λz. s z))) c_0 (plus (prd (λs. λz. s z)) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (prd (λs. λz. s z))) a))) s z) =>

λs. λz. s (((λm. λn. (iszero (prd (λs. λz. s z))) m n) c_0 (plus (prd (λs. λz. s z)) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (prd (λs. λz. s z))) a))) s z) =>

λs. λz. s (((λn. (iszero (prd (λs. λz. s z))) c_0 n) (plus (prd (λs. λz. s z)) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (prd (λs. λz. s z))) a))) s z) =>

λs. λz. s (((iszero (prd (λs. λz. s z))) c_0 (plus (prd (λs. λz. s z)) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (prd (λs. λz. s z))) a))) s z) =

λs. λz. s ((((λm. m (λb. fls) tru) (prd (λs. λz. s z))) c_0 (plus (prd (λs. λz. s z)) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (prd (λs. λz. s z))) a))) s z) =>

λs. λz. s ((((prd (λs. λz. s z)) (λb. fls) tru) c_0 (plus (prd (λs. λz. s z)) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (prd (λs. λz. s z))) a))) s z) =>

λs. λz. s ((((λs. λz. z) (λb. fls) tru) c_0 (plus (prd (λs. λz. s z)) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (prd (λs. λz. s z))) a))) s z) =>

λs. λz. s ((((λz. z) tru) c_0 (plus (prd (λs. λz. s z)) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (prd (λs. λz. s z))) a))) s z) =>

λs. λz. s (((tru) c_0 (plus (prd (λs. λz. s z)) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (prd (λs. λz. s z))) a))) s z) =

λs. λz. s (((λt. λf. t) c_0 (plus (prd (λs. λz. s z)) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (prd (λs. λz. s z))) a))) s z) =>

λs. λz. s (((λf. c_0) (plus (prd (λs. λz. s z)) (λa. (λy. (λx. g(λy. x x y)) (λx. g(λy. x x y)) y) (prd (prd (λs. λz. s z))) a))) s z) =>

λs. λz. s ((c_0) s z) =

λs. λz. s ((λs. λz. z) s z) =>

λs. λz. s ((λz. z) z) =>

λs. λz. s z =

c_1

כלומר קיבלנו שהחישוב של sum-s c_1 מסתיים ב-c_1, כצפוי על פי ההגדרה.

שאלה 6

a. $f:Bool \to Bool \vdash (f(\text{if false then true else false})) : T$

T הוא Bool. נוכיח זאת:

$$\cfrac{\cfrac{\text{true}:Bool\ ,\ \text{false}:Bool}{f:Bool\to Bool,\ (\text{if false then true else false}):Bool}\ (\text{T-IF})}{f(\text{if false then true else false}):Bool}\ (\text{T-APP})$$

(T-TRUE, T-FALSE)

b. $f:Bool \to Bool \vdash (\lambda x:Bool.\ f(\text{if } x \text{ then true else false})) : T$

T הוא Bool->Bool. נוכיח זאת:

$$\cfrac{\cfrac{\cfrac{\text{true}:Bool,\ \text{false}:Bool}{\underbrace{f:Bool\to Bool}_{\text{מהנתון.}},\ x:Bool \vdash (\text{if } x \text{ then true else false}):Bool}\ (\text{T-IF})}{f:Bool\to Bool,\ x:Bool \vdash f(\text{if}\ldots):Bool}\ (\text{T-APP})}{f:Bool\to Bool \vdash \lambda x:Bool.\ f(\text{if}\ldots):Bool\to Bool}\ (\text{T-ABS})$$

(T-TRUE, T-FALSE)

c. $x:Bool,\ y:Bool,\ f:T \vdash (f\,x\,y):Bool$

T הוא $Bool \to Bool \to Bool$. נבנה עץ:

$$(\text{T-APP})\ \dfrac{x:Bool,\ y:Bool,\ f:Bool\to Bool\to Bool}{}$$

$$(\text{T-APP})\ \dfrac{fx : Bool\to Bool\ ,\ y:Bool}{fxy : Bool\to Bool\to Bool.}$$

d. $\vdash (\lambda x:Bool.\ \lambda y:T.\ y\,x):Bool\to T\to Bool\to Bool$

T הוא $Bool\to Bool\to Bool$. נבנה עץ:

$$(\text{T-ABS})\ \dfrac{x:Bool,\ y:Bool\to Bool\to Bool \ \underset{\text{T-APP}}{\vdash}\ yx : Bool\to Bool}{x:Bool \vdash (\lambda y:Bool\to Bool\to Bool.\ y\,x):Bool\to Bool\to Bool\to Bool\to Bool}$$

$$(\text{T-ABS})\ \dfrac{}{(\lambda x:Bool.\ \lambda y:Bool\to Bool\to Bool.\ y\,x):Bool\to Bool\to Bool\to Bool\to Bool\to Bool.}$$