Assignment 3:

1. נראה כי let מוגדרת כ-special form. למרות שניתן לכתוב ביטויי let כ-למבדות ובכך מראים כי הם אינם הכרחיים לשפה L3, אם נשתמש בביטוי מסוג let בשפה ולא נגדיר אותו כ-special form, הפארסר והאינטרפרטר יתייחסו לביטוי זה כאל application expression ובכך ייווצר מצב שהביטוי לא יחושב כפי שהוא צריך להיות מחושב. ייתכן שיהיו חלקים שלא יחושבו כמו שהם אמורים להיות מחושבים ואף שיהיו חלקים שבכלל לא יחושבו, לדוגמא כמו ה-var-decl בחלק של הbindings.
2. התפקיד של valueToLitExp בשפה L3 הוא הפיכת ערך (Value) לביטוי מסוג literal expression המציין את ערכו. בגישת ה-Applicative Order אנו רוצים לקחת את הארגומנטים אשר כבר חושבו ומוגדרים כערכים ולשבץ אותם בגוף הפרוצדורה, אך זה לא אפשרי מבחינת טיפוסים שכן מה שיש בגוף הפרוצדורה הם expressions ואי אפשר לדוגמא להחליף varRef ב-value (זה לא יתקמפל), לכן על מנת שנוכל לבצע את ההחלפה של הביטויים לפי מודל זה אנו נדרשים להשתמש בvalueToLitExp.
3. הפונקציה valueToLitExp אינה נצרכת ב-Normal-Order מכיוון שבניגוד לגישת ה-Applicative Order, אם יש לנו application-expression האופרנדים של הפרוצדורה אינם מחושבים לפני הפעלתם, אלא רק כאשר מגיעים למצב של הפעלת אופרטור פרימיטיבי. כלומר הארגומנטים שמועברים לפרוצדורה אינם נהפכים לערכים כאשר הם מועברים בתור closures. לפיכך, כאשר מתבצע ApplyClosure הarguments מתקבלים כ- CExps ובכך ניתן לבצע את ההחלפה בbody של הפרוצדורה ללא בעיות קומפילציה.
4. הפונקציה valueToLitExp אינה נצרכת במודל הסביבות. למרות שתהליך חישוב ה-Application Expression במודל הסביבות גורר את חישוב האופרנדים כ-Values עוד לפני הפעלתם, כאשר מגיעים לפעולת ה-applyClosure אין צורך לבצע פעולות על הbody של הפרוצדורה, אלא מה שקורה זה הרחבת הסביבה שהייתה בזמן יצירת הפרוצדורה עם המשתנים וה-Values המתאים שחושבו. בכך, אין צורך בהחלפת ביטויים ולכן גם אין צורך בפונקציה valueToLitExp.
5. הסיבות שבגללן יהיה עדיף לעבוד עם גישת ה-Normal-Order ולא בשיטת ה-Applicative-Order הן:
   א) במקרים מסוימים, בגישת ה-normal order יוחזר ערך מהפרוצדורה ואילו ובגישת ה- applicative order התוכנית תקרוס.
   ב) במקרים מסוימים, בגישת ה-normal order יוחזר ערך מהפרוצדורה ואילו ובגישת ה- applicative order התוכנית עלולה להיכנס ללולאה אינסופית.
   ג) במקרים מסוימים, בגישת ה-normal order יש ביטויים בתוכנית שלא יחושבו ושאין באמת צורך בחישובים, זאת לעומת גישת ה-applicative-order ששם מחושבים כל הביטויים לפני הפעלת הפרוצדורה. במצבים אלו, גישת ה-normal-order יעילה יותר.

   דוגמא:

   (if #t 1 (+ 2 3 4)

   נשים לב שתמיד יוחזר הערך 1 ולכן הביטוי (+ 2 3 4) אשר יחושב בגישת ה-applicative-order ולא בגישת ה-normal-order אינו נדרש ובכך גישת ה-normal-order יעילה יותר בדוגמא זאת.
6. ישנם מקרים בהם גישת ה-applicative-order תהפוך חישובים ליעילים יותר מכפי שהתבצעו בגישת ה-normal-order.

   דוגמא:

```
(
 (lambda (x) (+ x x x x))
   (* 10 2)
)
```

בגישת ה-applicative-order קודם כל מחושבים האופרנדים, כלומר יחושב הביטוי (2 10 *) ויוצב בהתאם, כלומר יתבצע (20 20 20 20 +). בגישת ה-normal-order לעומת זאת, יוצב הביטוי (2 10 *) ויחושב רק בהמשך, כלומר נקבל את הביטוי
(2 10 *) (2 10 *) (2 10 *) (2 10 *) +) וניתן לראות שבגישה זאת נחשב את הביטוי (2 10 *) ארבע פעמים לעומת פעם אחת בגישת ה-applicative-order.

7. א. נבחין כי במודל ה-substitution, פעולת ה-renaming מתבצעת על משתנים שנמצאים בתוך Proc expression, וכאשר נתקלים בביטוי מסוג Proc expression אז מאתרים את המשתנים החופשיים ורק להם עושים את פעולת ה-substitute, כלומר רק למשתנים החופשיים עושים את ההחלפה לערכים המתאימים ולכן אם אין משתנים חופשיים אז לא תתבצע פעולת ההחלפה למשתנים הכבולים. מעבר לכך, הסיבה לכך שמתבצעת פעולת הrenaming היא כאשר יש משתנה חופשי כלשהו ובנוסף משתנה כבול, ובכך משתנה עלול לקבל ערך שהוא לא נכון ובכך התוכנית עלולה להחזיר ערך שגוי. כאשר כל המשתנים הם לא חופשיים אין את הסכנה הזאת, כלומר בסה"כ ניתן להסיק כי לא נדרשת פעולת renaming בהצבה הנאיבית.

ב. כפי שהוסבר בסעיף א, במודל ההחלפה הנאיבי לא נדרש לעשות לבצע לביטוי renaming, מעבר לזה אין הבדלים בין מודל ההחלפה הנאיבי ומודל ההחלפה הרגיל ולכן החוקים לאבלואציה של ההחלפה הנאיבית יהיו שונים רק בטיפול ב-Closure באופן הבא:

בגישה האפליקטיבית:

```
const applyClosure = (proc: Closure, args: Value[], env: Env): Result<Value> => {
        const vars = map((v: VarDecl) => v.var, proc.params);
        ~~const body = renameExps(proc.body);~~
        const litArgs = map(valueToLitExp, args);
        return evalSequence(substitute(proc.body, vars, litArgs), env);
}
```

בגישה הנורמלית:

```
If(isClosure(proc)) {
    const vars = map((p) => p.var, proc.params);
    ~~const body = renameExps(proc.body);~~
    return L3normalEvalSeq(substitute(proc.body, vars, args), env);
}
```

8.