Part I

1. כן. Special Form הוא ביטוי שיש לו שיטת איוולואציה שונה מהרגיל. ע"מ להעריך את Let, צריך קודם להעריך את המשתנים המוגדרים על ידו ולאחר מכן להפעיל עליהם את הClouser המוערך מגוף ה-Let. שיטה זו שונה מסדר האיוולואציה הרגיל בו מעריכים רק את הביטוי הנוכחי. המשמעות היא ש-Let איננו רגיל מבחינת דרך האיוולואציה שלו.
2. משפטים שהם קבילים מבחינה תחבירית אבל בהערכה אפליקטיבית היינו מקבלים שגיאה לעומת הערכה נורמאלית בה היינו עשויים לדלג על חישוב הערך שלהם וכך לא לאתר את השגיאה.
   1. (- #t 1) אופרטור פרימיטיבי שמקבל סוגי משתנים (Types) לא נכונים
   2. (/ n 0) אסורה חלוקה באפס (בעיה אריתמטית)
   3. (-) אופרטור שמקבל כמות לא נכונה של אופרנדים. הבעיה תאותר ב-applyPrimitive
   4. ((Lambda (x) (\* x x)) 1 2) קלואוז'ר שמקבל מספר לא נכון של ארגומטים. אבחנה: הבעיה תאותר ב-applyClouser
3. 3.1 נוסיף לאיחוד של Cexp את Value :  
   <cexp> ::= … | <Value>   
   3.2 ראשית, נסיר את השימוש בפונקציה ValueToLit . נוסיף בדיקה isValue? פונקציה L3applicativeEval ונחזיר את הביטוי עצמו במקרה שכן.  
   3.3 לדעתנו, השיטה הראשונה עדיפה, משום שהיא מאפשרת לנו לשמור על הפרדה בין סינטאקסט אבסטרקטי לבין הערכים בפועל. מנגד, הפעולה ValueToLit מייצרת אובייקט חדש לכל ערך שחושב במסגרת מודל ההחלפה, ולכן בזבזנית בזיכרון.
4. בaplicitive evaluation , קודם מחושבים ערכים מארגומנטים ורק לאחר מכן נעשית אוולואציה של הbody, . לכן, ע"מ לעשות את ההחלפה צריך להחזיר את הvalue לצורת Cexp.  
   לעומת זאת ב-normal evaluation , חישוב הארגומנטים "נדחה" ובעצם לא מתבצע לפני חישוב הbody , לכן אין צורך להשתמש בvalueToLit כי הארגומנטים הנתונים עדיין בצורת Cexp.
5. נורמאל מהיר מאפליקטיב:  
   (define foo (lambda (x y) x))   
   (foo 1 (\* 1 1 1 1 1 1 1 1 1 1 1 1 1 1))  
     
   בשיטה הנורמאלית, במימוש ה"נכון" (כמו שנעשה בחלק 3 בעבודה), בו איוולואציה נעשית רק בהוצאה מהסביבה ולא בהכנסה, נוכל לחסוך את החישוב של הארגומנט השני שהפונקציה foo מקבלת.  
   בשיטת האפליקטיב, נחשב את הערך של כל אחד מהארגומנטים, כולל את החישוב (המיותר) של ההכפלה של 1, ונחזיר 1.  
     
   אפליקטיב מהיר מנורמאל:  
   (define double100 (\* 100 2))

(define sum5 (lambda (x) (+ x x x x x)))

(sum5 double100)  
  
בקוד לעיל, בשיטה הנורמאלית נחליף כל אחד מהמשתנים בsum5 בביטוי (\*100 2) ולאחר מכן נחבר אותם אחד לשני ונאלץ לעשות הערכה של כל אחד מהם.  
לעומת זאת, בהערכה אפליקטיבית נחשב את (\* 100 2) פעם אחת בהעברת הארגומנט לsum5, ונחבר 200 חמש פעמים.

Part III

* במקרה הראשון, איקס מוכנס לתוך הסביבה עטוף בפרומיס והוא מוחזר כפרומיס ללא איוולואציה.  
  במקרה השני, איקס עטוף בפרומיס כמו קודם, אך הערך המוחזר המתבקש הוא 1.
* הבעיה היא שנעשית איוולואציה בהכנסה לתוך הסביבה שזה נוגד את הפרדיגמה של חישוב עצל, כי אין צורך ממשי בחישוב הערך בהכנסה לתוך הסביבה.  
  הפתרון הוא לעטוף את הביטוי המגדיר את המשתנה ב"פרומיס", להכניסו לסביבה ולדחות את החישוב למקרה הצורך.