שאלה 1:

1. כן, ניתן לאתחל אובייקט מסוג box ששומר ערך גם במודל הסביבות. במודל זה, ה-box יהיה טיפוס פרמיטיבי, והפונקציות box, unbox, set-box! שלו יהיו פרמיטיביות. הערך השמור בתוך ה-box לא תלוי סביבה.
2. לא. למרות שאפשר להגדיר איזהשהו set! חלקי שיעבור רק על ערכים ב-GE, ה-set! שאנחנו מכירים, עושה השמה למשתנה שהוגדר ונשמר בסביבה מסוימת, ובמודל ה-substitution אין לנו סביבות לכן לא ברור איפה המשתנה ישמר.
3. לא. האופרטור begin בכל מקרה מחזיר את הערך האחרון, לכן מתעלמים מהערכים המחושבים במקומות שאינם האחרון. מכיוון שבתכנות פונקציונלי טהור, הערכים המחושבים לא נשמרים, והסביבה לא תשתנה, אין ל-begin ערך.
4. ההבדל הוא שב-object identity שני אובייקטים שווים רק אם הם אכן אותו אובייקט בזיכרון (למעשה אם הם שני שמות לאותו דבר). הפרוצדורה לבדיקת השיוויון מהסוג הזה היא eq?. ב-state equality, אובייקטים נקראים שווים אם המצב הפנימי שלהם זהה (שזה סובייקטיבי, ותלוי למה בארכיטקטורה שלנו אנחנו קוראים "זהה"). למשל, אם יצרנו שתי רשימות במקום אחר בזיכרון, עם המספרם 1,2,3, הן תהיינה שונות מבחינת object identity, כי שתיהן במקום אחר בזכרון (ואם למשל, נשים משהו במקום הרשימה הראשונה, השניה לא תושפע), אבל הן תהיינה זהות מבחינת state equality, כי רשימה מוגדרת לפי האיברים שבתוכה. הפרוצדורה לבדיקת שיוויון מהסוג הזה ב-Racket היא equal?.
5. תכנות אימפרטיבי עדיף כי עבור איטרציות מספיק לנו לשמור משתנה אחד שמונה את מספר האיטרציה (ואולי עוד משתנה תנאי), כלומר מקום קבוע בזיכרון. לעומת זאת, בתכנות פרוצדורלי, כדי לממש איטרציה נאלצים לעבוד בצורה רקורסיבית, שבאופן נאיבי היא דורשת מקום לינארי לכמות האיטרציות בזיכרון (כי כל איטרציה פותחת עוד frame בזיכרון שיסגר רק אחרי כל האיטרציות).
6. הוא תומך ב-mutation הן ע"י הכנסת ה-box וה-set-box!, והן על ידי הוספת תמיכה ב-set! ע"י פרוצדורה שמשנה את ערך המשתנה ב-env שאנו מממשים במודל הסביבות.