**Q1.1 Why are special forms required in programming languages? Why can’t we simply define them as primitive operators? Give an example [4 points]**

**שאלה 1.1**

ה- special forms הם אלו שמגדירים את השפה, הוספת special forms לשפה מגדירה בכל פעם שפה חדשה ואילו הוספה של פרימיטיביים מעדכנת את האינטרפטר ולא יוצרת שפה חדשה.

לדוגמא, אם נוסיף טיפוס פרימיטיבי string לשפה L1 רק נרחיב את מבחר האפשרויות שלה לעבודה עם טיפוסים וערכים, אבל לא נשנה את אופן הפעולה של השפה. מנגד אם נוסיף את המבנה if לשפה L1, נוכל ליצור תכניות שעונות על צרכים נוספים ולא רק אלו שהיו מוכרים עד כה.

**Q1.2 Write a program in L1, containing more than one expression, where the evaluation of the program’s expression can be done in parallel (e.g., the interpreter can run a thread for each expression evaluation). Write a program in L1, containing more than one expression, where the evaluation of the program’s expression cannot be done in parallel. [4 points]**

**שאלה 1.2**

הגדרה מקבילית:

define a (+ 1 3)

define b (+ 4 2)

הגדרה שלא מאפשרת תכנות מקבילי לאקספרשנים:

define a 3

define b (+ a 2)

(+ a b)

**Q1.3 Let us define the L0 language as L1 excluding the special form ‘define’. Is there a program in L1 which cannot be transformed to an equivalent program in L0? Explain or give a contradictory example [4 points]**

**שאלה 1.3**

לא קיימת תכנית שניתן להריץ ב-L1 ולא ניתן להריץ ב-L0.

"define" מאפשרת לנו להגדיר משתנה ולשמור בתוכו ערך מסויים,

ב-L0 במקום להשתמש בערך שהגדרנו בעזרת "define" בכל פעם נרשום אותו מחדש.

**Q1.4 Let us define the L20 language as L2 excluding the special form ‘define’. Is there a program in L2 which cannot be transformed to an equivalent program in L20? Explain or give a contradictory example [4 points]**

**שאלה 1.4**

שפה L2 מאפשרת לנו להשתמש בעזרת המילה השמורה "define" בתכנות רקורסיבי,

בשפה L20 ביטלנו את האפשרות להגדיר משתנים בעזרת "define" ולכן לא נוכל להגדיר lambda לפי שם,

ובכך לא נוכל ליצור פרוצדורה שתקרא לעצמה.

(define sqrt-iter

(lambda (x root)

(if (good? x root)

root

(sqrt-iter x (improve x root)))))

**Q1.5 For the following high-order functions in L3, which get a function and a list, indicate (and explain) whether the order of the procedure application on the list items should be sequential or can be applied in parallel: - map - reduce - filter - all (returns #t is the application of the given boolean function on each of the given list items returns #t) - compose (compose a given procedure with a given list of procedures) [10 points]**

**שאלה 1.5**

מתודות שמאפשרות ריצה מקבילית: all, map, filter

ניתן לפתוח thread ראשי שיוצר thread עבור כל אובייקט במערך שעליו מפעילים פונקציה, ולהריץ את כל הפונקציות במקביל. לאחר מכן הthread הראשי יחבר בין תוצאות הפונקציות (שכל Thread הריץ) לפי הסדר. הריצה המקבילית מתאפשרת כי אין תלות בין חישוב לחישוב וסדר החישובים לא משנה.

מתודות שלא מאפשרות ריצה מקבילית:

1. reduce: במתודה זו כל פונקציה שרצה על אובייקט תלויה בתוצאת הפונקציה ולכן כל הפעלה של פונקציה על אובייקט במערך חייבת להתבצע לאחר זו שלפניה, כלומר לפי סדר. לכן לא ניתן לבצע הרצה מקבילית שכן לא ניתן להבטיח את סדר ריצת ה-thread-ים.
2. compose: מתודה זו מבצעת הרכבה של פונקציות ומעבירה את התוצאה מפונקציה אחת כארגומנט לפונקציה השניה וכן הלאה. יש חשיבות לסדר הרכבת הפונקציות ולכן לא נוכל להפעיל פונקציה מבלי שהפעלנו את "הפונקציה הקודמת לה" שכן אנחנו תלויים בערך החוזר מהפונקציה הקודמת. בריצה מקבילית לא נוכל להתחייב לקריאה לפונקציות בסדר כרונולוגי ולכן היא לא תתאפשר.

**Q1.6 Regarding L31 language, as defined in Q3b (below): what is the value of the following program? Explain.**

**שאלה 1.6**

הערך שיוחזר הוא 9,

תחילה הגדרנו את הערך b להיות 1,

אחר כך הגדרנו את הערך c להיות 2,

בהמשך הגדרנו מחלקה, עם 2 משתנים לוקאלים (של המחלקה) a,b .

למחלקה 3 מתודות, first ,second ,f

נגדיר מופע של המחלקה עם השדות 3,4, כלומר a=3 ,b=4.

לבסוף נגדיר מתודה שמקבלת משתנה אחד, כשהיא מפעילה את הפונקציה f של המחלקה, המתודה מקבלת משתנה c ששייך למתודה עצמה, בעת הפעלת f למחלקה אין משתנה לוקלי c ולכן תשתמש במשתנה הגלובלי שהוגדר, כלומר c=2.

לבסוף נבצע את החישוב: 9=2+3+4