מאור עטר – 318301231

גיא עזרא - 207329509

חלק ד: שאלות תאורטיות

א. נכון – כל פונקציה היא אובייקט מסדר ראשון, כלומר ניתן להעביר אותו לפונקציה כארגומנט, להחזיר מפונקציה, ולהכיל במבנה נתונים. פונקציה מסדר גבוה היא אובייקט מטיפוס 'function' ולכן היא מקיימת תכונות של אובייקט מסדר ראשון.

ב. נכון – פונקציה ללא שם (למדה) יכולה להחזיר אובייקטים מטיפוסים שונים לפני תנאי, (if else)למשל באמצעות ביטוי – אם...אז תחזיר אובייקט1, אחרת תחזיר אובייקט2.

ג. לא נכון – הפעלת פונקציה יוצרת סביבה חדשה (ע"י פתיחת מסגרת חדשה והרחבת הסביבה הנוכחית) ולא קשירה.

ד. לא נכון – לפונקציות מובנות כמו map אפשר להעביר כארגומנט גם פונקציות מובנות או בנויות ע"י המתכנת (כמו בדוגמא שנלמדה בכיתה של פונקציה fib, שהועברה כארגומנט לפונקציית map).

```
>>> tuple(map(fib, (1, 2, 3, 4)))
(0, 1, 1, 2)
```

ה. לא נכון – מילון הוא אכן מבנה גמיש לחלוטין עבור הערכים שלו אך לא עבור המפתחות שלו, כלומר המילון מוגבל מבחינת הטיפוס של המפתח, אינו יכול להיות mutable.

ו. לא נכון – לפי שיטת dispatch function עם message passing פונקציה dispatch מקבלת פרמטר message שהוא מועבר אל הפונקציה על מנת לבחור בפעולה המבוקשת לשימוש, אך בנוסף לכך היא יכולה לקבל פרמטר נוסף למשל: value במקרה של גישה אל ערך שנמצא במיקום מסוים. לדוגמא בהרצאה מימשנו את mutable rlist ואנו יכולים לשלוח אל ה dispatch function את ההודעה של getitem ואת המיקום של הערך:

```
def dispatch(message, value=None):
    nonlocal contents
    if message == 'len':
        return len_rlist(contents)
    elif message == 'getitem':
        return getitem_rlist(contents, value)
```

ז. נכון – בשפות עם lexical scoping אכן ניתן לממש טיפוסי נתונים חדשים שהם mutable תוך שימוש בפונקציות והשמה לא לוקאלית (nonlocal). כאשר נרצה לשנות איזשהו ערך שנמצא בסביבה מעל הסביבה הנוכחית (לא כולל הסביבה הגלובלית), נשתמש בהצהרת nonlocal. במקרה של מימוש טיפוס נתונים mutable נהיה מחויבים להשתמש בהגדרה הזו מכיוון שהנתונים שמורים במסגרת מעל ולא תהיה לנו גישה אליה בצורה אחרת. כלומר אפשר לממש טיפוס נותנים שהוא mutable כפונקציה עם השמה לא לוקאלית (nonlocal) לשינוי תוכן האובייקט.