מבנה שפות תוכנה – 3

רקורסיות

תכנות פונקציונלי נקרא "טהור" אם אין בו שינוי מצב, כלומר אין בו משתנים או אובייקטים שמשנים את מצבם הפנימי. כל עיבוד מידע נעשה על ידי יצירת ערכים חדשים כנדרש.

בתרגיל זה נדרש לפתור כל סעיף ב 2 דרכים ע"פ בגישה הפונקציונלית – ללא לולאות ומשתנים כלל, באמצעות:

- רקורסיה רגילה
- רקורסיה זנבית (פונקציה רקורסיבית המתוכננת כך שתאפשר ריצה בסיבוכיות מקום נמוכה יותר מאשר סיבוכיות מקום פרופורציונלית לעומק מחסנית הרקורסיה)

1. צרו באמצעות רקורסיה tuple של המספרים 1-1000 כלומר(1000,....,3,2,1)
2. כתבו פונקציה רקורסיבית המחשבת סכום איברי מערך, הריצו אותה על הtuple שיצרתם (בסעיף הקודם)
3. כתבו פונקציה הפותרת את בעיית הLCM – מכפיל משותף מינימאלי LCM(6,4)=12
4. כתבו פונקציה רקורסיבית בוליאנית isPalindrome המקבלת מספר שלם ובודקת אם הוא פלינדרום למשל 123454321
5. כתבו פונקציה רקורסיבית sortedzip שמקבלת רשימה של רשימות, ממיינת אותן בסדר עולה)בלי למיין במקום, תוך שימוש של הפונקציה sorted, ולא המתודה sort)!! ומחזירה את ה-zip שלהן. למשל:

```
>>> list(sortedzip([[3,1,2],[5,6,4],['a','b','c']])) [(1, 4, 'a'), (2, 5, 'b'), (3, 6, 'c')]
```

Lazy Evaluation, Generators

בכל התרגילים, אם לא נאמר אחרת, יש לכתוב בצורה הפונקציונלית הטובה ביותר.

1. לפניך שתי משימות, את כל אחת מהמשימות חובה לבצע בשתי שיטות:
   - ללא שימוש ב lazy evaluation
   - תוך שימוש ב lazy evaluation

א צרו מערך המכיל את המספרים 0-10000
כתבו את הקוד בפונקציה עצמאית
מדדו זמן הביצוע של הפונקציה ואת גודל האובייקט בזיכרון

ב צרו מהמערך הקודם מערך המכיל רק את 5000 האברים הראשונים
מדדו את זמן הביצוע וגודל הזיכרון
בדקו את הtype של המערך המלא וודאו שיש למערך החדש את אותו הtype

2. כתבו גנרטור לחישוב מספרים ראשוניים, בכל פניה לגנרטור ע"י next הוא יחזיר את המספר הראשוני הבא.

3. ממשו את שאלה 4 מהתרגיל הקודם (חישוב טור טיילור) ע"י גנרטורים
כך שבכל פנייה לגנרטור יתווסף לחישוב האיבר הבא בטור
דוג לפלט עבור הקלט 2 כלומר e^2:

1.0
3.0
5.0
6.333333333333333
7.0
7.266666666666667
7.355555555555555
7.3809523809523805