תרגיל בית 7

הנחיות כלליות:

- קראו בעיון את השאלות והקפידו שהתכניות שלכם פועלות בהתאם לנדרש.
- את התרגיל יש לפתור לבד!
- הקפידו על כללי ההגשה המפורסמים באתר. בפרט, יש להגיש את כל הפתרונות לשאלות יחד בקובץ ex7_012345678.py המצורף לתרגיל, לאחר החלפת הספרות 012345678 במספר ת.ז. שלכם, כל 9 הספרות כולל ספרת ביקורת.
- מועד אחרון להגשה: כמפורסם באתר.
- בדיקה עצמית: כדי לוודא את נכונותן ואת עמידותן של התוכניות לקלטים שגויים, בכל שאלה הריצו את תוכניתכם עם מגוון קלטים שונים, אלה שהופיעו כדוגמאות בתרגיל וקלטים נוספים עליהם חשבתם (וודאו כי הפלט נכון).
- אין לשנות את שמות הפונקציות והמשתנים שכבר מופיעים בקובץ השלד של התרגיל.
- היות ובדיקת התרגילים עשויה להיות אוטומטית, **יש להקפיד על פלטים מדויקים על פי הדוגמאות (עד לרמת הרווח).**
- אופן ביצוע התרגיל: שימו לב, בתרגיל זה עליכם להשלים את הקוד בקובץ המצורף.

- **הרצת טסטר: יחד עם התרגיל קיבלתם קובץ טסטר בשם CodeTests.py . לאחר שפתרתם את כל השאלות, אתם יכולים להעתיק בסוף קובץ הפתרון את הטקסט מקובץ הטסטר. הוא כולל מספר בדיקות בסיסיות כדי לראות שניתן לקרוא לפונקציות. אם הבדיקות עברו בהצלחה, יודפס הטקסט: Congrats!!! All preliminary tests passed!. שימו לב, הטסטר אינו בודק את נכונות התשובות אלא רק שחתימות הפונקציות מומשו נכון.**

- **את התרגיל יש להגיש ללא הטסטר**

שאלה 1
מספרי קטלן

מספרי קטלן הם מספרי טבעיים המופיעים בבעיות שונות בקומבינטוריקה. זוהי הנוסחא לחישוב איבר כללי בסדרת קטלן:

$$C_n = \frac{1}{n+1}\binom{2n}{n} \qquad \text{for } n \geq 0$$

אלו שמונת מספרי קטלן הראשונים (שימו לב שהספירה מתחילה מ-0 ולכן האיבר האחרון הוא C_7)

1, 1, 2, 5, 14, 42, 132, 429

הם מעניינים מאוד – למרות שבנוסחא הישירה לחישובם יש שבר, הרי המספר שהוא מתקבל תמיד שלם! אפשר להבין זאת מכך שיש גם נוסחא רקורסיבית לחישובם:

$$C_0 = 1 \quad \text{and} \quad C_{n+1} = \sum_{i=0}^{n} C_i\, C_{n-i} \quad \text{for } n \geq 0;$$

בתרגיל זה עליכם תחשבו את מספרי קטלן באמצעות הנוסחא הרקורסיבית.

a) ממשו פונקציה **catalan_rec(n)** המקבלת מספר שלם וחיובי **n** ומחזירה את מספר קטלן ה n.

b) ממשו פונקציה**catalan_mem(n,memo=None)** המוגדרת באופן דומה לפונקציה הרקורסיבית, ומשתמשת בממואיזציה על מנת לשפר זמני ריצה.
<u>הדרכה:</u> שימו לב שבחישוב מספר קטלן אנו משתמשים מספר פעמים בחישוב מספרי קטלן הקודמים. ניתן לשמור ערכים אלו במילון כפי שראיתם בתרגול.

דוגמאות הרצה:

```
>>> catalan_mem(4)
14
>>> catalan_rec(5)
42
```

שאלת בונוס

c) בשאלה זו נרצה לבדוק האם השימוש בממואיזציה חוסך לנו קריאות רקורסיביות, וכמה קריאות הוא חוסך. נרצה להוסיף משתנה בשם counter שיספור את מספר הכניסות.
המשתנה מאותחל ל-1, כלומר כאשר קוראים לפונקציה, הכניסה הנוכחית נספרה כבר.
<u>הדרכה:</u>
ממשו את הפונקציות **catalan_rec_with_count(n,counter=1),** **catalan_mem_with_count(n,memo=None,counter=1)**, אשר מוגדרות בדומה לפונקציות שממשתם בסעיפים קודמים, ומחזירות tuple שאיברו הראשון הוא מספר קטלן ה n ואיברו השני הוא מספר הפעמים שהפונקציה נראה על מנת לחשב את מספר קטלן ה n
<u>רמז:</u> מאחר והכניסה הראשונה לפונקציה נספרת, חישבו מתי ערך ה counter יעודכן.

שימו לב – אם הפונקציה שלכם מחזירה tuple, אתם יכולים לגשת לערך הראשון ב tuple באמצעות סוגריים מרובעים. לדוגמא:

```
res=catalan_rec_with_count(7)
print res[0]
print res[1]
-------------output-------------
>>>429
>>>1215
```

דוגמאות הרצה:

```
>>> catalan_rec_with_counter(7)

(429, 1215)

>>> catalan_mem_with_counter(7)

(429, 55)
```

שאלה 2
בעיית הכספומט

a) ממשו פונקציה רקורסיבית **atm_rec(amount,bills,n)** המקבלת מספר חיובי שלם **amount**, וכן tuple בשם **bills** המכיל מספרים חיוביים שלמים המייצגים ערכים שונים של שטרות וכן מספר שלם חיובי **n** המייצג את מספר השטרות הדרוש. הפונקציה תחזיר True אם ניתן להרכיב את הסכום amount באמצעות n שטרות בדיוק שערכיהם נמצאים ברשימה bills (שימו לב – חייבים להרכיב את הסכום ממספר שטרות מדויק). בכל מקרה אחר היא תחזיר False.
שימו לב, בשליחת tuple בעל איבר אחד, יש לשים פסיק אחרי האיבר (1,)
<u>הדרכה</u>: עבור כל ערך שנמצא ב bills קראו קריאה רקורסיבית לפונקציה, עם הערכים המתאימים של amount ו n. אל תשכחו לשים לב מה יהיו תנאי העצירה, שימו לב שסביר שיהיה יותר מתנאי עצירה אחד, היזכרו בתרגיל sublist sum שפתרנו בתרגול.

b) ממשו פונקציה **atm_mem(amount,bills,n,memo=None)** המוגדרת בדומה לפונקציה בסעיף 1, אך הפעם השתמשו בממואיזציה כדי למנוע חישוב כפול.
<u>הדרכה:</u> צרו מילון שמפתחותיו הם tuples של סכומים ומספר השטרות הדרוש, וערכיו הם ערכים בוליאניים.

דוגמאות הרצה:

```
>>>atm_rec(70,(5,10),8)
True
>>>atm_mem(10,(2,3,5),6)
False
>>>atm_rec(100,(20,),5)
True
```

שאלה 3

בעיית מסלול בפירמידה

פירמידה הינה משולש של מספרים כך שלכל מספר יש שני מספריו מתחתיו בפירמידה, פרט למספרים בשורה התחתונה. מסלול בפירמידה מתחיל בראשה, ויכול להמשיך דרך המספר הימני או השמאלי של המספר הנוכחי, עד שהמסלול מגיע לתחתית הפירמידה. בפירמידה שלהלן (4,5,3,8) הינו מסלול. בתרגיל זה המטרה היא למצוא מסלול בעל סכום מקסימלי. סכום המסלול הינו סכום המספרים לאורך המסלול. לדוגמא עבור המסלול (4,5,3,8) הסכום הוא 20. במשולש שלפנינו, המסלול המקסימלי הינו (4,7,4,6) וסכומו הוא 21

4
5 **7**
3 **4** 2
8 3 **6** 1

a) ממשו פונקציה רקורסיבית **max_trail(pyramid)** המקבלת רשימה מקוננת **pyramid** המייצגת פירמידה – כלומר, האיבר הראשון ברשימה יהיה רשימה באורך 1 המייצגת את ראש הפירמידה. האיבר השני יהיה רשימה באורך 2 שתייצג את משקלם של 2 המספרים שתחתיו. לדוגמא עבור הפירמידה בדוגמא נקבל: [[4],[5,7],[3,4,2],[8,3,6,1]]. הפונקציה תחזיר את סכום המסלול המקסימלי.
<u>הדרכה:</u> נתחיל מהשורה העליונה ובכל שלב נרד שורה. נחשב בנפרד את סכום המסלול המקסימלי אם נבחר במספר הימני התחתון או השמאלי התחתון. נחזיר את הטוב יותר מבין השניים. אל תשכחו – מהו תנאי העצירה כאשר יורדים כל פעם שורה אחת מטה?
<u>רמז:</u> ניתן להוסיף פרמטרים לפונקציה שלהם ערכים דיפולטיביים, ולהתייחס למיקום ה row,col בפירמידה, כאל ראשה. max_trail(pyramid,row=0,col=0)

דוגמאות הרצה:

```
>>> max_trail([[4],[5,7],[3,4,2],[8,3,6,1]])
21
>>> max_trail_mem([[3],[6,12],[6,3,10],[4,65,5,6]])
83
```

b) ממשו פונקציה **max_trail_mem(pyramid,row=0,col=0,memo=None)** המוגדרת באותו אופן כמו הפונקציה max_trail ומשתמשת בממואיזציה על מנת לחסוך בחישובים.
<u>הדרכה:</u> נסו לחשוב במה כדאי להשתמש כמפתחות במילון memo – מהו המזהה הייחודי של כל חישוב.