תרגיל בית 5

הנחיות כלליות:

- קראו בעיון את השאלות והקפידו שהתוכניות שלכם פועלות בהתאם לנדרש.
- את התרגיל יש לפתור לבד!
- הקפידו על כללי ההגשה המפורסמים באתר. בפרט, יש להגיש את כל השאלות יחד בקובץ ex5_012345678.py המצורף לתרגיל, לאחר החלפת הספרות 012345678 במספר ת.ז. שלכם, כל 9 הספרות כולל, ספרת ביקורת.
- אופן ביצוע התרגיל: בתרגיל זה עליכם להשלים את הקוד בקובץ המצורף.
- 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).
- השאלות נבדקות באופן אוטומטי. לכן, עליכם לרשום את הקוד שלכם אך ורק במקומות המתאימים לכך בקובץ השלד.
- ניתן להניח כי הקלט שמקבלות הפונקציות תקין (אלא אם נכתב אחרת).
- אין לשנות שמות פונקציות או משתנים שקיימים בקובץ השלד של התרגיל.
- אין למחוק את ההערות שמופיעות בשלד (חוץ מהשורות בהן מופיעה המילה pass).
- אין להשתמש בקריאה לספריות חיצוניות (אסור לעשות **import**).
- מועד אחרון להגשה: כמפורסם באתר.

הנחיות מיוחדות לתרגיל זה:

- **על כל הפונקציות שתכתבו להיות רקורסיביות (פתרונות לא רקורסיביים לא יתקבלו)**
- **אין להשתמש בלולאות (for, while)**
- **הניחו כי הקלט תקין. למשל, אם כתוב לכם שהפונקציה מקבלת רשימה לא ריקה של מספרים אי שליליים, ניתן להניח שזה הקלט שתקבלו, ואין צורך לבדוק את זה בגוף המימוש.**
- **הפעם הכנסנו לנוחיותכם בדיקות בסוף קובץ השלד (אך שימו לב שהן לא בהכרח מכסות את כל המקרים)**

שאלה 1

כתבו פונקציה רקורסיבית `reverse_string` אשר מקבלת כקלט מחרוזת ומחזירה את המחרוזת בהיפוך סדר התווים.

הבהרות:

- ניתן להשתמש ב slicing בשביל הקריאה הרקורסיבית, אך אסור לכם להפוך את המחרוזת באמצעות slicing (ע"י step שלילי). כמו כן, אין להשתמש בשום פונקציה אחרת שמבצעת היפוך מחרוזות, כדוגמת הפונקציה reverse.

דוגמאות הרצה:

```
>>> reverse_string("abc")
'cba'
>>> reverse_string("Hello!")
'!olleH'
```

שאלה 2

כתבו פונקציה רקורסיבית `find_maximum` שמקבלת רשימה של מספרים אי-שליליים (אפס ומעלה) ומחזירה את הגדול מביניהם. במידה והרשימה ריקה, הפונקציה תחזיר את המספר -1.

הבהרה: אין להשתמש בפונקציה המובנית max.

דוגמאות הרצה:

```
>>> find_maximum([9,3,0,10])
10
>>> find_maximum([9,3,0])
9
>>> find_maximum([])
-1
```

הדרכה:

בכל שלב ברקורסיה, אם נפריד את הרשימה לאיבר האחרון ושאר הרשימה, ישנן שתי אפשרויות:

1. האיבר האחרון ברשימה הוא הגדול ביותר
2. האיבר הגדול ביותר נמצא בשאר הרשימה

אם כך, בכל שלב ברקורסיה נחשב את ערכי שתי האפשרויות ונחזיר את הגדול מבניהם.

שאלה 3

מחרוזת נקראת פלינדרום אם קריאתה מהסוף להתחלה ומהתחלה לסוף תניב את אותו רצף תווים. נגדיר שבמחרוזת שהינה פלינדרום יש לפחות תו אחד, התו הראשון שלה זהה לתו האחרון, התו השני זהה לתו שלפי האחרון וכן הלאה. הזהות בין התווים צריכה להיות מוחלטת, Aba אינו פלינדרום כיוון שהתו A אינו זהה לתו a.

לדוגמה, המחרוזות 'a', 'abba' הינן פלינדרומים ואילו 'abcab' אינה פלינדרום.

עליכם לכתוב את הפונקציה הרקורסיבית `is_palindrome` המקבלת מחרוזת לא ריקה s ומחזירה `True` אם המחרוזת הינה פלינדרום, ו-`False` אחרת.

הבהרה: אין להפוך את מחרוזת הקלט או חלקים ממנה כחלק מהפתרון.

דוגמאות הרצה:

```
>>> is_palindrome("aa")
True

>>> is_palindrome("aa ")
False

>>> is_palindrome("caca")
False

>>> is_palindrome("abcbbcba")
True
```

הדרכה: מחרוזת (לא ריקה) היא פלינדרום אם מתקיימים שני התנאים הבאים:

- התו הראשון זהה לתו האחרון.
- תת המחרוזת המורכבת מכל התווים החל מהתו והשני ועד התו לפני האחרון הינה פלינדרום.

כלומר, abcba הוא פלינדרום כיוון שהתו הראשון זהה לתו האחרון, ובנוסף, תת המחרוזת bcb גם היא פלינדרום. זוהי הגדרה רקורסיבית המקטינה את אורך המחרוזת ב- 2 בכל צעד רקורסיה. עליכם להוסיף גם תנאי עצירה מתאים.

שאלה 4

עליכם לטפס במעלה גרם מדרגות בן n (מספר שלם גדול מאפס) מדרגות. בכל צעד טיפוס אתם יכולים לבחור אם לעלות מדרגה אחת בלבד או שתי מדרגות בבת אחת. בכמה דרכים שונות ניתן לטפס לקצה גרם המדרגות?

כתבו את הפונקציה הרקורסיבית climb_combinations שמקבלת מספר שלם גדול מאפס n ומחזירה את מספר האפשרויות לעלות גרם המדרגות בו יש n מדרגות.

דוגמאות הרצה:

```
>>>climb_combinations(3)
3
```

<u>הסבר:</u> יש בדיוק 3 דרכים לעלות 3 מדרגות:

- לעלות מדרגה אחת 3 פעמים
- לעלות שתי מדרגות בבת אחת ואז לעלות מדרגה אחת
- לעלות מדרגה אחת ואז לעלות שתי מדרגות בבת אחת

```
>>>climb_combinations(10)
89
```

שאלה 5

מחרוזת תקינה של סוגריים היא מחרוזת שבה לכל סוגר שמאלי ')' מתאים סוגר ימני '(' בהמשך המחרוזת ולהפך. כלומר, בכל פעם שפותחים סוגריים (עם סוגר שמאלי) צריך לסגור אותם בהמשך (עם סוגר ימני) ולא ניתן לסגור סוגריים במידה ולא הופיע קודם סוגר שמאלי שנשאר פתוח. למשל, המחרוזת '(()())()' תקינה ואילו המחרוזות הללו אינן תקינות:

```
')'
'('
')(('
```

גם המחרוזות הבאות לא תקינות:

```
'(()'
```

<u>הסבר:</u> שימו לב שכל סוגר ימני מבצע סגירה רק לסוגר שמאלי אחד, בדיוק בצורה שבה אנחנו רגילים להשתמש בסוגריים. לכן, נשאר בדוגמה זו סוגר שמאלי פתוח, וזה לא תקין.

```
'()()('
```

<u>הסבר:</u> הסוגר הימני המסומן לא תקין, כי הסוגר השמאלי היחיד לפניו כבר סגור.

כתבו פונקציה רקורסיבית `is_valid_paren` שתקבל מחרוזת המורכבת מתווים שונים (לא רק סוגריים) ותחזיר `True` אם המחרוזת תקינה מבחינת הסוגריים ואחרת `False`.
הפונקציה תקבל ערך נוסף , `cnt` ,אשר יעזור במניית הסוגריים השמאליים והימנים (ראו הדרכה בהמשך).

חתימת הפונקציה כפי שכתובה בקובץ השלד ולהלן, מגדירה שערך ברירת המחדל (default) עבור הפרמטר `cnt` הוא 0. כלומר, אם לא קוראים לפונקציה עם ערך מפורש ל-`cnt`, ערכו יהיה שווה ל 0.

```
def is_valid_paren(s, cnt=0):
```

דוגמאות הרצה:

```
>>> is_valid_paren("(.(a)")
False

>>> is_valid_paren("p(()r((0)))")
True
```

הקריאה האחרונה שקולה לקריאה:

```
is_valid_paren("p(()r((0)))", 0)
```

הדרכה: כשעוברים על מחרוזת, ברגע שפגשנו פתיחה של סוגריים, כלומר '(', צריך לוודא שבהמשך הם נסגרים ע"י ')'. צריך לשים לב גם למקרה שבו פוגשים סוגר ימני מבלי שיש סוגר שמאלי שהוא מתאים לו, למשל, התו השלישי במחרוזת הבאה: "()(()". במחרוזת זו מספר הסוגריים הפותחים זהה למספר הסוגריים הסוגרים, אך מדובר בתבנית לא תקינה. השתמשו ב cnt על מנת לעקוב אחר מספר הסוגריים שנפתחו וטרם נסגרו.