יסודות מדעי המחשב – תרגיל בית 3

מתרגל אחראי: אופיר יעיש

הנחיות כלליות:

- מועד אחרון להגשה: כמפורסם בתיבת ההגשה ב-Moodle.
- מטרת התרגיל הינה לתרגל כתיבה, ולרכוש מיומנות בכלים שנלמדו עד כה בכיתה.
- קראו את העבודה מתחילתה ועד סופה לפני שאתם מתחילים לפתור אותה. ודאו שאתם מבינים את כל המשימות. רמת הקושי של המשימות אינה אחידה.
- את התרגיל יש לפתור לבד! לרבות איסור שימוש בכלים טכנולוגיים כגון ChatGPT.
- בתיבת ההגשה במערכת ה-VPL, ישנו קובץ שלד לכתיבת הקודם שלכם. עבור כל משימה כתבו הפתרון שלכם במקום המתאים למשימה.
- מלבד פקודת ה-pass, אין למחוק שום קטע קוד הנמצא בשלד. עליכם רק להוסיף את הפתרון שלכם בתוכו.
- אין לשנות את שם/שמות ה-Requested files.
- השאלות יבדקו באופן אוטומטי. הפלט שעליכם להחזיר בכל תרגיל צריך להיות בדיוק כפי שנדרש. כמו כן, באופן אקראי יבדקו גם עבודות באופן ידני.
- כאשר תבוצע בדיקה ידנית, תתבצע גם בדיקת Readability - שימו לב שאתם משתמשים בשמות משתנים אינפורמטיביים וכותבים הערות בכל סעיף.
- בדיקה עצמית: כדי לוודא את נכונותן ואת עמידותן של הפונקציות לקלטים שונים, בכל שאלה הריצו אותן עם מגוון קלטים: אלה שמופיעים בדוגמאות וקלטים נוספים עליהם חשבתם. וודאו כי הפלט נכון. הבדיקה תתבצע על מגוון דוגמאות ולא בהכרח אלה שיינתנו פה.
- ניתן להשתמש בחומר הנלמד עד לפרסום העבודה ורק בחומר הזה.
- אין להשתמש בחבילות או מודולים חיצוניים (דוגמאת math), למעט מקרים שבהם צוין אחרת במפורש.
- במידה שלא צוין אחרת, **יש להניח את נכונות הקלט על פי תיאור המשימה**.
- על מנת לקבל ניקוד מלא, יש לענות נכונה על כל השאלות במסמך זה. משקל כל שאלה הוא זהה.
- במידה שלא עניתם על שאלה מסוימת, נא מלאו את הפונקציה בכל מקרה על מנת שהקוד שלכם יצליח לעבוד.

הנחיות חשובות למשימה זאת:

- היצמדו להוראה של כתיבת פונקציות רקורסיבית במידה שהתבקשתם לכך.
- **במשימה זאת כאשר כתוב שעליכם לכתוב פונקציה רקורסיבית, אתם רשאים (ואף לעיתים חייבים) לכתוב פונקציות עזר שיבצעו את התהליך הרקורסיבי כך שהפונקציות שהוגדרו בשאלה יהוו מעטפת לפונקציית העזר הרקורסיבית.**
- לאורך כל העבודה, נאסר השימוש ב-built-in methods של string ו-list למעט הפעלת הפונקציה len.
- היכן שמצוין שיש לפתור את התרגיל בצורה רקורסיבית, חל איסור מוחלט על שימוש ב-for או while במפורש (גם list comprehension). הנחייה זאת כוללת גם את פונקציות העזר או פונקציות המעטפת.
- חלק איסור מוחלט להשתמש במשתנים גלובליים.
- חל איסור מוחלט להגדיר פונקציות בתוך פונקציות. להלן דוגמה לפונקציה inner func המוגדרת בתוך פונקציה:

```python
def outer_func():
    hidden_global = 10

    def inner_func():
        return hidden_global

    return inner_func()
```

- חל איסור מוחלט על שימוש בפונקציות built-in בשפה כגון sum, max וכד׳. ניתן לשאול בפורום אם ישנה אי בהירות לגביי פונקציה מסוימת.
- באופן כללי, אנו לא בודקים את יעילות הקוד שלכם, אך לא יינתן ציון על בדיקה שחרגה את זמן הריצה המקסימלי של מערכת הבדיקה או עומק הרקורסיה המקסימלי.
- **אנו מרשים לעצמנו לבדוק רק חלק מהנחיות האלה בבדיקה הראשונית, ובבדיקה הסופית לבדוק את כל ההנחיות. לדוגמה, ייתכן שבדיקה הראשונית לא ייבדק שימוש במשתנים גלובליים, אך בבדיקה הסופית כן.**

אנא עקבו אחר ההנחיות על מנת להימנע מאי נעימויות בהמשך. כזכור, אין ערעורים פרטניים.

שאלה 1:

ממשו את הפונקציה **הרקורסיבית** linear_sum(x, result) המקבלת רשימה $x = [x_0, x_1, \dots, x_{n-1}]$ של מספרים שלמים וערך שלם result, ומחזירה את הערך הבוליאני True אם קיימים מקדמים $a_i \in \{-1, 0, 1\}$ עבור $0 \le i \le n-1$ כך שהצירוף הלינארי הבא $a_0x_0 + a_1x_1 + \cdots + a_{n-1}x_{n-1}$ שווה בדיוק ל-result. אחרת הפונקציה תחזיר את הערך הבוליאני False.

דוגמאות:

עבור הקריאה linear_sum([2,3,6,7,10], 15) יוחזר הערך True שכן:

$$1 * 2 + (-1) * 3 + 1 * 6 + 0 * 7 + 1 * 10 = 15$$

עבור הקריאה linear_sum([5, 14, 7, 3], 20) יוחזר הערך False שכן לא קיים צירוף לינארי כנ"ל שסכומו 3.

שאלה 2א

תת-קבוצה סדורה (הגדרה): בהינתן 2 מחרוזות str1 ו-str2 נאמר כי str2 היא תת-קבוצה סדורה של str1 אם str1 מכילה את כל התווים של str2 באותו הסדר שבו הם מופיעים ב-str1. לדוגמה, מחרוזת 'abc' היא תת-קבוצה סדורה של 'ladbcfe', אבל אינה תת-קבוצה סדורה של 'lbdacfe' כיוון שהתווים 'a' ו-'b' אינם מופיעים באותו הסדר במחרוזות 'abc' ו-'lbdacfe'.

ממשו את הפונקציה **הרקורסיבית** ordered_subset(str1, str2) המקבלת 2 מחרוזות str1 ו-str2, ומחזירה את הערך הבוליאני True אם מתקיימים 2 התנאים הבאים:

1. str2 היא תת-קבוצה סדורה של str1.
2. כל 2 תווים עוקבים ב-str2 אינם תווים עוקבים ב-str1 (כפי שניתן לראות בדוגמאות להלן).

אם אחד מ-2 התנאים אינו מתקיים (או שניהם), יוחזר הערך הבוליאני False.

הניחו כי str1 אינה ריקה **ומכילה תווים שונים**.

דוגמאות:

עבור הקריאה ordered_subset("ladbcfe", "abc") יוחזר False.

עבור הקריאה ordered_subset("ladbxcfe", "abc") יוחזר True.

שאלה 2ב

ממשו את הפונקציה **הרקורסיבית** k_size_subsets(n, k) המקבלת 2 מספרים שלמים n ו-k, ומחזירה כרשימה את כל תתי הקבוצה בגודל k של הקבוצה {1,2,3,...,n} בסדר כלשהו. כל תת-קבוצה ברשימה המחוזרת תיוצג ע"י מחרוזת של איברי תת הקבוצה. על כל תווי המחרוזת המייצגת תת-קבוצה להיות בסדר עולה משמאל לימין. ניתן להניח כי $0 \le k \le n$ ו-$1 \le n \le 9$.

דוגמאות:

עבור הקריאה k_size_subsets(5, 3) תוחזר הרשימה (עד כדי שינוי בסדר המחרוזות המייצגות תתי-קבוצות):

['123', '124', '125', '134', '135', '145', '234', '235', '245', '345']

עבור הקריאה k_size_subsets(5, 0) תוחזר הרשימה עם המחרוזת הריקה:

['']

שימו לב: עבור כל מחרוזת, על תווי המחרוזת להיות בסדר עולה. לכן לדוגמה, המחרוזת '342' שמייצגת את תת הקבוצה {2,3,4} לא תתקבל, והמחרוזת המייצגת שתתקבל היא '234'.

השאלה הבאה תעסוק במילונים מקוננים (nested).

הגדרה – מילון מקונן (nested) הינו מילון שקיימים בו ערכים מסוג מילון.

ניתן להגדיר את עומק הקינון במילונים באופן הבא:

עומק 0 – למילון אין ערכים מסוג מילון (מילון לא מקונן). לדוגמה:

{1: "a", 2: "b"}

עומק 1 – למילון קיימים ערכים מסוג מילון כאשר העומק המקסימלי שלהם הוא 0. לדוגמה:

{1: {1: "a", 2: "b"}, 2: "b"}

עומק 2 – למילון קיימים ערכים מסוג מילון כאשר העומק המקסימלי שלהם הוא 1. לדוגמה:

{1: {1: "a", 2: "b"}, 2: {1: {1: "a", 2: "b"}, 2: "b"}}

וכן הלאה.

<u>שאלה 3א:</u>

ממשו את הפונקציה dict_depth(d) המקבלת מילון d ומחזירה את עומק הקינון שלו כ-int. בנוסף, אין להניח שום הנחה על סוג הקלט: במידה ש-d אינו מסוג מילון יש לזרוק שגיאה מסוג TypeError עם מחרוזת הודעה לבחירתכם.

<u>הערה:</u> המימוש כאן אינו חייב להיות רקורסיבי לחלוטין. ז״א ניתן להשתמש בלולאות, אך חייבת להיות קריאה רקורסיבית.

<u>דוגמאות:</u>

עבור הקריאה dict_depth({1: "a", 2: "b"})

תוחזר הספרה 0

עבור הקריאה dict_depth({1: {1: "a", 2: "b"}, 2: "b"})

תוחזר הספרה 1

עבור הקריאה dict_depth({1: {1: "a", 2:"b"}, 2: {1: {1: "a", 2: "b"}, 2: "b"}})

תוחזר הספרה 2

עבור הקריאה dict_depth(1)

תיזרק שגיאה מסוג TypeError עם מחרוזת הודעה לבחירתכם. לדוגמה:

TypeError("d is not a dict")

<u>שאלה 3ב:</u>

ממשו את הפונקציה nested_get(d, key) המקבלת מילון, d, ומספר שלם key ומחזירה רשימה של כל הערכים שהמפתח שלהם הוא key **ושאינם מסוג מילון (dict)**.

שימו לב – המילון d יכול להיות מקונן בעומק מסוים, או לא מקונן בכלל (עומק 0). לכן יכולים להיות כמה מפתחות ששווים ל-key. על הפתרון שלכם להיות כללי כך שיוכל להתאים למילונים בעומקים שונים.

ניתן להניח שכל הערכים במילון יהיו או מחרוזות או מילונים נוספים (על המפתחות במילונים אין הנחות מקדימות). בנוסף אין חשיבות לסדר הופעת הערכים ברשימה המוחזרת (אתם יכולים להחזיר אותם באיזה סדר שנוח לכם).

<u>הערה:</u> המימוש כאן אינו חייב להיות רקורסיבי לחלוטין. ז"א ניתן להשתמש בלולאות, אך חייבת להיות קריאה רקורסיבית.

<u>דוגמאות:</u>

עבור הקריאה nested_get({1: "a", 2: "b"}, 2) תוחזר הרשימה:

['b']

עבור הקריאה nested_get({1: {1: "a", 2: "b"}, 2: "b"}, 3) תוחזר הרשימה הריקה:

[]

עבור הקריאה nested_get({1: {1: "a", 2: "b"}, 2: {1: {1: "c", 2: "b"}, 2: "b"}}, 1) תוחזר הרשימה (לא בהכרח בסדר הזה):

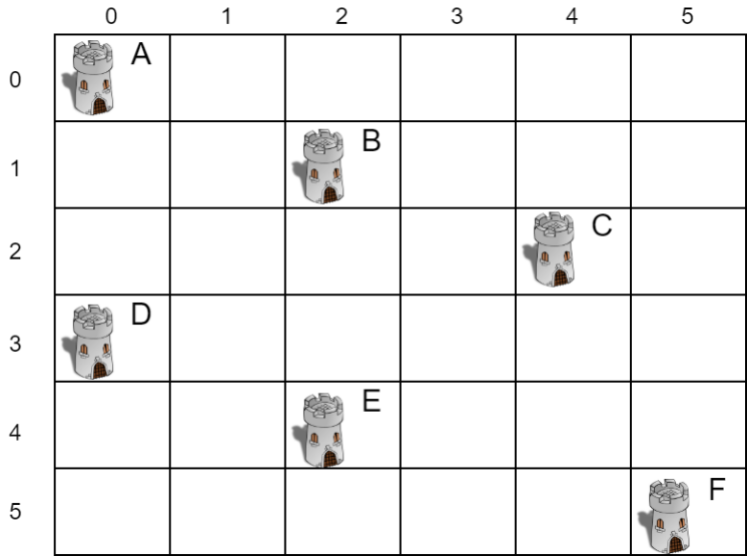
['a', 'c']

<u>שאלה 4:</u>

בשאלה זו נפתור את בעיית n-המגדלים. בבעיה נתון לוח ריבועי. מספרי השורות והעמודות בלוח נספרים מ-0.

על כל משבצת יכול להיות מגדל יחיד. מרחק בין שני מגדלים הוא מספר ההזזות **המינימלי** שנדרש לבצע על מנת להזיז מגדל אחד למקומו של השני. **<u>הזזה אחת של מגדל תוגדר להיות הזזה של המגדל לאחת מארבע המשבצות הסמוכות לו: מעל, מתחת, מימין ומשמאל (לא ניתן להזיז את המגדל באלכסון).</u>**

לדוגמה, המרחק בין מגדל F למגדל E באיור הבא הוא 4 כיוון שנדרשים 4 צעדים כדי להגיע ממשבצת (5, 5) למשבצת (4, 2).



בבעיית n-המגדלים נתון מספר שלם לא שלילי d ולוח ריבועי בעל n שורות ועמודות, ועלינו להציב n מגדלים כך ש:

1) בכל שורה יהיה מגדל אחד בלבד.
2) המרחק בין כל שני מגדלים יהיה גדול ממש מ-d.

נקרא ל-d **"מרחק סף"**.

לדוגמה, עבור d=2, הלוח בתמונה לעיל מציגה פתרון לבעיה.

<u>ייצוג הלוח ומיקום המגדלים בשאלה:</u> נייצג לוח ריבועי בגודל $n \times n$ בעזרת רשימת באורך n. הערך באינדקס i של הרשימה מייצג את העמודה שבה מוצב המגדל בשורה ה-i-ית בלוח.

לדוגמה, הלוח באיור לעיל ייוצג על ידי הרשימה הבאה:

board = [0, 2, 4, 0, 2, 5]

מטרת סעיפי השאלה לחלק את פתרון הבעיה לתת משימות קטנות יותר:

<u>סעיף א':</u>

ממשו את הפונקציה distance(row1, col1, row2, col2) המחשבת את המרחק בין שני מגדלים המוצבים במשבצות (row1, col1) ו-(row2, col2). ניתן להניח כי הקלט תקין.

<u>הערה:</u> המימוש כאן אינו חייב להיות רקורסיבי.

<u>דוגמה:</u>

עבור הקריאה distance(5, 5, 4, 2) יוחזר הערך הערך 4.

סעיף ב':

ממשו את הפונקציה add_tower(board, d, row, col) המקבלת רשימה board המייצגת את הלוח, מספר d המייצג את מרחק הסף, ושני מספרים המייצגים שורה row ועמודה col בלוח. הפונקציה תחזיר את הערך הבוליאני True אם ניתן להציב במשבצת (row, col) מגדל כך שהמרחק שלו מכל מגדל הנמצא **באחת השורות מעליו** יהיה גדול ממש ממרחק הסף d. כמו כן, במידה שניתן להציב את המגדל, הפונקציה תציב את המגדל בלוח. כלומר, הפונקציה תציב את הערך col בתא row של הרשימה board. אם לא ניתן להציב מגדל, על הפונקציה להחזיר את הערך הבוליאני False מבלי לשנות את הרשימה.

ניתן להניח כי הקלט תקין.

שימו לב – עליכם לעשות כאן שימוש בפונקציה שמימשתם בסעיף א'.

הערה: המימוש כאן אינו חייב להיות רקורסיבי.

דוגמאות:

בהינתן הלוח שמיוצג על ידי:

board = [0, 3, 5, 0, 0, 0]

עבור הקריאה add_tower(board, 2, 3, 3) יוחזר הערך False כי המשבצת (3,3) נמצאת במרחק 2 מהמשבצת (1,3) הנמצאת שתי שורות מעליו, ולכן מרחק זה אינו גדול ממש מ-d=2. כמו כן, הרשימה board לא תשתנה.

עבור הקריאה add_tower(board, 2, 3, 1) יוחזר הערך True. כמו כן, תשתנה הרשימה ל-

board = [0, 3, 5, 1, 0, 0]

סעיף ג':

ממשו את הפונקציה **הרקורסיבית** n_towers(n, d) המקבלת גודל לוח ריבועי n ומרחק סף d. הפונקציה תחזיר רשימה באורך n המייצגת פתרון כלשהו לבעיית n-המגדלים עם מרחק סף d. אם לא קיים פתרון, הפונקציה תחזיר רשימה ריקה. ניתן להניח כי הקלט תקין.

המלצה (לא חובה) - לשם פשטות, ניתן להגדיר לוח התחלתי עם n מגדלים הנמצאים בעמודה שבאינדקס 0. כלומר להגדיר רשימה התחלתית המייצגת לוח עם אפסים שעליה יתבצעו השינויים במימוש הפונקציה.

דוגמאות:

1. עבור הקריאה n_towers(6, 2), רשימה אפשרית שתוחזר היא:

[0, 2, 4, 0, 2, 4]

2. עבור הקריאה n_towers(6, 6) תוחזר הרשימה הריקה:

[]