יסודות מדעי המחשב – תרגיל בית 2

מתרגל אחראי: בר שייבט

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.
- 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 1א:

ממשו את הפונקציה הרקורסיבית linear_sum(x, result) המקבלת רשימה של מספרים שלמים וערך שלם result, ומחזירה את הערך הבוליאני True אם קיים צירוף של האיברים, כך שסכומם 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: 10 + 2 + 3 = 15.

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

שאלה 1ב:

תת-קבוצה סדורה (הגדרה): בהינתן 2 מחרוזות str1 ו-str2 נאמר כי str2 היא תת-קבוצה סדורה של str1 אם כל התווים של str2 נמצאים ב str1 ובסדר זהה. 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 ו str2 אינם ריקים ומכילים תווים שונים.

דוגמאות:

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

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

<u>שאלה 2א:</u>

לפנינו סטודנטים הניגשים למבחן ביסודות מדעי המחשב בעל זמן קצוב. המבחן ארוך אך קליל, כלומר, כל שאלה אשר הסטודנט/ית ניגש/ת לפתור, אם נותר לה מספיק זמן, יתקבלו כל הנקודות עבורה.

ממשו את הפונקציה הרקורסיבית

solve_test (questions , total_time)

הפונקציה צריכה למצוא על אילו שאלות כדאי לענות ועל אילו לוותר כדי לקבל את הציון הגבוה ביותר.

questions - משתנה מסוג list המורכב מlistים בעלי המבנה הבא: (time, score), כלומר כל list פנימי מייצג שאלה, time - הזמן שיש להקדיש לשאלה כדי לקבל את הנקודות ו-score הניקוד שמקבלים עבור אותה שאלה.

total_time - הזמן שמוקצה למבחן

הפונקציה צריכה להחזיר משתנה מסוג integer - הציון המקסימלי שאפשר לקבל במבחן.

עבור הקריאה:

questions = [[20,5], [40,9],[30,8]]

solve_test (questions, 55)

יוחזר הציון 13, סכום הניקוד של השאלה הראשונה והשלישית 13 = 8 + 5. שימו לב שלא ניתן לענות על חלק משאלה וייתכן וסטודנט/ית לא ינצלו את מלוא הזמן של המבחן (במקרה זה ניתנו 55 דקות למבחן, ונוצלו 20 + 30 = 50 דקות בלבד).

<u>שאלה 2ב</u>

הסטודנטים רקמו מזימה והחליטו כולם להימנע מלענות על שאלה מסויימת, ולאחר המבחן לדרוש פקטור עבור אותה שאלה משום שהייתה קשה עד מאוד ולראייה אף אחד לא הצליח. במקרה זה הנבחנ/ת לא יקדיש זמן לשאלה אך <u>מחצית מהניקוד שלה</u> יתקבל (עיגול כלפי מטה).

ממשו את הפונקציה

solve_test_with_factor(questions , total_time)

אשר מחשבת את הציון המקסימלי שניתן להשיג, כאשר אפשר לקבל <u>מחצית</u> מניקוד שאלה יחידה מבלי להקצות לה זמן. **שימו לב,** ניתן להשתמש בפונקציה זו כ- wrapper (הגדרה בתרגול 4, שקופית 24) ולקרוא לפונקציה רקורסיבית (עם שם לבחירתכם) בעלת מספר arguments שונה או להוסיף arguments לפונקציה הנוכחית עם ערך default.

עבור הקריאה:

questions = [[20,5], [40,9] ,[35,8] ,[35,7]]

solve_test (questions, 55)

יוחזר הערך 17 - שאלה 1 ו3 נעשו ומחצית מערך שאלה 2 התקבל כפקטור, כלומר 17 = 8+5+(0.5*9)floor

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:

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

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

{ "a": 1, "b" : { “c” : 2 }}

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

{ "a": 1, "b" : { “c” : 2, “d” : {“e” : 3}}}

וכן הלאה.

<u>שאלה 3א:</u>

לפנינו תיקיות במחשב המיוצגות באמצעות מילון מקונן. כאשר ה-key הינו שם התיקייה וה-value הוא תוכן התיקייה. אם הvalue מסוג integer נניח שהkey הוא שם הקובץ (ולא תיקייה) והinteger מייצג את גודל הקובץ במגבייט.

דוגמא להמחשה:

{“my documents”:30, “music”:{“zohar argov”:10, “avihu pinhasov”:20}

בדוגמה זו קיים קובץ my documents בגודל 30 מגה, ובתיקייה music ישנם שני קבצים, במשקל 10 מגבייט ו-20 מגבייט.

ממשו את הפונקציה directory_depth(dir) המחזירה את עומק התיקייה, כלומר, מקבלת מילון dir ומחזירה את עומק הקינון שלו כ-integer. בנוסף, אין להניח שום הנחה על סוג הקלט: במידה ש-dir 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 directory_depth({"a":1, "b":2})

תוחזר הספרה 0

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

תוחזר הספרה 1

עבור הקריאה print(directory_depth({"c": {"a": 1, "b": 2}, "d": {"c": {"a": 1, "b": 2}, "b": 2}}))

תוחזר הספרה 2

עבור הקריאה directory_depth(1)

תיזרק שגיאה מסוג TypeError עם מחרוזת הודעה לבחירתכם. לדוגמה:

TypeError("dir is not a dict")

<u>שאלה 3ב:</u>

ממשו את הפונקציה directory_music_size(dir, is_music=false) המקבלת תיקייה (מילון למעשה) ומחזירה את סך הגודל של כל קבצי המוזיקה. קובץ מוגדר כקובץ מוזיקה אם בשמו או בלפחות מאחת מתיקיות האב שלו, מופיעה המילה music באיזשהי צורה (לדוגמא "music_file", "my_music_is_awesome").

<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:

directory_music_size({“my documents”:30, “music”:{“zohar argov”:10, “avihu pinhasov”:20})

יוחזר המספר 30.

עבור הקריאה:

directory_music_size({“more_music”:40, “music”:{“zohar argov”:10, “avihu pinhasov”:20})

יוחזר המספר 70.

directory_music_size({“more_music”:40, “music”:{“zohar argov”:10, “avihu pinhasov”:20},”just a folder”:{“new1”:5,”new2”:6,”new_last”:7,”new_music”:9}})

יוחזר המספר 79.

<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 Eבאיור 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.

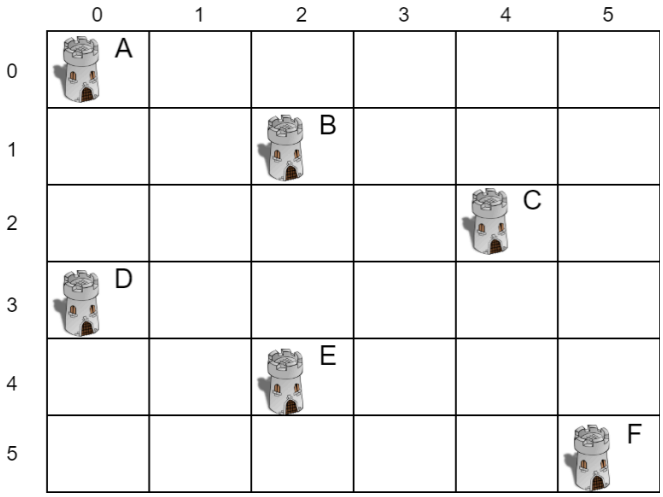
נקרא ל-d **"מרחק סף"**.

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

<u>ייצוג הלוח ומיקום המגדלים בשאלה:</u> נייצג לוח ריבועי בגודל 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.

<u>סעיף ב':</u>

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 **<u>באחת השורות מעליו</u>** 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'.

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

<u>דוגמאות:</u>

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]

<u>סעיף ג':</u>

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.

<u>דוגמאות:</u>

1. עבור הקריאה n_towers(6, 2), רשימה <u>אפשרית</u> שתוחזר היא:

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

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

[]