אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2022

תרגיל בית מספר 2 - להגשה עד 10/11/2022 בשעה 23:55

קיראו בעיון את הנחיות העבודה וההגשה המופיעות באתר הקורס, תחת התיקייה assignments. חריגה מההנחיות תגרור ירידת ציון / פסילת התרגיל.

הגשה:

- תשובותיכם יוגשו בקובץ pdf ובקובץ py בהתאם להנחיות בכל שאלה.
- התשובות בקובץ ה pdf חייבות להיות מוקלדות **ולא** בכתב יד.
- השתמשו בקובץ השלד skeleton1.py כבסיס לקובץ ה py אותו אתם מגישים.
  לא לשכוח לשנות את שם הקובץ למספר ת"ז שלכם לפני ההגשה, עם סיומת py.

- בסה"כ מגישים שני קבצים בלבד. עבור סטודנטית שמספר ת"ז שלה הוא 012345678 הקבצים שיש להגיש הם hw1_012345678.pdf ו- hw1_012345678.py.
- בנוסף, את התשובות לשאלה 6 יש למלא בטופס שנמצא בתוך הרכיב של משוב עמיתים תחת תרגיל 2 ב- moodle.

- הקפידו לענות על כל מה שנשאלתם.

- תשובות מילוליות והסברים צריכים להיות תמציתיים, קולעים וברורים.
  להנחיה זו מטרה כפולה:
  1. על מנת שנוכל לבדוק את התרגילים שלכם בזמן סביר.
  2. כדי להרגיל אתכם להבעת טיעונים באופן מתומצת ויעיל, ללא פרטים חסרים מצד אחד אך ללא עודף בלתי הכרחי מצד שני. זוהי פרקטיקה חשובה במדעי המחשב.

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2022

שאלה 1

בשאלה זו תצטרכו לפתור מספר "חידות תכנות". אין קשר בין הסעיפים. מומלץ קודם להשקיע מחשבה בדרך הפתרון ולאחר מכן לגשת לכתוב את הפתרון.

א. נגדיר את הפונקציה $print\_rectangle(length, width)$ שמקבלת 2 מספרים שלמים חיוביים מהטווח 3 עד 100 – length, width ומחזירה מחרוזת שמכילה מלבן של כוכביות "*" שהאורך שלו הוא length והרוחב שלו הוא width בכוכביות. תזכורת: התו לירידת שורה במחרוזת הוא "\n".

```
>>> print_rectangle(4,5)
*****
*   *
*   *
*****
```

ב. נגדיר את הפונקציה $x\_o\_winner(board)$ שמקבלת לוח משחק של איקס עיגול (להלן חוקי המשחק - https://www.exploratorium.edu/brain_explorer/tictactoe.html ובנוסף אתם יכולים לשחק נגד המחשב) ומחזירה האם המנצח הוא איקס "x" או עיגול "o" ואם אין מנצח מחזירה "no winner". הלוח הוא רשימה של 3 מחרוזות באורך 3 כל אחת. בכל מחרוזת התווים האפשריים הם: "e" – משבצת ריקה, "x" – סמל של שחקן האיקסים, "o" – סמל של שחקן העיגולים.

```
>>> x_o_winner(["eee", "xxx", "eoo"])
'x'
>>> x_o_winner(["eee", "eee", "eee"])
'no winner'
```

ג. בסעיף זה נעסוק בהתאמת סוגרים. בהינתן מחרוזת המכילה 3 סוגי הסוגרים "{ }", "[]", "()", אותיות, מספרים וסימני פיסוק (רווח, נקודה, פסיק, נקודתיים) נרצה לכתוב את הפונקציה $valid\_braces(s)$ שתחזיר לנו True אם הסוגרים במחרוזת s תקינים או False אם הם לא תקינים. סוגרים נחשבים תקינים אם לכל סוגר פותח – "{])" יש סוגר סוגר מתאים לו – "} [(" ובאותו הסדר. כמו כן, כדי שסוגרים יתאימו הם צריכים להיות מאותו הסוג, זאת אומרת "(}" יחזיר False. שימו לב שלדוגמא המחרוזת "({) }" תחזיר False מכיוון שסדר הסוגרים אינו נכון.

```
>>> valid_braces("(ab{cd}ef)")
True
>>> valid_braces("{this(is]wrong")
False
>>> valid_braces("{1:(a,b),2:[c,d)}")
False
```

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2022

<u>שאלה 2</u>

בשאלה זו נממש מספר פונקציות אקראיות תוך שימוש בפונקציה הבסיסית $random.random()$,
<u>וללא שימוש</u> בפונקציות אחרות מהספרייה $random$.
כזכור, הספרייה $random$ מכילה פונקציה בשם $random()$ שמחזירה מספר מטיפוס float בקטע $[0,1)$,
כאשר לכל מספר* יש סיכוי שווה להיבחר:
** ליתר דיוק, לכל מספר <u>שפייתון יודע לייצג</u> בקטע $[0,1)$ יש סיכוי שווה להיבחר.*

```
>>> import random
>>> random.random()
0.13937543523525686
>>> random.random()
0.6376812941041776
```

א. ממשו את הפונקציה $coin()$ שמחזירה True בסיכוי חצי ו-False בסיכוי חצי.

ב. ממשו את הפונקציה $roll\_dice(d)$, שמדמה הטלת קובייה עם d פאות ומחזירה מספר שלם אקראי בין 1 ל-d, כאשר לכל מספר סיכוי שווה להיבחר. הניחו כי $d \geq 2$ ושלם.

ג. השתמשו בסעיף ב׳ על מנת לממש את הפונקציה $roulette(bet\_size , parity)$ אשר מדמה **משחק רולטה** עם הימור בגודל bet_size על ערך $parity \in \{"even", "odd"\}$, ומחזירה את **ערך הזכייה**. <u>חוקי המשחק</u>:
- "מסובבים את הרולטה" – מגרילים מספר אקראי בין 0 ל-36, כולל (שימו לב שניתן להגריל 0 בשונה מסעיף ב׳), כאשר לכל מספר סיכוי שווה להיבחר.
- אם הרולטה נופלת על 0 – הפסדנו, ללא תלות במשתנה $parity$, והפונקציה מחזירה 0.
- אחרת:
  i. אם הזוגיות של המספר שהוגרל תואמת למשתנה $parity$, ניצחנו – והפונקציה מחזירה $bet\_size * 2$. לדוגמה, אם $parity = "odd"$ והוגרל המספר 3 אז ניצחנו. זאת אומרת שאם bet_size היה 7, אז הפונקציה תחזיר 14.
  ii. אחרת, הפסדנו והפונקציה מחזירה 0.

  <u>הנחיה מחייבת:</u> בסעיף זה יש לקרוא לפונקציה $roll\_dice(d)$, **<u>ואסור</u>** לקרוא לפונקציה $random.random()$ (או לכל פונקציה אחרת מספרייה חיצונית).

ד. ממשו את הפונקציה $roulette\_repeat(bet\_size, n)$ המחשבת את **הרווח המצטבר** מ-n משחקים חוזרים ברולטה, על ידי קריאות חוזרות לפונקציה $roulette(bet\_size, parity)$. <u>בכל קריאה</u> לפונקציה $roulette$, השתמשו ב-$coin()$ על מנת להגריל את ערך המשתנה $parity$ שאיתו תקראו לפונקציה. שימו לב שהרווח במשחק יחיד מורכב מסכום הזכייה <u>פחות</u> סכום ההימור. בפרט, הרווח במשחק יחיד הוא **שלילי** כאשר אנחנו מפסידים, **וחיובי** כאשר אנחנו מנצחים. ענו בקובץ ה- pdf: האם בממוצע המשחק משתלם לשחקן?

ה. ממשו את הפונקציה $shuffle\_list(lst)$ המקבלת רשימה של מספרים ו"מערבבת אותם", בדומה למה שעושה הפונקציה shuffle של מחלקת הרשימות שראיתם בהרצאה. אין להשתמש בפונקציות מובנות של list (בפרט כמובן לא בפונקציה shuffle) או בפונקציות של random, אלא רק בפונקציות שכבר מימשתם $coin()$ או $roll\_dice(d)$.

```
>>> shuffle_list([1, 2, 3, 4])
[2, 4, 1, 3]

>>> shuffle_list(["a", "b", "c", "d"])
["c", "b", "d", "a"]
```

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2022

ו. בהרצאה דיברנו על הילוך מקרי פשוט ולא מוטה (simple unbiased random walk) כאשר ההילוך בממד 1.
ממד 1 הוא ציר המספרים השלמים, וצעד בממד הזה הוא או צעד ימינה (+1) או צעד שמאלה (-1). הילוך הוא סדרה של צעדים כאלה. בסעיף זה, ראשית הציר יהיה המספר 0.
קודם כל, ממשו את הפונקציה count_steps(d) אשר מקבלת מספר שלם אי שלילי d ומבצעת הילוך בממד 1 עד שהיא מגיעה למרחק d מהראשית. זאת אומרת, הפונקציה תבצע צעדים עד שהיא תגיע למספר d+ או d-.
על הפונקציה להחזיר את מספר הצעדים שהיא ביצעה עד שהיא הגיע למרחק d.
לאחר שמימשתם את count_steps(d), ממשו את avg_count_steps(d), שתחשב את הממוצע על פני מספר רב של הרצות של count_steps(d). הפונקציה avg_count_steps(d) תריץ את הפונקציה count_steps(d) עם אותו ה-d שקיבלה כפרמטר ותחזיר את ממוצע הצעדים של count_steps(d) על פני כל ההרצות.
בחרו מספר מספיק גדול של הרצות כדי שתשתכנעו שאכן הממוצע מדויק. ניתן כמובן לחפש את התשובה (ואת ההוכחה) באינטרנט, אבל אינכם צריכים לדעת להוכיח את המסקנה בצורה אנליטית (מתמטית), אלא לכתוב קוד שמוכיח אותה בצורה אמפירית.

ז. הרחיבו את הסעיף הקודם כדי לבצע simple unbiased random walk בשני ממדים.
בשני ממדים:
- כל נקודה היא צמד מספרים (x,y) (במקרה הזה שלמים).
- ראשית הצירים היא הנקודה (0,0).
- הצעדים האפשריים הם צעד ימינה (0 ,1+), צעד שמאלה (0 ,1-), צעד למעלה (1+ ,0) וצעד למטה (1- ,0).
- המרחק בין 2 נקודות מחושב על ידי המרחק האוקלדי ביניהן – $d((a1,a2),(b1,b2)) = \sqrt{(a1-b1)^2+(a2-b2)^2}$.

ממשו את הפונקציה count_steps_2dims(d) שמקבלת מספר שלם אי שלילי d ומבצעת הילוך מקרי במרחב הדו ממדי שמתחיל בראשית הצירים במרחק הזה ונגמר במרחק d (ע״פ המרחק האוקלידי) מראשית הצירים. על הפונקציה count_steps_2dims(d) להחזיר את מספר הצעדים שבוצעו עד שההילוך הגיע למרחק הרצוי.

שאלה 3

בשאלה זו נעסוק במימוש פעולות אריתמטיות על מספרים בייצוג בינארי.

להלן מספר הערות והנחיות התקפות לכלל הסעיפים בשאלה:

- לאורך השאלה נייצג מספרים בינאריים באמצעות מחרוזת המכילה את התווים "0" ו-"1" בלבד.
- לאורך השאלה אין לבצע המרה של אף מספר בינארי לבסיס עשרוני או לכל בסיס אחר. בפרט, אין להשתמש כלל בפונקציות int ו-bin של פייתון או בפונקציה convert_base שראינו בתרגול 3.
- לאורך השאלה, ניתן להניח כי מחרוזת הניתנת כקלט היא "תקינה", כלומר, מכילה אך ורק את התווים "0" ו-"1", וכי התו השמאלי ביותר במחרוזת הוא "1" (מלבד המחרוזת "0" אשר מייצגת את המספר 0). בפרט, מחרוזת למספר שאינו אפס לא תכיל אפסים מובילים והמחרוזת המייצגת את אפס תכיל "0" יחיד.
- לכל פונקציה בשאלה אשר מחזירה כפלט מחרוזת בינארית יש לוודא כי המחרוזת תקינה על פי ההגדרה הקודמת. (למשל, הפלטים "0100" ו-"000" אינם תקינים ואילו הפלטים "100" ו-"0" תקינים).
- לאורך השאלה נעבוד עם מספרים אי-שליליים בלבד. בפרט, ניתן להניח כי המחרוזות הבינאריות הניתנות כקלט לפונקציות השונות מייצגות מספרים אי-שליליים בלבד.
- הרצה של הפונקציות בשאלה על מחרוזות באורך של 10 ספרות צריכה להסתיים בזמן קצר (לכל היותר שניה).

א. ממשו את הפונקציה $inc(binary)$ (קיצור של increment) אשר מקבלת מחרוזת המייצגת מספר שלם אי שלילי בכתיב בינארי (כלומר מחרוזת המורכבת מאפסים ואחדות בלבד). הפונקציה תחזיר מחרוזת המייצגת את המספר הבינארי לאחר תוספת של 1.
להלן המחשה של אלגוריתם החיבור של מספרים בינאריים (בדומה לחיבור מספרים עשרוניים עם נשא (carry)):

```
          1   _(carried digits)

        1 0 1  (binary)
+           1
-------------
=       1 1 0
```

<u>הנחיה מחייבת</u>: יש לממש את האלגוריתם בהתאם להמחשה: ישירות באמצעות לולאות.

דוגמאות הרצה:

```
>>> inc("0")
'1'
>>> inc("1")
'10'
>>> inc("101")
'110'
>>> inc("111")
'1000'
>>> inc(inc("111"))
'1001'
```

ב. ממשו את הפונקציה $add(bin1, bin2)$ אשר מקבלת שתי מחרוזות המייצגות מספרים אי שליליים שלמים בכתיב בינארי (כלומר מחרוזות המורכבות מאפסים ואחדות בלבד). הפונקציה תחזיר מחרוזת המייצגת את המספר הבינארי המתקבל מחיבור bin1 ו-bin2.
<u>הנחיה מחייבת</u>: יש לממש את האלגוריתם בהתאם להמחשה בסעיף א׳ : ישירות באמצעות לולאה ואין להשתמש בפונקציה inc.

```
>>> add("1","0")
'1'
>>> add("1","1")
'10'
>>> add("11" ,"110")
'1001'
```

ג. ממשו את הפונקציה $pow\_two(binary, power)$ אשר מקבלת מחרוזת המייצגת מספר בינארי אי שלילי ומספר אי שלילי (מטיפוס int). הפונקציה תחזיר את הייצוג הבינארי של $binary$ כפול 2 בחזקת power, זאת אומרת שהפונקציה תחזיר את הייצוג הבינארי של $binary * 2^{power}$.

```
>>> pow_two("1010", 2)
'101000'
>>> pow_two("1", 3)
'1000'
```

ד. ממשו את הפונקציה $div\_two(binary, power)$ אשר מקבלת מחרוזת המייצגת מספר בינארי אי שלילי ומספר אי שלילי (מטיפוס int). הפונקציה תחזיר את הייצוג הבינארי של $binary$ חלקי 2 בחזקת power מעוגל כלפי מטה, זאת אומרת שהפונקציה תחזיר את הייצוג הבינארי של $\left\lfloor \frac{binary}{2^{power}} \right\rfloor$.

```
>>> div_two("1010", 2)
'10'
>>> div_two("1", 3)
'0'
```

ה. ממשו את הפונקציה leq(bin1, bin2) אשר מקבלת שתי מחרוזות המייצגות מספרים שלמים אי שליליים בכתיב בינארי (כלומר מחרוזות המורכבת מאפסים ואחדות בלבד). הפונקציה תחזיר True אם bin1≤bin2 ו-False אחרת.

```
>>> leq("1010","1010")
True
>>> leq("1010","0")
False
>>> leq("1010","1011")
True
```

ו. ממשו את הפונקציה $to\_decimal(binary)$ אשר מקבלת מחרוזת המייצגת מספר בינארי אי שלילי וממירה אותו למספר דצימאלי, זאת אומרת מחזירה int שהוא הערך של המחרוזת. אסור להשתמש בפונקציה זו באף אחד מהסעיפים האחרים.

```
>>> to_decimal("1000")
8
>>> to_decimal("1001")
9
>>> to_decimal("1")
1
```

ז. בהרצאה ראינו כי מספר בבסיס b בעל d ספרות דורש בבסיס c מספר ספרות השווה ל- $\lceil d * \log_c b \rceil$. הוכיחו טענה זו. **<u>כתבו בקובץ ה-PDF את ההוכחה.</u>** הערה: אין צורך להסתבך בהוכחה ארוכה, ההוכחה מכילה שורות בודדות, ומסתמכת על החסמים העליון והתחתון שראינו גם כן בהרצאה לגודלו של מספר בעל n ספרות בבסיס b.

ח. הסיקו מהו "יחס ההמרה" מבחינת מספר הספרות במעבר מבסיס 2 לבסיס 16. הסבירו את תשובתכם. **<u>כתבו בקובץ ה-PDF את התשובה וההסבר.</u>**

<u>שאלה 4</u>

בשאלה זו נעסוק בטבלאות. דרך בסיסית לייצג טבלה ב-python היא רשימה של רשימות. אורך רשימת העל היא מספר השורות, וכל תת רשימה היא שורה בטבלה. זאת אומרת, שאת הטבלה הבאה:

1	"Red"	3.2
2	"Green"	"blue"
"a"	"b"	"c"

נייצג כרשימה: [[1, "Red", 3.2], [2, "Green", "blue"], ["a", "b", "c"]].

להלן 3 פונקציות לעבודה עם טבלאות:

```
def create_table(n, value): # creates a n by n table
     table = [[value] * n] *n
     return table

def set_value(table, i, j, value):
     table[i][j] = value

def get_value(table, i, j):
     return table[i][j]

def print_table(table):
     st = ""
     for I in range (len(table)):
          st += str(table[i]) + '\n'
     return st
```

א. הקוד מכיל שגיאה הגורמת להתנהגות בלתי רצויה, לדוגמא:

```
>>> t = create_table(3,0)
>>> set_value(t, 1, 1, 'a')
>>> print(print_table(t))
[0, "a", 0]
[0, "a", 0]
[0, "a", 0]
```

כתבו והסבירו **<u>בקובץ ה-PDF</u>** באיזו שורת קוד יש טעות, מדוע היא גורמת לבעיה ותנו שורת קוד חלופית אשר מתקנת אותה.

ב. להלן פונקציה:

```
def foo():
    x = "abcd"
    y = x
    x += "efg"
    lst = [x, y, 3, print]  # breakpoint 1
    lst[2] = 5.4
    lst[0] = 40  # breakpoint 2
    def goo(x, y):
        k = 10
        lst2 = [x, y, foo, k, lst]
        lst2[0] = "hyjk"
        return lst2  # breakpoint 3
    big_lst = goo(x,y) # breakpoint 4
foo()
```

ציירו והסבירו **<u>בקובץ ה-PDF</u>** את תמונת הזיכרון מבחינת מרחב הכתובות ומרחב השמות של הפונקציה foo לאחר שהמפרש מבצע כל אחת מהשורות בהן מופיע ההערה breakpoint. באופן חריג, אפשר בסעיף הזה לצייר את התשובה על דף ולצרף את הציור כל עוד הציור ברור לחלוטין ונסרק על ידי אפליקציית סריקה. יש לזכור, כמו שנאמר בהרצאה שהאתר "python tutor" לא תמיד משקף באופן מדויק את תמונת הזכרון ולכן מומלץ לבחון את הדברים באמצעות בדיקת כתובות הזכרון.

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2022

שאלה 5

עבור מספר טבעי n נגדיר את $\boldsymbol{s(n)}$ להיות סכום כל המחלקים של n, לא כולל n עצמו.
לדוגמה, המחלקים של 4 הם $\{1,2\}$, ולכן $s(4) = 1 + 2 = 3$ (שימו לב ש-2 נספר פעם אחת).
מספר טבעי n נקרא **מספר משוכלל** (Perfect Number) אם $s(n) = n$. לדוגמה, 6 הוא מספר משוכלל כי:

$$s(6) = 1 + 2 + 3 = 6$$

א. ממשו את הפונקציה $divisors(n)$ המחזירה רשימה של כל המחלקים של n, לא כולל n, בסדר עולה.
הנחיה מחייבת: יש לממש את הפונקציה באמצעות List Comprehension.
דוגמת הרצה:

```
>>> divisors(6)
  [1, 2, 3]
>>> divisors(7)
  [1]
```

ב. ממשו את הפונקציה $perfect\_numbers(n)$ המחזירה רשימה של n המספרים המשוכללים הראשונים. **כתבו בקובץ ה-PDF** את זמני הריצה עבור $n = 3$. מה קורה עבור $n = 5$?
דוגמת הרצה:

```
>>> perfect_numbers(1)
  [6]
>>> perfect_numbers(2)
  [6, 28]
```

ג. מספר טבעי n נקרא **מספר שופע** (Abundant Number) אם $s(n) > n$. לדוגמה, 12 הוא מספר שופע כי:

$$s(12) = 1 + 2 + 3 + 4 + 6 = 16 > 12$$

ממשו את הפונקציה $adundant\_density(n)$ אשר מחשבת את צפיפות המספרים השופעים מ-1 עד n, כלומר את היחס:

$$\frac{|\{k \in \mathbb{N} \mid k \leq n \text{ and } k \text{ is abundant}\}|}{n}$$

ערך ההחזרה צריך להיות מספר מטיפוס float בקטע $[0,1]$.
דוגמת הרצה:

```
>>> abundant_numbers(20) # 12, 18, 20 are abundant numbers
  0.15
```

ד. ידוע כי צפיפות המספרים השופעים, כש-n שואף לאינסוף, היא בין 0.2474 ל-0.2480 (צפיפותם המדויקת היא שאלה פתוחה). על מנת להשתכנע בנכונות הטענה, הריצו את הפונקציה עבור ערכים $n = 50, 500, 5000$ **וכתבו את התוצאות בקובץ ה-PDF**. סכמו בקצרה את הממצאים.

ה. מספר טבעי n נקרא **מספר דמוי משוכלל** (Semi-perfect number) אם הוא שווה לסכום של **כל או חלק** מהמחלקים שלו (לא כולל n עצמו, ומבלי לסכום את אותו הגורם יותר מפעם אחת). למשל, המספר 18 הוא מספר דמוי משוכלל: המחלקים של 18 הם [1, 2, 3, 6, 9], ואכן

$$18 = 3 + 6 + 9$$

ממשו את הפונקציה $semi\_perfect\_3(n)$ אשר מקבלת מספר n ובודקת האם ניתן לכתוב אותו **כסכום של 3 מהמחלקים שלו**. אם כן, הפונקציה תחזיר את רשימת המחלקים שסכומם שווה ל-n, בסדר עולה. אחרת, הפונקציה תחזיר את הערך None.

הערה:

- במידה שיש יותר משלשה אחת מתאימה, החזירו שלשה כלשהי.

דוגמת הרצה (שימו לב ש-20 מספר דמוי משוכלל, אבל אינו סכום של אף שלשה מהמחלקים שלו):

```
>>> semi_perfect_3(18)
   [3, 6, 9]
>>> semi_perfect_3(20) # 20 = 1 + 4 + 5 + 10
   None
```

שאלה 6

שאלה זו היא המשך תהליך משוב העמיתים אותו התחלתם בתרגיל הבית הקודם. בשלב זה, תחולקו במהלך הימים הקרובים לקבוצות עבודה של כ-4 סטודנטים וסטודנטיות על ידי צוות הקורס. כל חבר וחברה בקבוצה יקבלו מימושים של הפונקציה max_even_seq שהוגשו על ידי יתר חברי וחברות הקבוצה, אותם יהיה עליהם להעריך על פי ההנחיות בסעיפים הבאים. אל הפתרונות תוכלו לגשת ברכיב משוב עמיתים במודל, שם גם עליכם להזין את ההערכות. **ההערכות יתבצעו בצורה אנונימית לחלוטין אך יש להקפיד על שפה מכבדת**. בדומה לשלב הקודם בתהליך, הניקוד לשאלה זו יינתן על עצם הגשתה ולא על איכותה. עם זאת, אנחנו מעודדים אתכם להשקיע בה מחשבה כבכל שאלה אחרת.

א. חישבו על ארבעה קלטים לבדיקת **הנכונות** של הפונקציה max_even_seq. נסו לכסות את כל המקרים התקינים האפשריים, כולל מקרי קצה ומקרים מיוחדים. בדקו את הנכונות כל אחד משלושת הפתרונות על ארבעת הקלטים שבחרתם[1], והזינו כ"ציון לאמת מידה 1" את מס' הבדיקות שהפתרון עבר בהצלחה (לדוג', אם פתרון כלשהו החזיר תשובה נכונה עבור שני קלטים ותשובה שגויה עבור שני האחרים, הציון שיוזן הוא 2). ב"הערה לאמת מידה 1", פרטו אילו בדיקות הרצתם בפורמט הבא:

```
n: _____, k: ___, expected result: _____, actual result: ______
...
```

ב. חישבו על ארבעה קלטים למדידת **הסיבוכיות** של הפונקציה max_even_seq. נסו לבחור קלטים גדולים ושונים אחד מהשני ככל הניתן. מדדו את היעילות של כל אחד משלושת הפתרונות <u>ושל הפתרון שלכם</u> על ארבעת הקלטים[1,2], והזינו כ"ציון לאמת מידה 2" את מס' הקלטים עבורם הפתרון היה מהיר יותר מהפתרון שלכם (לדוג', אם פתרון כלשהו היה מהיר יותר עבור שלוש קלטים ואיטי יותר עבור הרביעי, הציון שיוזן הוא 3). ב"הערה לאמת מידה 2", פרטו אילו מדידות ביצעתם בפורמט הבא:

```
n: _________, k: ___, your time: _________, my time: _________
...
```

ג. העריכו את **הבהירות** של כל אחד משלושת הפתרונות מציון 0 (לא בהיר בכלל) ל-4 (בהיר לחלוטין) והזינו את הניקוד כ"ציון לאמת מידה 3". מומלץ לקחת בחשבון את העקרונות שהועלו בהרצאה (שמות משמעותיים למשתנים, צריכת זיכרון, טיפול במקרי קצה, מבנה לולאות, שכפול קוד, פשטות וסוכר תחבירי), אך ניתן גם להתייחס לדברים אחרים לפי ראות עיניכם. ב"הערה לאמת מידה 3", נמקו כמיטב יכולתכם מדוע זה הניקוד שבחרתם לתת (זכרו: הערות לשיפור יכולות להועיל רק אם הן ברורות ולא מעליבות).

<u>סוף.</u>

---

[1] במקרה שהפתרון לא ניתן להרצה, בדקו אם ניתן לבצע בו תיקונים קטנים כך שבכל זאת תוכלו להריץ אותו. במידה שלא, כתבו "הקוד של הפתרון לא ניתן להרצה" כהערה לאמת המידה.

[2] את המדידה יש לבצע בדומה <u>לשאלה 3 בתרגיל בית 1</u>. מומלץ לחזור על המדידות מספר פעמים ולהשוות בין זמני הריצה הממוצעים.