

**דף סיכום בבחינה**

מספר שאלה	הערה	ליקוד מרבי	צימל
1.1		4.00	3.00
1.2		6.00	1.00
1.3		6.00	6.00
1.4		4.00	4.00
1.5		10.00	2.00
2.1		5.00	4.00
2.2	פתרונות לא ברור בכלל.	12.00	4.00
2.3		8.00	6.00
2.4		10.00	2.00
2.5		7.00	1.00
3.1		6.00	4.00
3.2		15.00	2.00
3.3		10.00	0.00

**זינן בבחינה סופי : 39.00****הבחינה הבודקה בעמודים הבאים**

## הוראות לנבחן בצדיו השני של הדף

**אין לכתוב מעבר לשוליים משני צידי הדף | אסור לתלוש דפים קון המחברת**

041847

נא לרשום את ציון הבדיקה וקודדו

בהתאם לדוגמא שמעבר לד'

07/03/2022 2310019 1 18/16  
מועד: 2/2022

הנדסת נתוניים  
מבוא למדעי המחשב בשפט פ'יתון  
03821111101102



2310019018

המחלקה	שם
תאריך בדינה	
מקצוע בדינה	

טבלת עזר לכיקום היניקוד	
מספר	סמל שאלה
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
	סה"כ

רשמו וקודדו את הציון כאו	
יחסית	עשרה
<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> 1	<input type="checkbox"/> 0
<input type="checkbox"/> 9	<input type="checkbox"/> 9
<input type="checkbox"/> 8	<input type="checkbox"/> 8
<input type="checkbox"/> 7	<input type="checkbox"/> 7
<input type="checkbox"/> 6	<input type="checkbox"/> 6
<input type="checkbox"/> 5	<input type="checkbox"/> 5
<input type="checkbox"/> 4	<input type="checkbox"/> 4
<input type="checkbox"/> 3	<input type="checkbox"/> 3
<input type="checkbox"/> 2	<input type="checkbox"/> 2
<input type="checkbox"/> 1	<input type="checkbox"/> 1

ציון הבדיקה:

יש לקודד את ציון הבדיקה ע"י סימון X בקבוקיות משמאל, לצורר סדרה אוטומטית של הציון.

X שסומן בטנוות ניתן למחוק ע"י השורה.

שם המרצה \_\_\_\_\_  
חתימה \_\_\_\_\_  
תאריך \_\_\_\_\_

עליך להניח תעודה מזהה עם תמונה על השולחן | להשתיק את הטלפון הסלולרי ולהפסיק לתקיק כל מכשיר אלקטרוני  
סטודנט הזכאי להארכת זכרן חייב להציג אישור מודפס (לא אלקטרוני)

הצהרה

הרini מצהיר/ה כי קראתי את ההוראות לנבחן בדף האחוריו למחברת. ואני ברשותי טלפון סלולרי, שעון חכם או כל מכשיר אלקטרוני אחר.

הנחיות למורים: התזרים להלן מודלים כיצד לסכם ציון (אופציונלי) בטבלה משמאלי, ובטבלה מימין (חוובה) לרשום ולקודד את ציון הבדיקה על שער המחברת (מעבר לדף).

טבלת עזר לטסוקם הנקודות		נא לתרשם את ציון הבדיקה ולקודדו בהתאם לדוגמא שמעבר לדף		
מספר השאלה	מספר הניקוד	אחדות	שורה	מאות
23	1		7	4
17	2	1	0	9
24	3	1	1	8
10	4	X	7	7
	5		6	6
	8		5	5
	9		4	X4
	10		3	3
	11		2	2
	12		1	1
	74			סה"כ

רשמו וקודדו את הציון כאן				
ציון הבדיקה:	ש לקודד את ציון הבדיקה ע"י סימון X בזווית משמאל, לצורך סריקה אוטומטית של הציון.			
שם המרצה:	X שסומן בטענות ניתן למבחן ע"י השחורה.			
חתימה:				
תאריך:				

#### הוראות לבוחן

- 1: ש להשalte להזאת זמינות/זאת/וות
- 2: חל אסור להחזיר חומר עזר כלשהו בכיתה או מחוץ לה, מהvr למוחר לפוי הנחיות המרצה. החזקת חומר אסור מהו הוראה בבדיקה ועלולה להוביל להרחקה מהלימודים לתקופה קצרה או לנצח.
- 3: עוז בטודונט את חדר הבדיקה לאחר חלוקת השאלונים, דיו כדי "בוחן" בבדיקה.
- 4: אסור לנבחן לשוחח בזמן הבדיקה, או לעוזב את מקומו ללא קבלת רשות.
- 5: טודונט רשאי לבחור אם להשתמש בעט או בעפרון. במקרה שכחובן הבדיקה בעפרון נבחינה לא נטרקה, לא יהיה ניתן לערער על הבדיקה. אם הבדיקה נטרקה יהיה רשיין לערער על הבדיקה ובתנאי שאיכות הסריקה תקינה. יש לציין

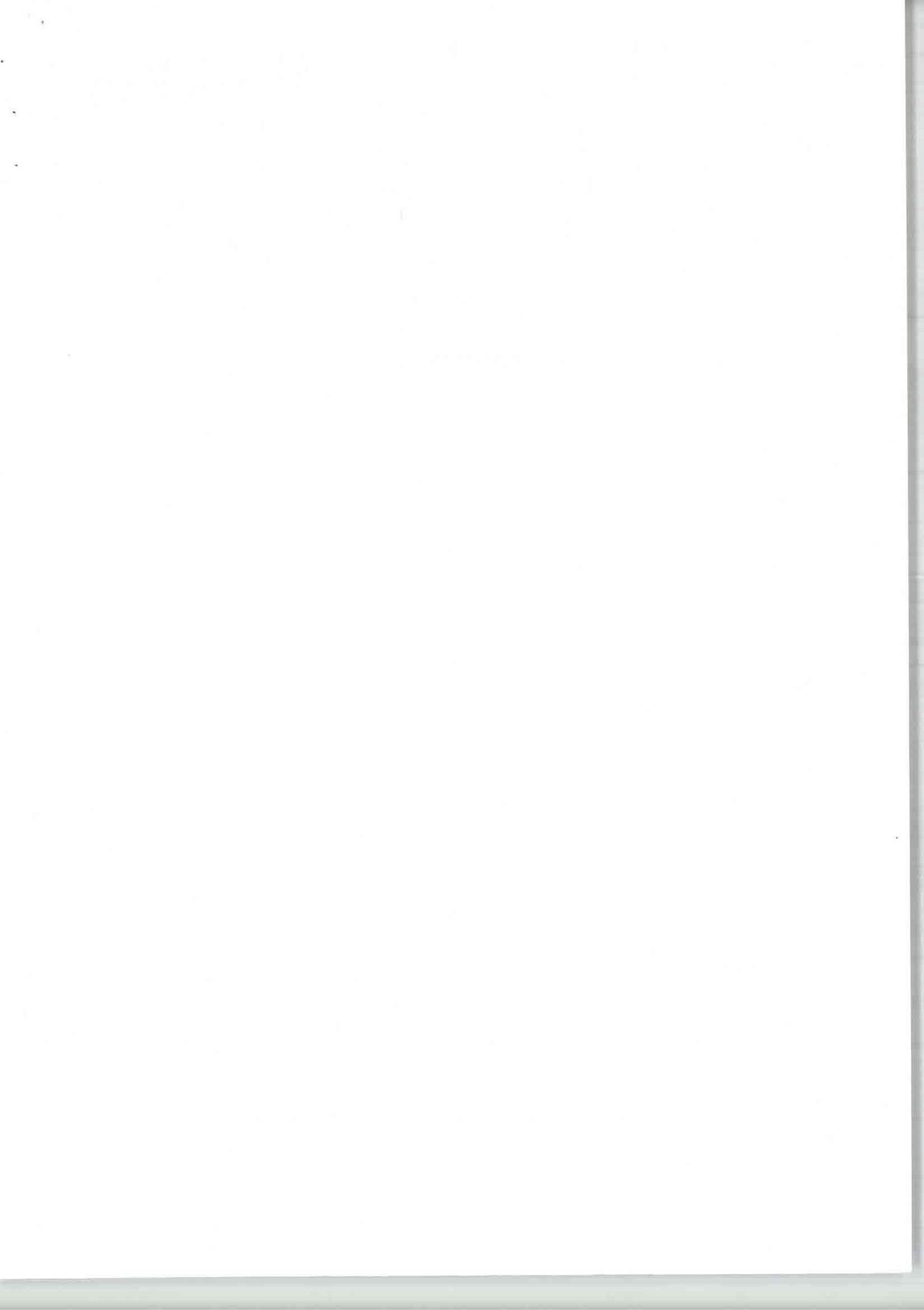
18/6

תאריך המבחן:	07.03.2022
שםות המרצים:	ד"ר אסף זריצקי
שםות המתרגלים:	מר ישעה צברי, גב' יעל מה-טוב, גב' שיר כהן
שם הקורס:	מבוא למדעי המחשב לשפת פיתון
מספר הקורס:	38211111, 37211111
שנה:	2022, מועד ב'
משך הבחינה:	שלוש שעות
חומר עזר:	דף A4 כתוב בכתב יד בשני הצדדים

**אני קראו היטב את ההוראות שלללו:**

- בבחן 3 שאלות הכוללות סעיף משנה. בבחן 103 נקודות. כדי לקבל את מלא הניקוד יש לענות נכון על כל השאלות. ניקוד כל סעיף מצוין לידי. אין בהכרח קשר בין ניקוד הסעיף ובין רמת הקושי שלו.
- מומלץ לקרוא כל שאלה עד סופה, על כל סעיפה, לפני תחילת הפתרון.
- את הפתרונות יש לכתוב במסגרות המסומנות לכל שאלה בטופס הבחינה. המחברת שקיבלתם היא מחברת טיוטה, והיא לא תימסר כלל לבדיקה. **בסוף הבחינה נאוסף אך ורק את דף התשובות.** כל שאר החומר יועבר לගירסה.
- בכל סעיף ניתן להשתמש בקוד שהתקשתם לכתוב בסעיפים הקודמים, גם אם לא פתרתם אותם.
- ניתן **להגיה שהקלט תקין**, אלא אם נכתב אחרת בשאלת.
- במידה ואיינכם יודיעים את התשובה לסעיף שלם כלשהו, רשמו "לא יודעת" (במקום תשובה) ותזכו ב- 20% מניקוד הסעיף. אם רשום "לא יודעת", ההתייחסות היא לכל הסעיף.
- אין להשתמש בחבילות או במודולים, אלא אם נאמר במפורש. כאשר אתם מתבקשים להשתמש בחבילה אין צורך לבצע import.

**בהצלחה!**

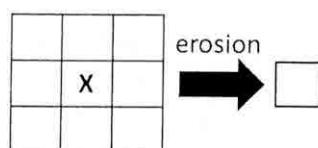
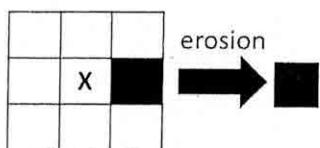


### שאלה 1 (30 נקודות)

הקשר ותזכורת: בשיעור ראותם דוגמה של הפעלת קונבולוציה (convolution) על תמונה עבור הפחתת רעשים באמצעות מיצוע ערכי האפור בסביבה של כל פיקסל (denoising with local means). הסביבה של כל פיקסל הייתה בגודל  $3 \times 3$  פיקסלים, כאשר הפיקסל האמצעי היה זה שעודכן בתמונה הפלט.

בשאלה זו תמשו קונבולוציה עבור "שחיקה" (erosion) של תמונה בינהרית.

מה המשמעות של "שחיקה"? כל פיקסל לבן בתמונה הפלט שבסביבתו קיים פיקסל שחור – יהפוך לפיקסל שחור בתמונה הפלט. רק פיקסליהם שהם וכל סביבתם לבנים בתמונה הפלט – ישארו לבנים בתמונה הפלט. לדוגמה, עבור סביבה בגודל  $3 \times 3$  פיקסלים, פיקסל לבן בתמונה הפלט יהיה לבן בתמונה הפלט רק אם כל שמונה הפיקסלים סביבה בפלט גם לבנים, אחרת הוא יהיה שחור.



ראו דוגמאות, שימוש לב שהתוצאה של השחיקה מוצגת ורק בפיקסל המרכז בסביבה (החישוב על הפיקסלים האחרים ערך בנפרד).

תמונה בינהרית תיוצר באמצעות רשיימה דו-מידית של ערכי פיקסלים (c-int) בגודל  $m \times n$  כאשר ערך כל פיקסל הוא 0 (שחור) או 1 (לבן). בשאלה זו נניח שהסביבה היא ריבועית. כך, בדוגמה העליונה, גודל הסביבה היה 3. על מנת שתתיה הגדרה חד משמעית של הפיקסל המרכזי, נניח כי גודל הסביבה תמיד יהיה אי-זוגי, לכל סעפי השאלה.

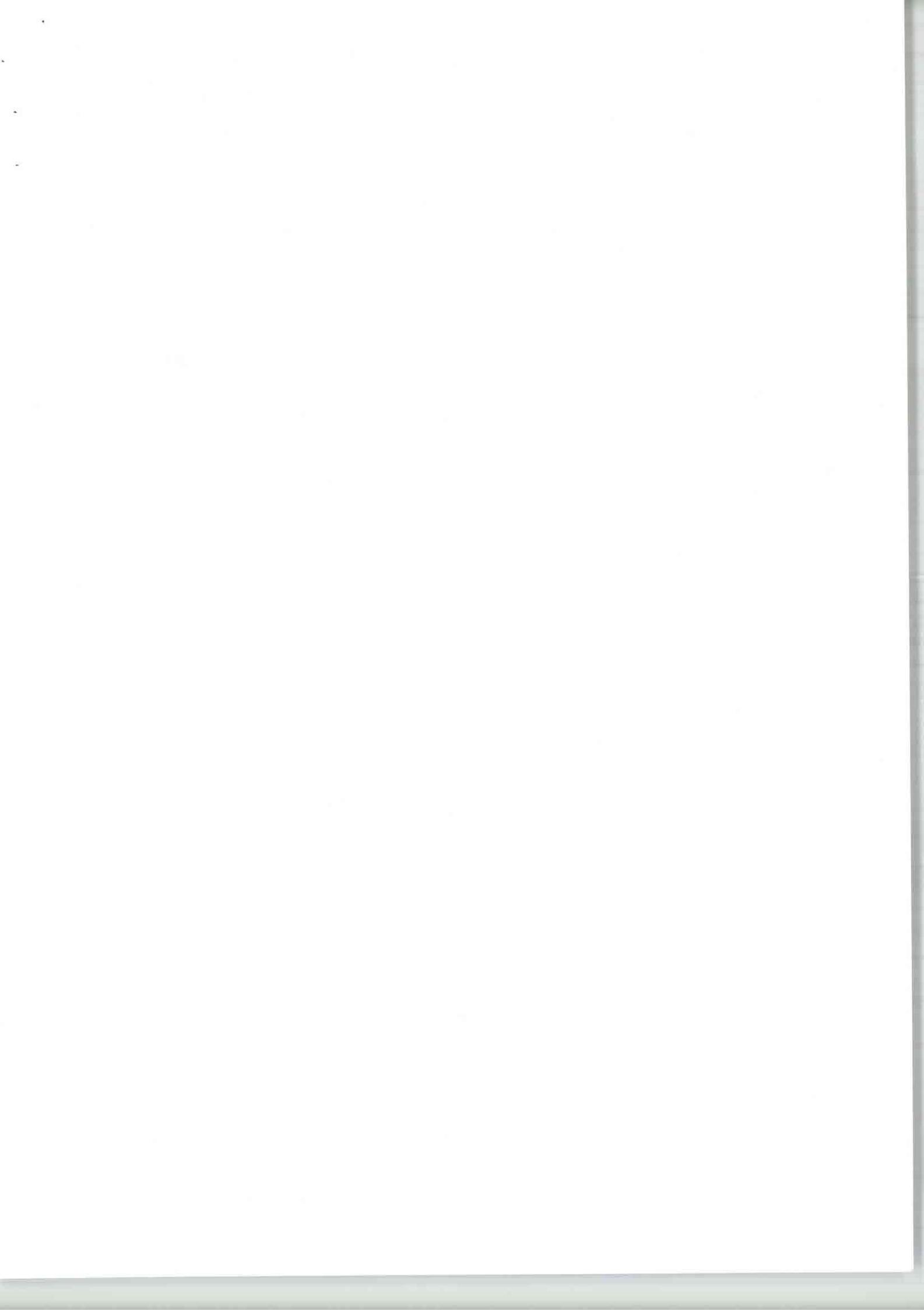
#### סעיף א' (4 נקודות)

משו את הפונקציה `erode` מקבלת כקלט רשיימה דו-מידית בגודל  $d \times d$  בשם `img_patch`, המייצגת סביבה של פיקסל (כלומר חלק מתמונה הפלט) ומחזירה את תוצאה השחיקה – ערך 0 או 1.

דוגמאות ריצה:

```
>>> patch = [[1, 1, 0],
              [1, 1, 1],
              [0, 0, 0]]
>>> erode(patch)
0
>>> patch = [[1, 1, 1],
              [1, 1, 1],
              [1, 1, 1]]
>>> erode(patch)
1
>>> patch = [[1]]
>>> erode(patch)
1
```

$$\begin{array}{l} A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} \\ 0 = \text{שחור} \end{array}$$



```
>>> patch = [[0]]
>>> erode(patch)
0
```

```
def erode(img_patch):
    count = 0
    for i in range(len(img_patch)):
        for j in range(len(img_patch[0])):
            if img[i][j] == 1:
                count += 1
            m1 = len(img_patch) // 2
            m2 = len(img_patch) // 2 + 1
            if count == len(img_patch) // 2 or count == len(img_patch):
                if len(img_patch) % 2 == 0:
                    if img_patch[m1][m1] == 1:
                        return 1
                else:
                    if img_patch[m2][m2] == 1:
                        return 1
    return 0
```

+3  
(1.1)

### חומר הבנה של השאלה

סעיף ב' (6 נקודות)

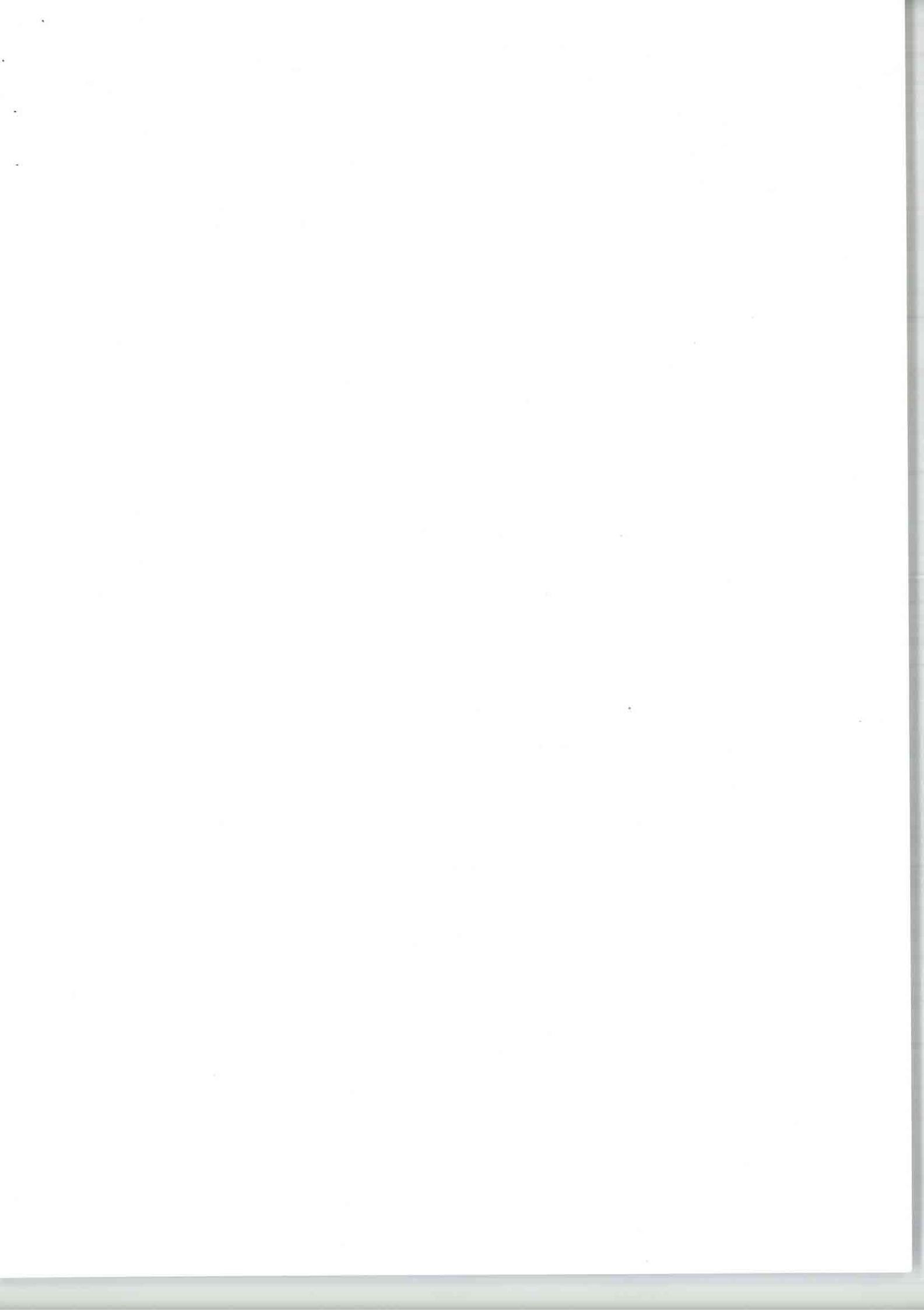
משוו את הפונקציה `image_erosion(image, patch_size)` אשר מקבלת כקלט שני ארגומנטים:  
 • תמונה בינהית המווצגת באמצעות רשימה דו-מימדית בגודל  $m \times n$ .  
 • גודל סביבה אי זוגי.

הfonkzia תחזיר גרסה "שחוקה" של תמונה הקלט לפי גודל הסביבה שהוגדר. שימוש לב, אין לשנות את תמונה הקלט. גודל תמונה הפלט תהיה זהה לחמונה הקלט. פיקסלים שנמצאים במסגרת התמונה ישארו ללא שינוי בתמונה הפלט.  
**תזכורת:** ערכו כל פיקסל בתמונה הפלט מחושבים ע"פ הסביבה המתאימה בתמונה הקלט.

### דוגמאות ריצה:

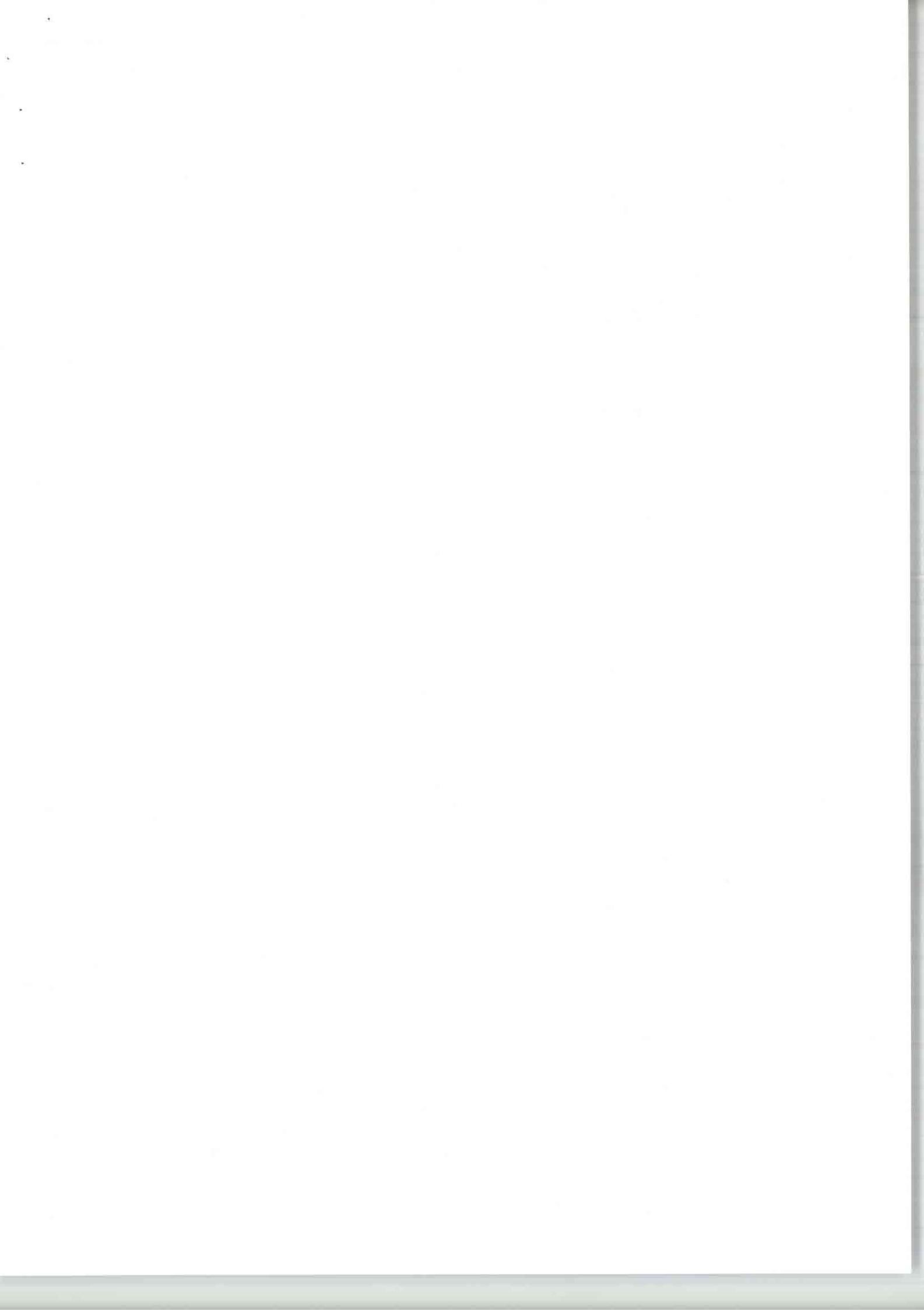
שימוש לב: הפיקסלים המודגשים הם היחידים עליהם הפעלה פועלת השחיקה. שאר הפיקסלים נמצאים על מסגרת התמונה.

```
>>> image = [[1, 1, 1, 1, 0],
             [1, 1, 1, 1, 1],
             [1, 1, 1, 0, 0]]
>>> image_erosion(image, 3)
[[1, 1, 1, 1, 0],
 [1, 1, 1, 1, 1],
 [1, 1, 1, 0, 0]]
```



```
>>> image_erosion(image, 1)
[[1, 1, 1, 1, 0],
 [1, 1, 0, 1, 1],
 [1, 1, 1, 0, 0]]
```

```
def image_erosion(image, patch_size):
    if patch_size == 1:
        return image
    for i in range(len(image) - len(image[0]) + 1)[1], len(image) // 2:
        for j in range(len(image[0]) - 1):
            if if
                +1
                (1,2)
                לען יודע
```



לסעיפים הבאים נתונה לכם פונקציה `evaluate(image)` אשר מקבלת כקלט תמונה בינהריה ומחזירה כפלט מספר שמהו זה ממד מספרי לתמונה. השימוש של `evaluate` אינו חושף בפניכם ואינו נחוץ לפתרון השאלה.

בכל הדוגמאות בהמשך השאלה תחזיר את מספר הפיקסלים הלבנים בתמונה, אך כמובן שעל הפתרון שלכם לתרום בכל שימוש אפשרי של הפונקציה.

```
>>> image = [[1, 1, 1, 1, 0],
   [1, 1, 1, 1, 1],
   [1, 1, 1, 0, 0]]
>>> evaluate (image)
```

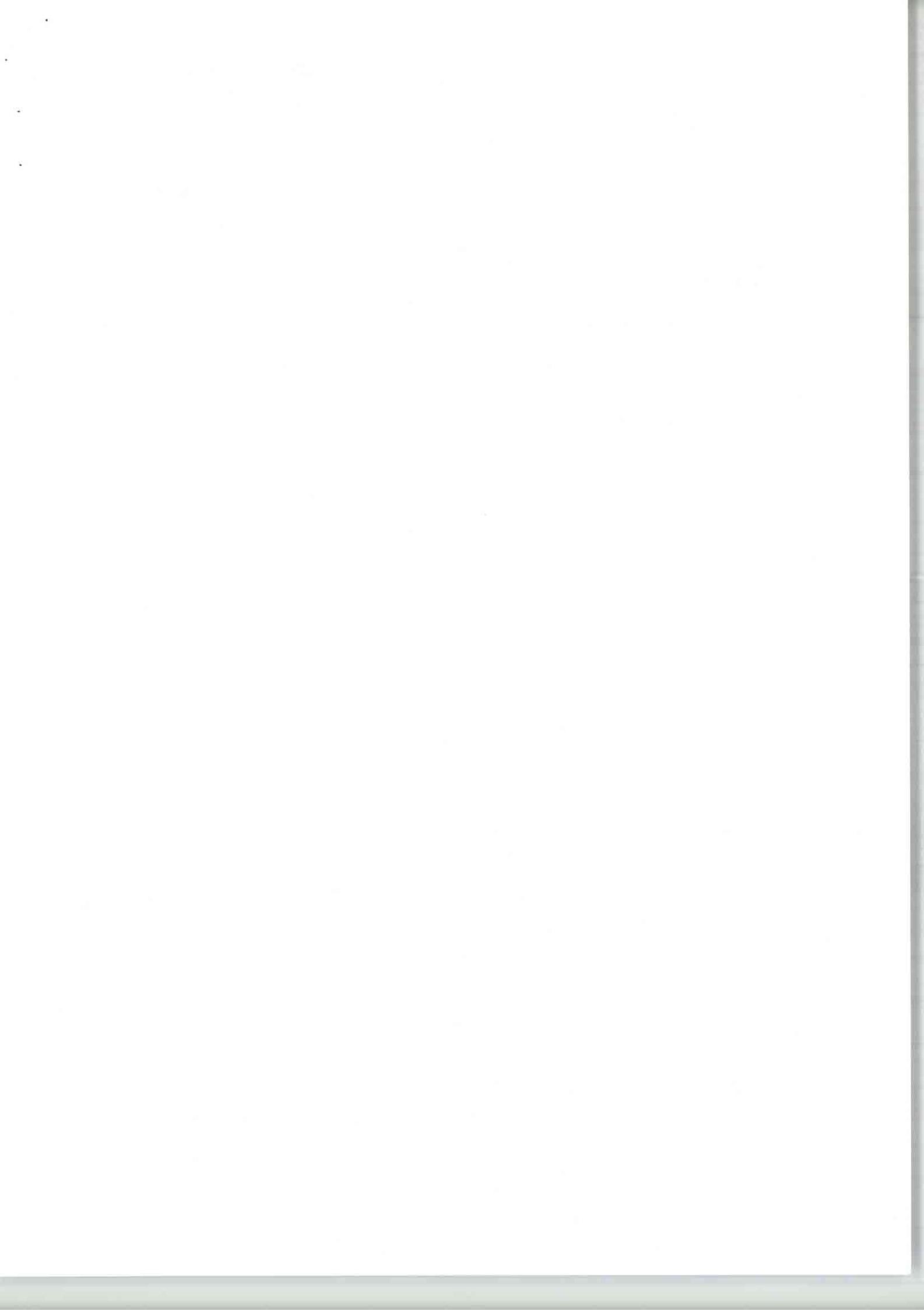
12

הוחלט לשמר את התמונות במבנה נתונים שישמור על סדר לפי הממד המספרי של כל תמונה. לשם כך נבחר מבנה הנתונים עץ חיפוש בינארי (Binary search tree). כל קדוק בעץ יכול את הממד המספרי של התמונה כמפתח (key) ואת התמונה עצמה כערך (image).

לרשומכם מימוש חלקי של המחלקה `ImageBST`. שימו לב, שבשונה מהשימוש שראיתם בכיתה, מחלוקת העץ מחזיקה את הקדוק עצמו ולא קיימת מחלוקת נוספת שמחזיקה את שורש העץ.

```
class ImageBST:
    def __init__(self, image):
        self.left = None
        self.right = None
        self.image = image → ערך
        self.score = evaluate(image) → יוזה

    def __repr__(self):
        הנីមួនដែលបានបង្កើត។ ការបង្ហាញនេះនឹងបង្ហាញថា គេបានបង្កើតចំណាំទូទៅ។
        ត្រូវបានបង្ហាញថា គេបានបង្កើតចំណាំទូទៅ។
```

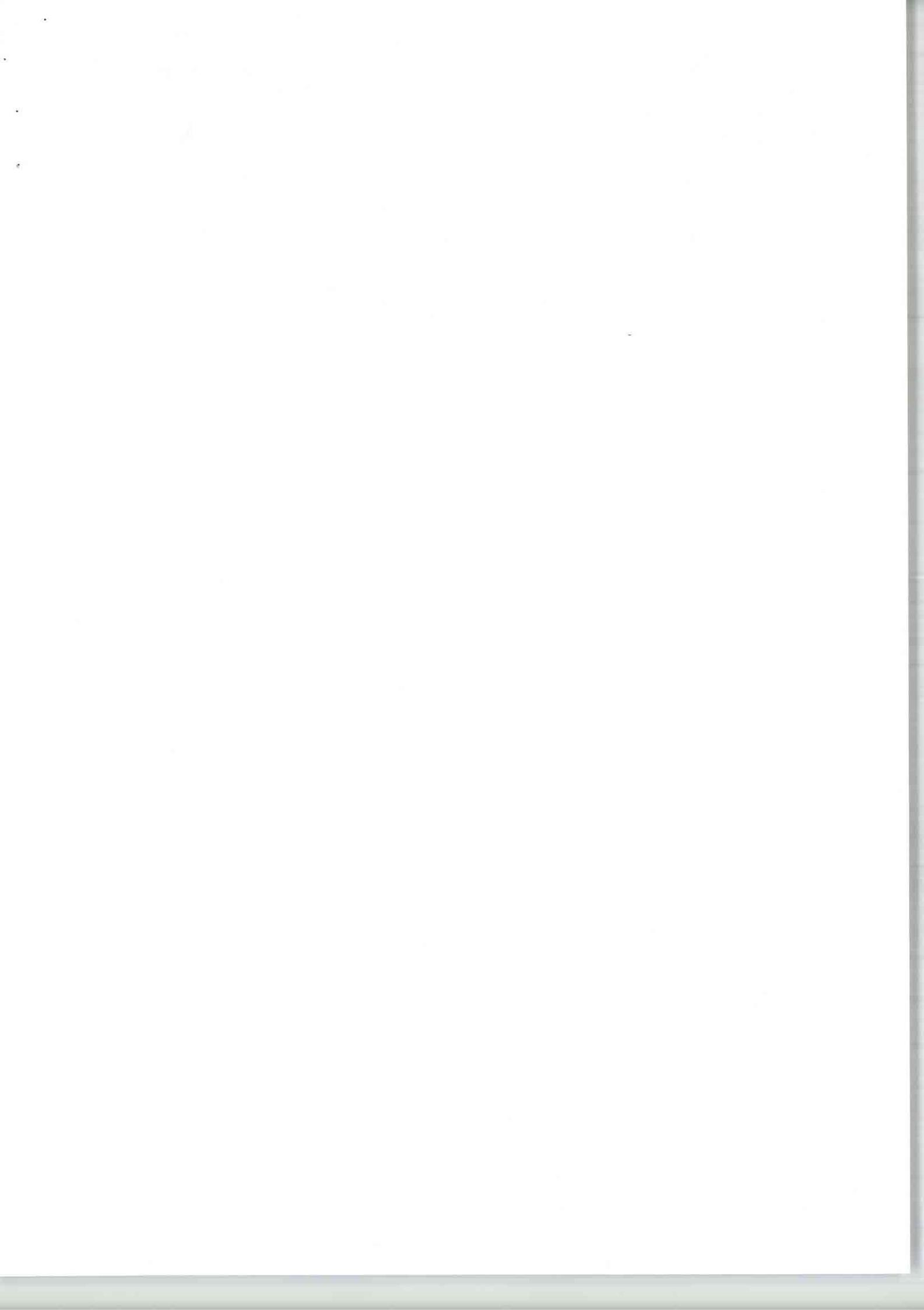


סעיף ג' (6) נקודות

השלימו במחלקה ImageBST את השיטה insert(self, image) אשר מקבלת כקלט תמונה ביבנארית ומוסיפה אותה לעץ חיפוש הביבנארי. הניחו כי לא יאוחסנו בעץ שתי תמונות עם מדד מספרי זהה.

דוגמאות ריצה:

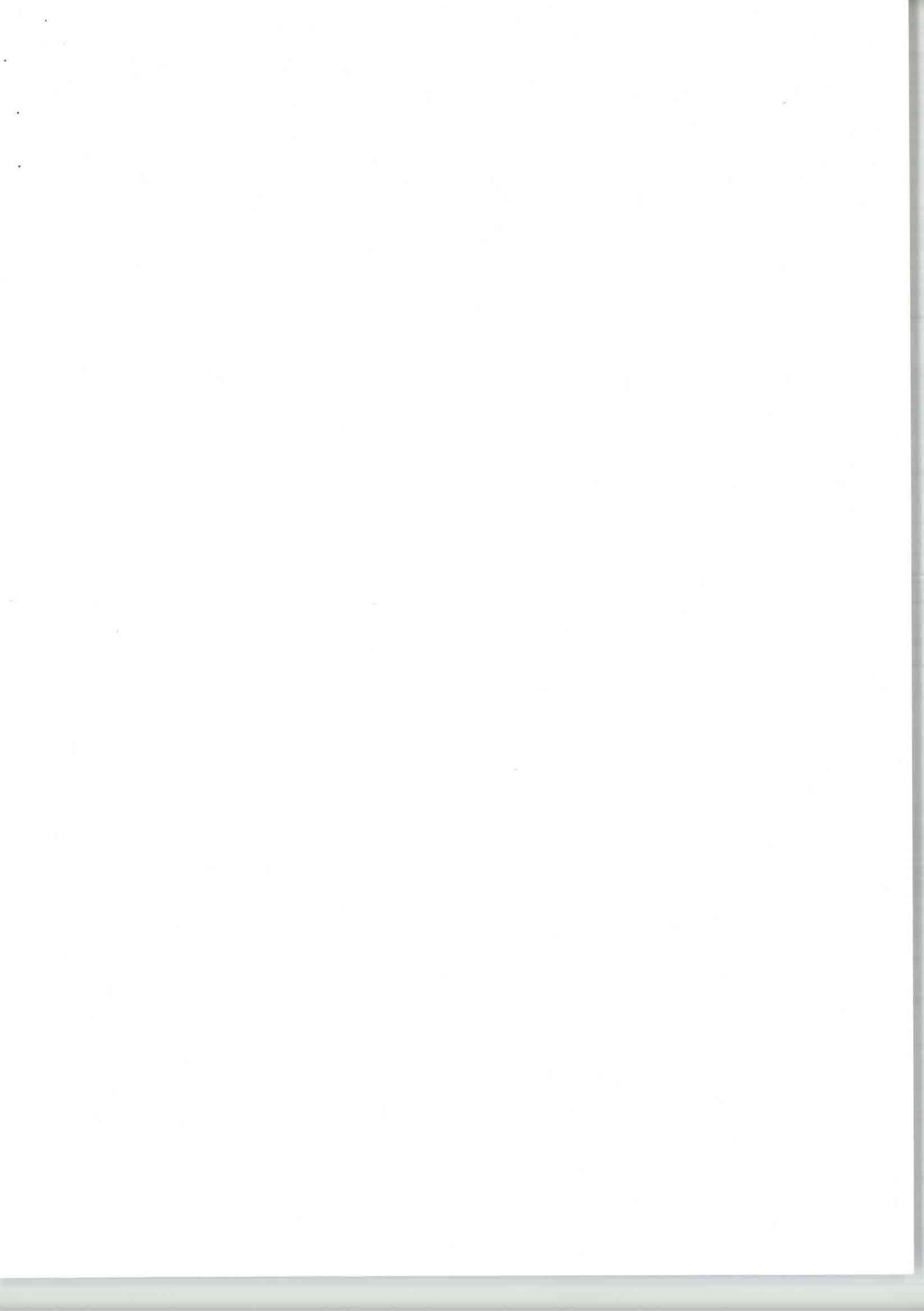
```
>>> image_1 = [[0, 0, 0],  
                 [0, 0, 0],  
                 [1, 0, 1]]  
>>> image_2 = [[1, 1, 1],  
                 [1, 1, 1],  
                 [1, 1, 1]]  
>>> image_3 = [[0, 0, 1],  
                 [0, 0, 1],  
                 [1, 0, 1]]  
>>> evaluate(image_1)  
2  
>>> node_1 = ImageBST(image_1)  
>>> node_1  
2  
/\  
# #  
>>> node_1.insert(image_2)  
>>> node_1  
2  
/\_  
# 9  
/\  
# #  
>>> node_1.insert(image_3)  
>>> node_1  
2  
/\_____  
# 9  
__/\  
4 #  
/\  
# #
```





```
def insert(self, image):
    if self.left == None and self.right == None:
        self.score = image.Score
        self.image = image
    if self.Score == image.Score:
        return +6
    (1.3)
    if self.Score < image.Score:
        if self.left is None:
            self.left = ImgBst(image)
        else:
            return insert(self.left, image)
    if self.Score > image.Score:
        if self.right is None:
            self.right = ImgBst(image)
        else:
            return insert(self.right, image)
    return
```

כל הבוד! BA





סעיף ד' (4 נקודות)

השלימו במחלקה ImageBST את השיטה get\_max(self) אשר מוחזירה את התמונה בעלת המדר המספרי המקסימלי.

**שימוש לב** - על הפתרון להיותiesel. מעבר על כל קודקי העץ לא יזכה בניקוד כלל (אלא במקרים של עץ מנוקב כאשר לכל קודקן יש בדיקת אחד).

דוגמאות ריצה:

דוגמת הריצה תואמת לעץ שהוצג בסעיף ג'.

```
>>> node_1.get_max()
[[1, 1, 1],
 [1, 1, 1],
 [1, 1, 1]]
```

```
def get_max(self):
    if self.left is None and self.right is None:
        return None
    image = self.right
    while right != None:
        image = image.right
    +4
    (1.4)
    return image.score
```

**כל הכבוד! BA**

סעיף ח' (10 נקודות)

בסעיף זה תמשו חישוב תמונה מבנה נתונים לפי דמיון. דמיון בין זוג תונות יוגדר לפי ההפרש בערך מוחלט בין המדר המספרי (evaluate) של שתי התמונה. ככל הערך הדמיון קטן יותר, כך שתתי התמונה דומות יותר. למשל, בדוגמה שבסעיף ג' הערך המוחלט של ההפרש בין image\_1 ל-image\_3 הוא 2, ולכן הדמיון הינו 2. הדמיון בין image\_2 ל-image\_1 הוא 7. לעומת זאת, הדמיון בין image\_1 ל-image\_2 הוא 1.





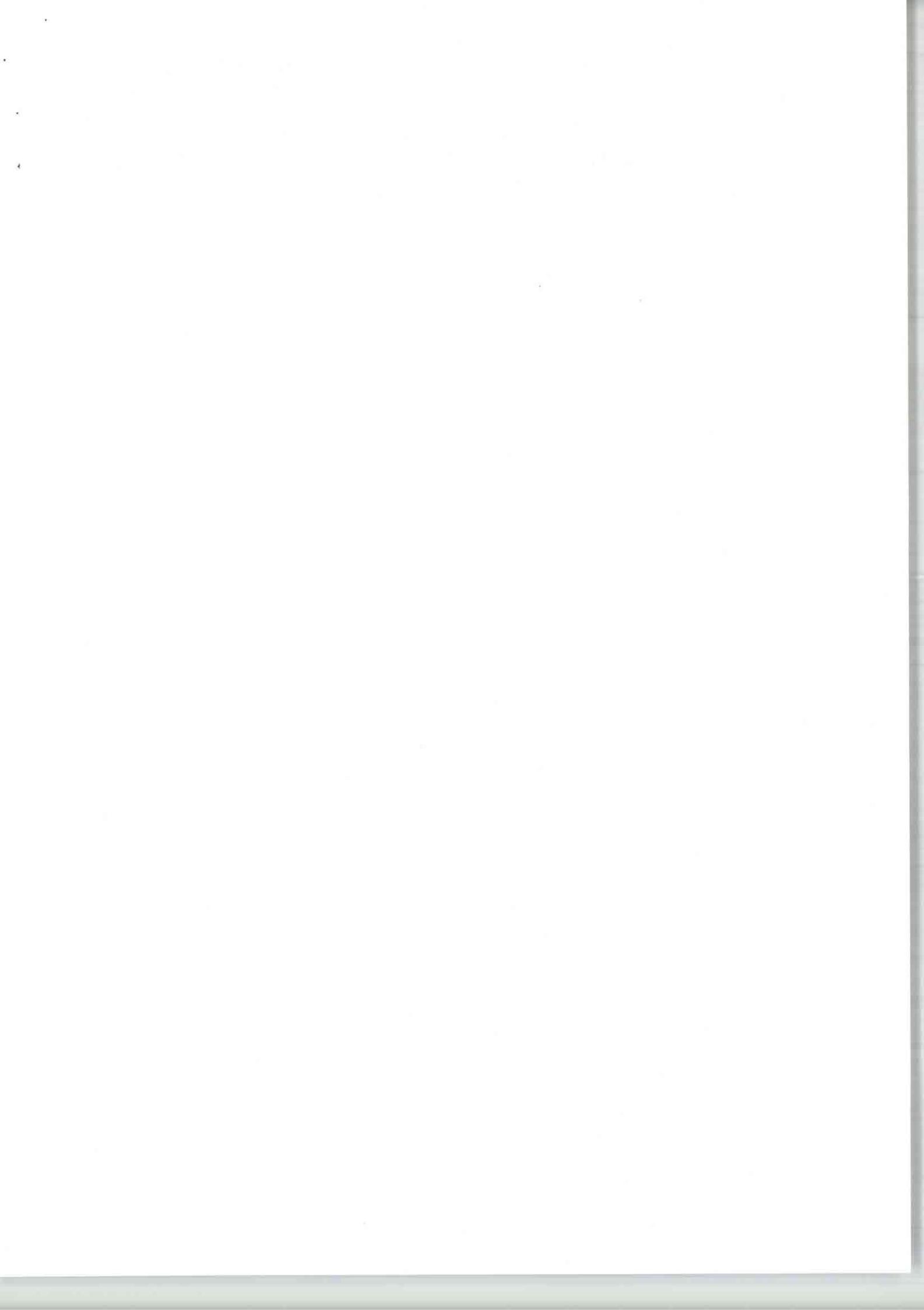
השלימו במחלקה ImageBST את השיטה `find_most_similar(self, img)` אשר מקבלת כקלט תמונה ביןארית `(img)` ומחזירה את התמונה הדומה ביותר לתמונה הקלט. במקרה שיש בעץ יותר מתמונה אחת דומה ביותר, והחזרו אותה מהן. **שים לב** - על הפתרון להיות יעיל. מעבר על כל קודקודי העץ לא יזכה בניקוד כלל (אלא במקרים של עץ מנון כאשר לכל קודקו יש בדוק בן אחד).

**רמז:** השתמשו בערך אינסוף `float('inf')` כדי להתחיל את החישוב.

דוגמאות ריצה: בהתאם לעץ שהוצע בסעיף ג'.

```
>>> node_1
2
/ \
#   9
  / \
4   #
 / \
#  #

>>> image_4 = [[0, 0, 1],
               [0, 0, 0],
               [1, 0, 0]]
>>> evaluate(image_4)
2
>>> node_1.find_similar(image_4)
[[0, 0, 0],
 [0, 0, 0],
 [1, 0, 1]]
# Represents image with evaluation of 2
>>> image_5 = [[1, 1, 1],
               [0, 1, 1],
               [1, 0, 0]]
>>> evaluate(image_5)
6
>>> node_1.find_similar(image_5)
[[0, 0, 1],
 [0, 0, 1],
 [1, 0, 1]]
# Represents image with evaluation of 4
```



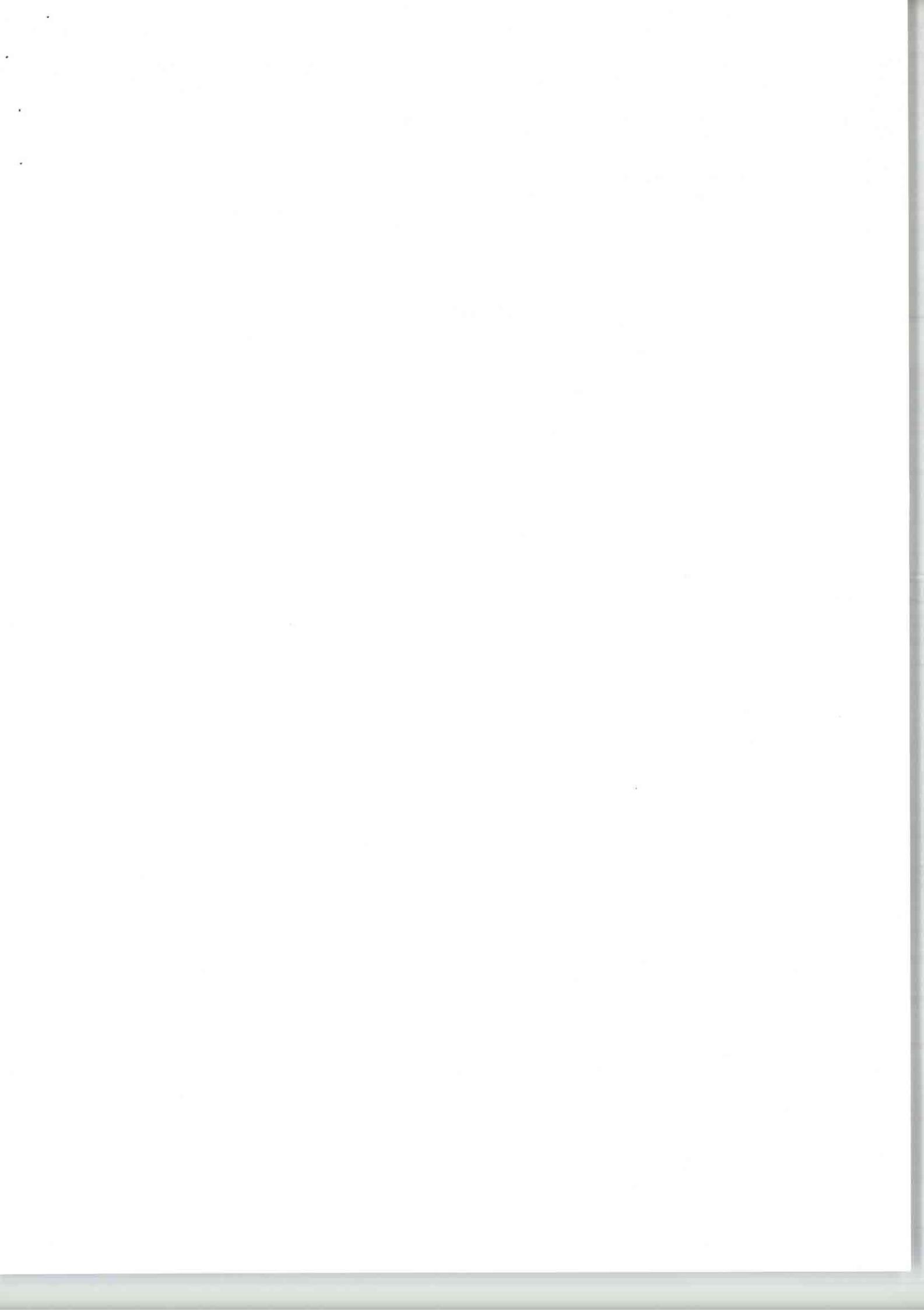
```
def find_similar(self, img):
```

רשות הדקל

+2

(1.5)

לא ידוע





## שאלה 2 (42 נקודות)

בחברת WhatsGram מפתחים מערכת לניהול הודעות טקסט. בשאלת זאת, תමמשו מבני נתונים שייחסנו ויציגו הודעות בהתאם לדרישות החברה.

### סעיף א' (5 נקודות)

משו את המחלקה WGM (קיצור של WhatsGramMessage) אשר מייצגת הודעה בודדת. לכל הודעה יש תוכן (content), שם השדה, ועדיפות (priority). מספר שלם לא שלילי. ככל שהערך גבוה יותר כך העדיפות גבוהה יותר, שם השדה: .(priority).

על המחלקה לתרום בשיטות הבאות:

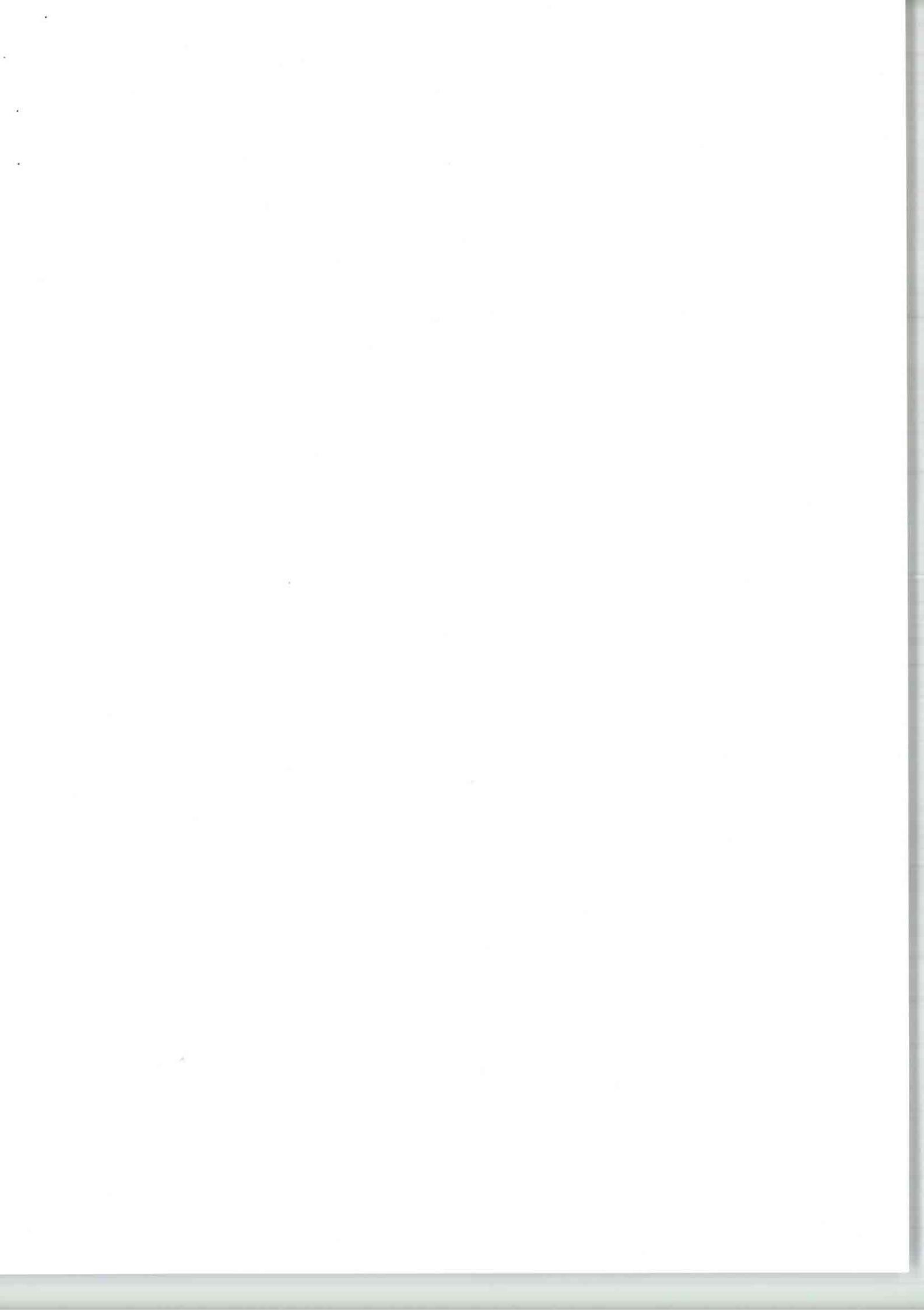
- \_\_init\_\_(self, cont, prior) – בניאי. מקבל את תוכן הודעה כמחרוזת (cont) ואת עדיפות ההודעה כמספר שלם לא שלילי (prior).
- שיטה לתמיכה בששת פעולות השוואת. ההשוואה בין אובייקטים מטיפוס WGM מוגדרת לפי שדה העדיפות שלהם (priority).
- השיטה \_\_repr\_\_(self) שמחזירה את תוכן הודעה כמחרוזת.

**ניקוד מלא יינתן עבור שימוש עם מנגנום קוד (ניתן להשתמש בחבילת functools).**

### דוגמת ריצה:

```
>>> m1 = WGM('Hi!', 0)
>>> print(m1)
"Hi!"  

>>> m2 = WGM('BRB', 1)
>>> m3 = WGM('ASAP', 2)
>>> m4 = WGM('IDK', 2)
>>> m2 > m1
True
>>> m2 > m4
False
>>> m3 >= m4
True
```



class WGM:

```
def __init__(self, cont, prior):
```

```
    self.cont = cont
```

```
    self.prior = prior
```

**לא ככה משתמשים total\_ordering**

```
def __eq__(self, other):
```

```
    return self.prior == other.prior
```

```
def __gt__(self, other):
```

```
    if self.prior > other.prior:
```

**אם הוא שווה? None**

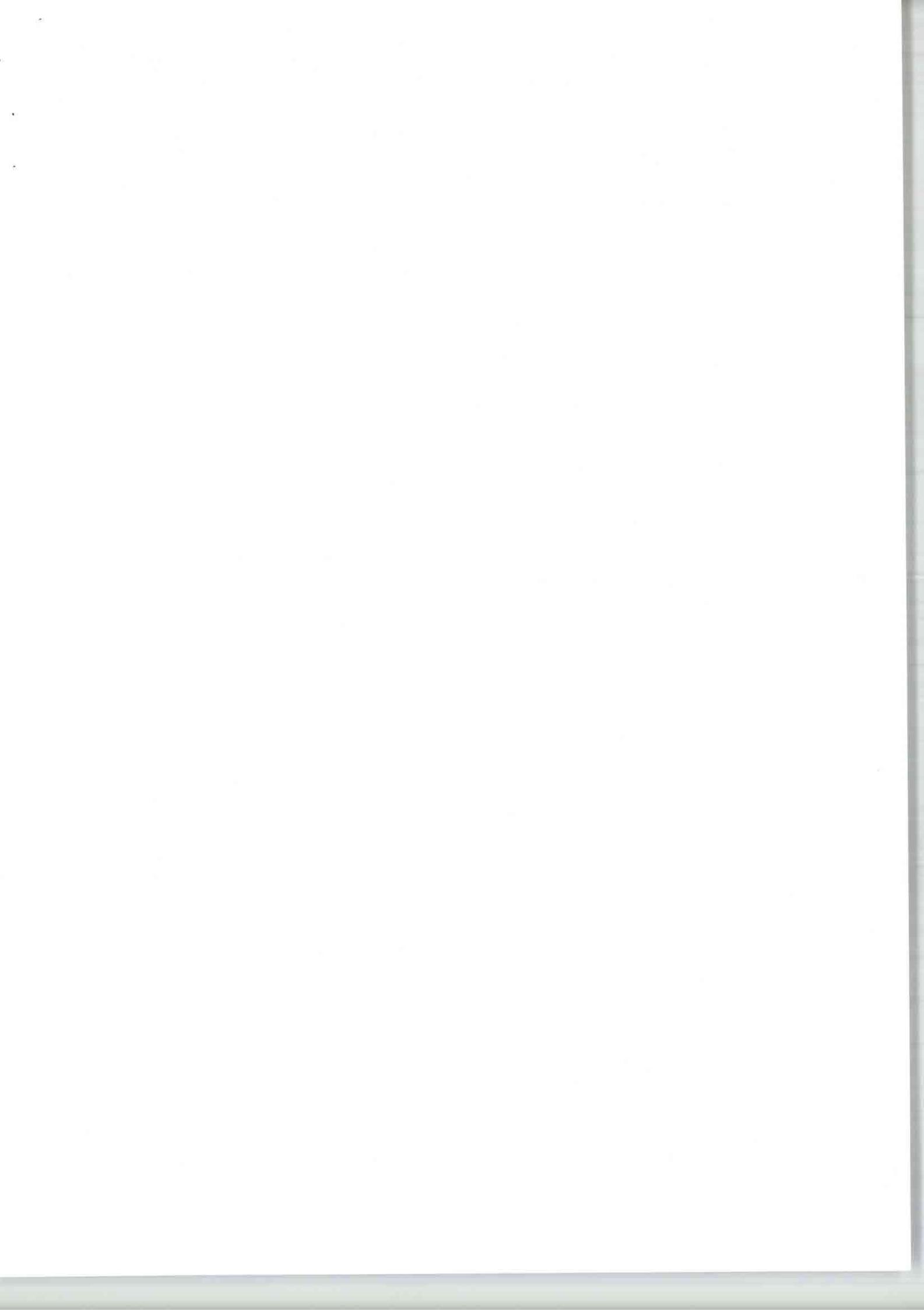
```
    if self.prior < other.prior:
```

```
        return False
```

```
def __repr__(self):
```

```
    return str(self.cont)
```

4  
(2.1)





חברת WhatsGram פיתחה מבנה נתונים בסיסי יעיל אשר זמין לרשותכם שנקרא **WGS** (קיצור של מבנה הנ吐נים מוגדר עם הממשק (API) **curlkitן**: (WhatsGramStorage

- - בנאי. מתחילה מבנה ריק. **\_\_init\_\_(self)**
- - מחזירה את מספר האיברים במבנה **C-int**. **\_\_len\_\_(self)**
- - מוסיפה איבר (val) מטיפוס object בזנב המבנה. **insert\_at\_beginning(self, val)**
- - מוסיפה איבר (val) מטיפוס object בזנב המבנה. **insert\_at\_end(self, val)**
- - מחזירה עותק של האיבר בראש המבנה. אם מספר האיברים במבנה הוא 0, מחזירה **None**. **get\_head(self)**
- - מחזירה עותק של האיבר בזנב המבנה. אם מספר האיברים במבנה הוא 0, מחזירה **None**. **get\_tail(self)**
- - מסירה את האיבר בראש המבנה. האיבר שייהי בראש המבנה לאחר ההסרה יהיה השני במבנה לפני ההסרה. **remove\_head(self)**
- - מסירה את האיבר בזנב המבנה. האיבר שייהי בזנב המבנה לאחר ההסרה יהיה האחד לפני האיבר בזנב המבנה לפני ההסרה. **remove\_tail(self)**
- -מחזירה איטרטור (iterator) על האיברים במבנה. **\_\_iter\_\_(self)**
- -מחזירה עותק عمוק (deep copy) של האיבר הבא. זורקת שגיאה מסווג כאשר אין יותר אובייקטים להחזיר. **\_\_next\_\_(self)**

על מנת לשמר על יעילות ואחדות בסיס הקוד בחברה, בימוש הסעיפים הבאים (**ב' ו-ג'**) אתם רשאים להשתמש אך ורק במוועע אחד של WGS ומופעים WGM, ולהימנע מהעתקה عمוקה באמצעות השיטה deepcopy ~~~~~ מותרת לשימוש(copy.copy).

#### סעיף ב' (12 גקודות)

בסעיף זה תממשו את WGMDS (WhatsGramMessagesDataStructure), מבנה נתונים המאחסן ומציג הודעות (מטיפוס WGM) לפי סדר קבלת ההודעות ועדיפות.

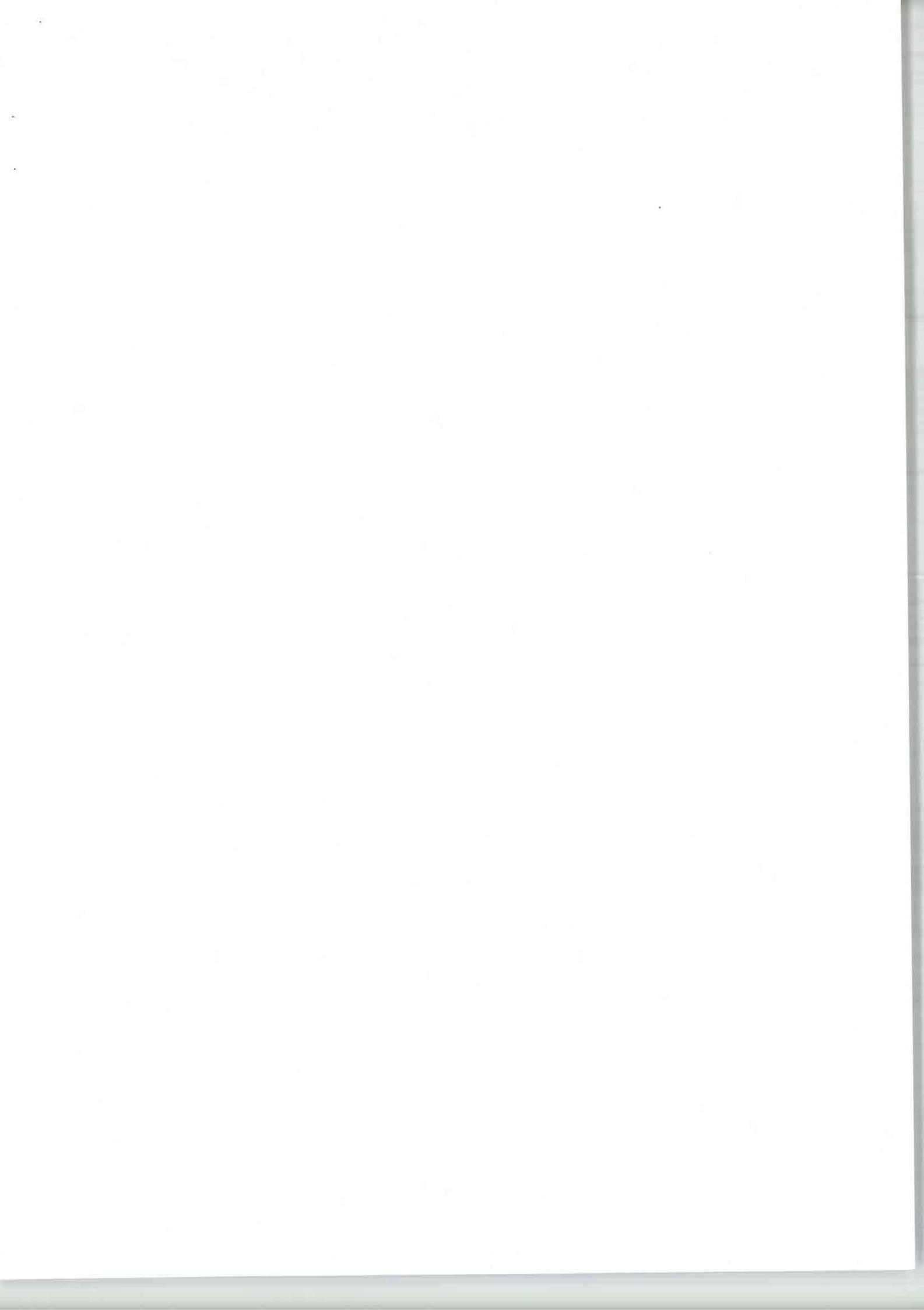
על המחלקה לתרום בשיטות הבאות:

- בנאי. איןו מקבל קלט. **\_\_init\_\_(self)**
- מוסיפה הודעה חדשה (msg, msg, מטיפוס WGM) למבנה הנתונים. **add\_msg(self, msg)**
- מחזירה את תוכן ההודעה בעלת העדיפות הגבוהה ביותר. אם קיימות כמה הודעות כללה (בעלות עדיפות זהה), תוחזר ההודעה האחורה שהתקבלה מבניהן. **השיטה מסירה את ההודעה מבנה הנתונים.** **pop\_msg(self)**

שימוש לב שלנוחיותכם הקצנו משכיצת נפרדת לכל שיטה.

#### דוגמת ריצה:

```
>>> mm_dt = WGMDS()
>>> mm_dt.add_msg(WGM('HA!', 0))
>>> mm_dt.add_msg(WGM('HAME', 2))
```



```
>>> mm_dt.add_msg(WGM('Ready?', 3))
>>> mm_dt.add_msg(WGM('KAME', 2))
>>> mm_dt.pop_msg()
Ready?

>>> mm_dt.pop_msg()
KAME

>>> mm_dt.pop_msg()
HAME

>>> mm_dt.pop_msg()
HA!
```

```
class WGMDs:
    def __init__(self):
        pass
```

✗



```
def add_msg(self, msg):  
    new = WGM(msg)  
    if len(self) == 0:  
        self.insert_at_beginning(new)  
    else:  
        self.insert_at_end(new)
```

למה יכולה **self** להיות מוגדרת?



```

def pop_msg(self):
    if len(self) == 0:
        return None
    max = self.head.next
    curr = self.get_head()
    while curr.next != None and max > curr:
        curr = curr.next
    else:
        max = curr
    my_list = []
    curr = self.head
    while curr.next != max:
        my_list.append(curr)
        curr.remove()
    curr.remove()
    for i in range(-1, len(my_list)):
        my_list.insert(0)

```

מה זה `??(msg curr.next)`

4  
(2.2)

פתרונות לא ברור בכלל.





### סעיף ג' (8 נקודות)

החברה החליטה כי אין צורך בתייעזר הודעות לפי שדה priority הקיימים במופע מטיפוס WGM. לכן, מפתחי החברה החליטו לפתח מבנה נתונים נוסף המאחסן ומציג הודעות (מטיפוס WGM) לפי סדר קבלת ההודעה, ולא תלוות בעדיפות. בסעיף זה תממשו את המחלקה WGMDS\_ByTime לפי הדרישה.

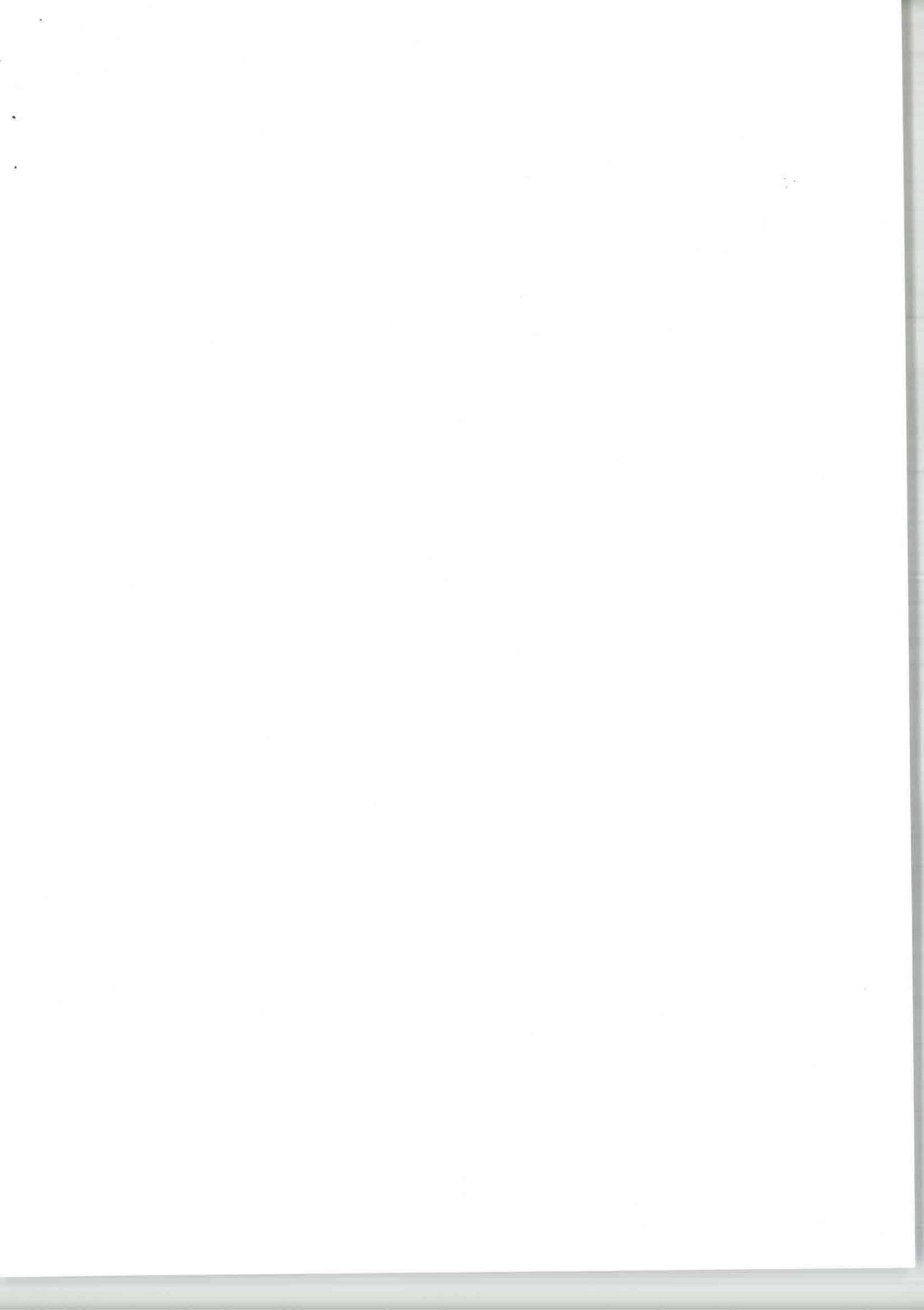
על המחלקה לחזור בשיטות הבאות:

- – \_\_init\_\_(self) – בניאי. אינו מקבל קלט.
- – \_\_add\_msg(self, msg) – מוסיף הודעה חדשה (msg, מטיפוס WGM) למבנה הנתונים.
- – \_\_pop\_msg(self) – מחזירה את תוכן ההודעה המוקדמת ביותר שהתקבלה. השיטה מסירה את ההודעה מבניה הנתונים.

שימוש לב, ניקוד מלא יינתן עבור שימוש קצר עם מינימום שכפול קוד. פתרון שיכלול קוד מיותר יזכה לכל היותר בחצי מניקוד הסעיף. רמז: אין צורך להשתמש בלולאות.

דוגמת ריצה:

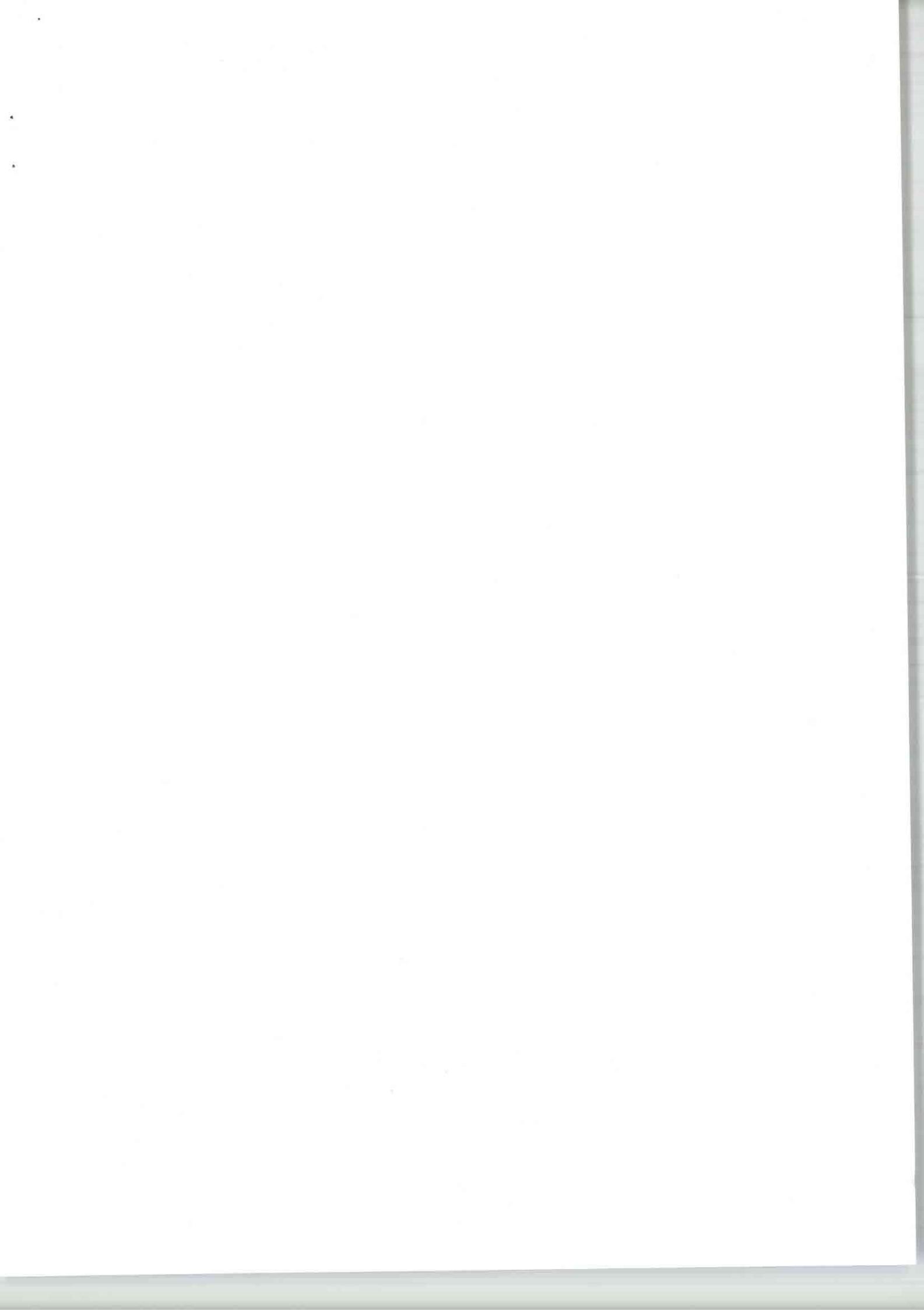
```
>>> mm_dt = WGMDS()
>>> mm_dt.add_msg(WGM('HA!', 0))
>>> mm_dt.add_msg(WGM('HAME', 2))
>>> mm_dt.add_msg(WGM('Ready?', 3))
>>> mm_dt.add_msg(WGM('KAME', 2))
>>> mm_dt.pop_msg()
HA!
>>> mm_dt.pop_msg()
HAME
>>> mm_dt.pop_msg()
Ready?
>>> mm_dt.pop_msg()
KAME
```



```
class WGMDS - By Time:  
    def __init__(self):  
        self.messages = []  
  
    def add_msg(self, msg):  
        self.insert_at_end(msg)  
  
    def pop_msg(self):  
        item = self.get_tail()  
        self.remove_tail()  
        return item
```

6  
(2.3)

הפתרון הכל חסכוני היה מנצח ירושה מהמבנה הקודם שמיימשתם.



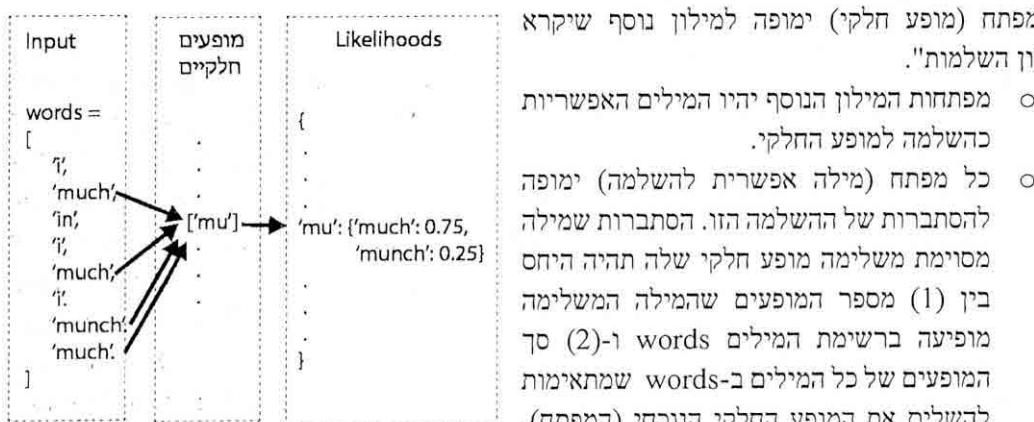


בתוכנות מסרים רבות קיים מגנון שלמת מילים אוטומטי. בסעיפים הבאים (ד'ה') תמצאו מגנון זה.

#### סעיף ד' (10 נקודות)

נדיר מופע חלקו של מילה כתת מחרוזת של אותה מילה שאורכה גדול מ-0 ותחילה בטו הראשון של המילה (כולל המילה עצמה). לדוגמה, להלן המופיעים החלקיים למילה "munch": "m", "mu", "mun", "munch". ניתן לראות שקיימות מחרוזות מופיעים החלקיים למספר מילים. לדוגמה, המחרוזות "mu" היא מופע חלקו של המילים "munch", "much", "murder" וככל מילה נוספת שפותחת ב- "mu". ממשו את הפונקציה calc\_subwords\_likelihoods(words) מקבלת רשימה של מילים (words), עם חזרות כולם מילה עשויה להופיע יותר מפעם אחת, ומוחזירה מילון לפי ההנחות הבאות:

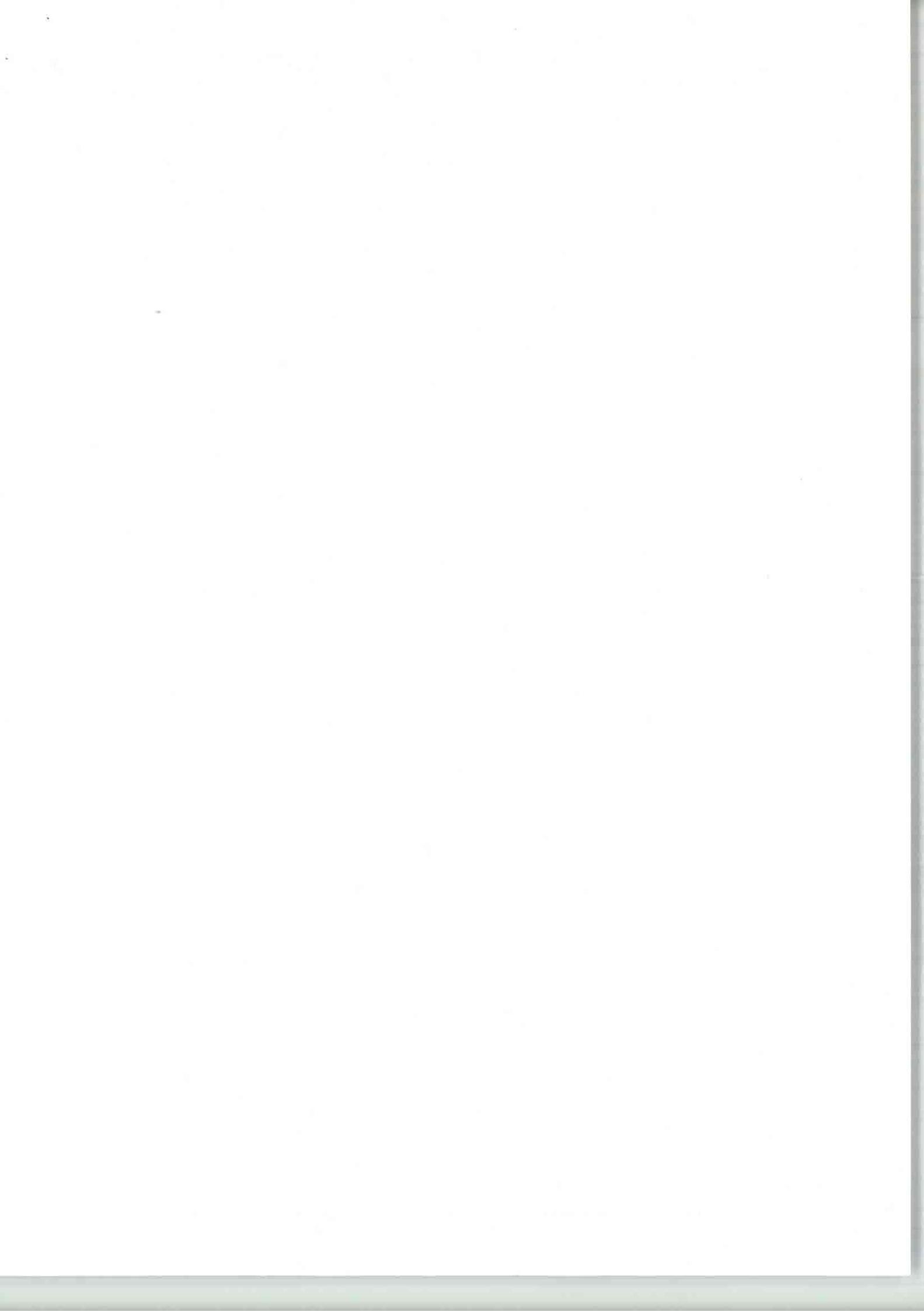
- כל המופיעים החלקיים של כל המילים ברשימה יהיו מפתחות במילון.



לדוגמה, עבור המופיע החלקי "mu" ורשימת המילים ["munch", "much", "much", "much"] המפתח "munch" ימושם 0.25 מושם ש- "munch" מופיע פעם אחת ברשימה המילים מתוך ארבע מופעים מילים אפשריות להשלמה. (ראו אויר ודוגמת ריצה).

#### דוגמת ריצה:

```
>>> words ['i', 'much', 'in', 'i', 'much', 'i', 'munch', 'much']
>>> calc_subwords_likelihoods(words)
{'i': {'i': 0.75, 'in': 0.25},
'm': {'much': 0.75, 'munch': 0.25},
'mu': {'much': 0.75, 'munch': 0.25},
'muc': {'much': 1.0},
'much': {'much': 1.0},
'in': {'in': 1.0},
'mun': {'munch': 1.0},
'munc': {'munch': 1.0},
'munch': {'munch': 1.0}}
```

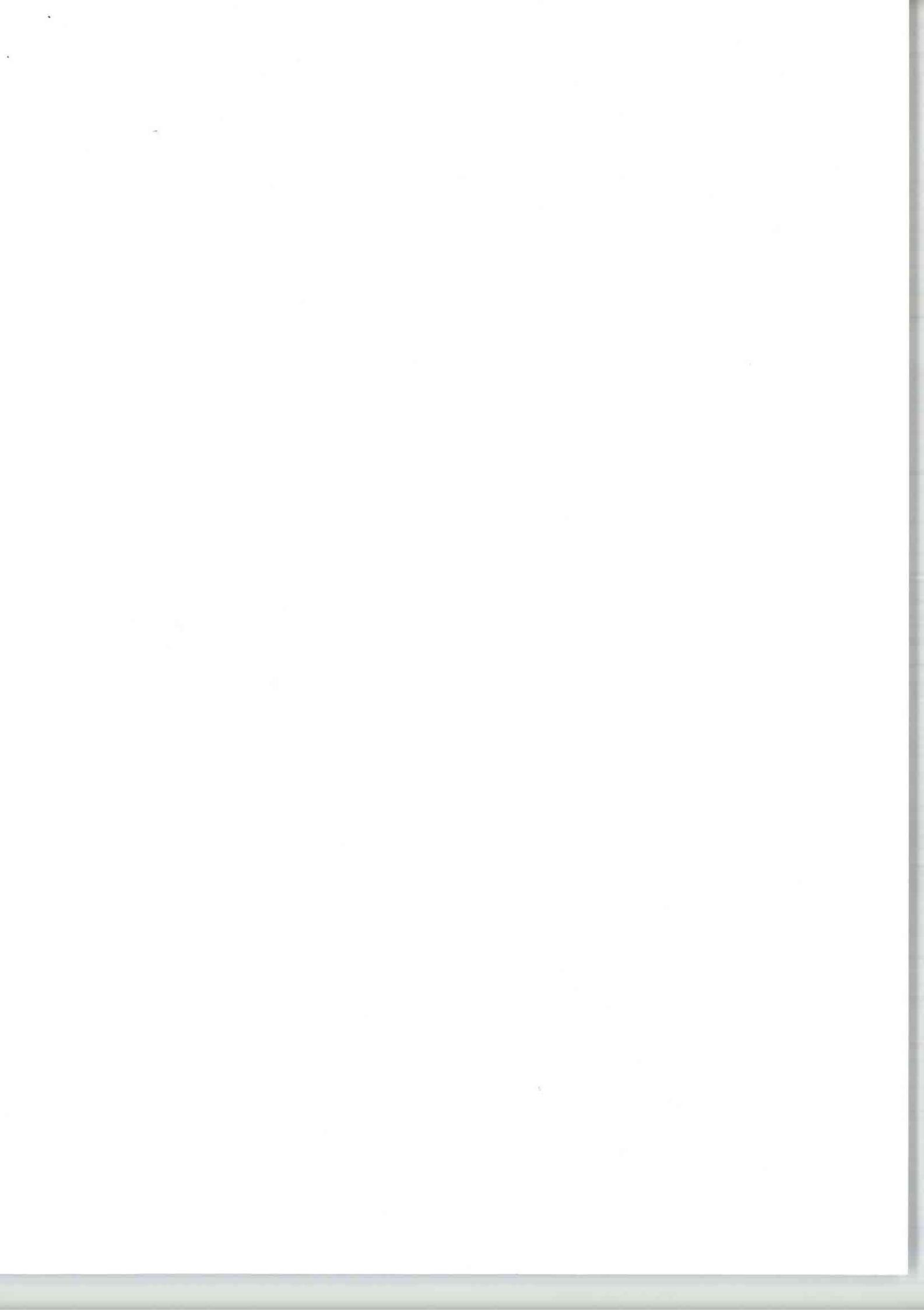




```
def calc_subwords_likelihoods(words):
```

לען וריאנט

2  
(2.4)



(סעיף ה' 7 נקודות)

בסעיף זה תמשכו את הפונקציה `complete_subwords(words_to_comp, tokens_freq)` המקבלת רשימה של מופעים חלקים של (`words_to_comp`) ומילון שלמה כפי שחושב בסעיף הקודם (`tokens_freq`). הפונקציה מחזירה מחרוזת בה כל מופיע חלקו הוחלף בהשלמה בעלת ההסתברות הגבוהה ביותר עבורה.

דוגמאות:

- ההשלמה בעלת ההסתברות הגבוהה ביותר למופיע חלקו לא מוכר (לא קיים מפתח ב-`tokens_freq`) היא המופיע החלקי עצמו.
- בהינתן שתי שלמות (או יותר) בעלות הסתברות זהה למופיע חלקו, כל אחת מההשלמות תתקבל.

דוגמת ריצה:

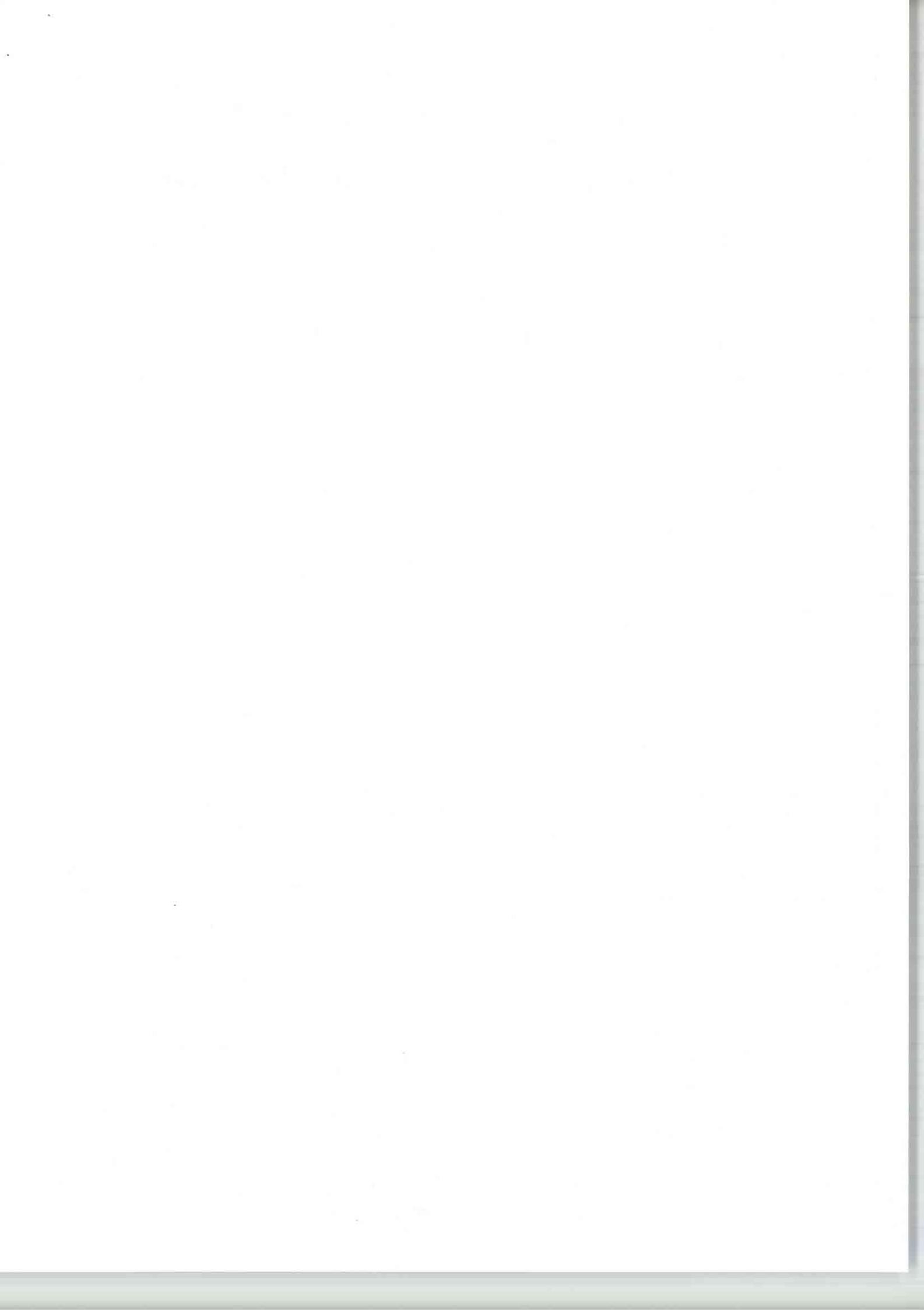
```
>>> words = ['i', 'love', 'loath', 'pizza', 'people']
>>> subwords_prob = calc_subwords_likelihoods(words)
>>> subwords_prob
{'i': {'i': 1.0}, 'l': {'love': 0.75, 'loath': 0.25}, 'lo': {'love': 0.75, 'loath': 0.25}, 'lov': {'love': 1.0},
'love': {'love': 1.0}, 'loa': {'loath': 1.0}, 'loat': {'loath': 1.0}, 'loath': {'loath': 1.0}, 'p': {'pizza':
0.5, 'people': 0.5}, 'pi': {'pizza': 1.0}, 'piz': {'pizza': 1.0}, 'pizz': {'pizza': 1.0}, 'pizza': {'pizza':
1.0}, 'pe': {'people': 1.0}, 'peo': {'people': 1.0}, 'peop': {'people': 1.0}, 'peopl': {'people': 1.0},
'people': {'people': 1.0}}
>>> words_to_complete = ['i', 'lo', 'piz']
>>> complete_subwords(words_to_complete, subwords_prob)
“i love pizza”
```



```
def complete_subwords(words_to_comp, tokens_freq):
```

✓ 3/11 ref

1  
(2.5)





### שאלה 3 (31 נקודות)

בשני ה壽יפים הבאים תמשו משחק לוח בשם "מסלול ראשון".

המשחק "מסלול ראשון" מורכב מלוח משחק בגודל  $N \times N$  ודמות משחק שנמצאת במשבצת מסוימת על הלוח. הדמות יכולה לבצע מהלך על פי הכלל הבא: שני צעדים בכיוון אחד (למעלה, למטה, שמאל או ימין) ואז צעד אחד בכיוונו ניצב לכיוון הראשון. לדוגמה, אם הדמות הולכה שני צעדים למעלה או היא תוכל ללכת צעד אחד ימין או שמאל. באופן דומה, אם הדמות הולכה שני צעדים ימינה היא תוכל ללכת צעד אחד למעלה או למטה.

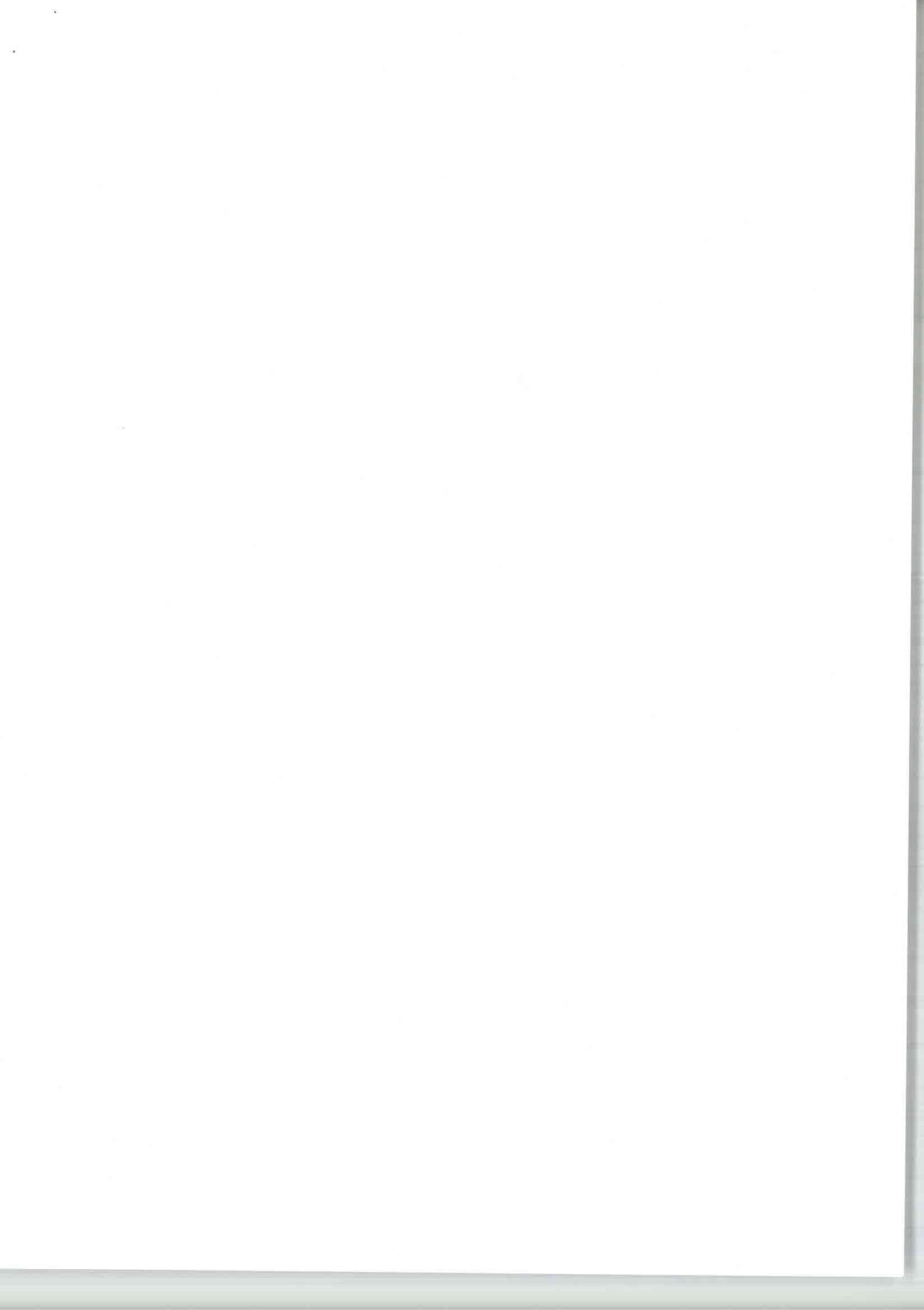
לדוגמה – בשרטוט מטה ניתן לראות את הדמות במרכז הלוח ("דמות") ומוסמנות ב-X כל המשבצות אליהן הדמות יכולה להגיע במהלך אחד.

מספר שורה	0	1	2	3	4	5
מספר עמודה	0	1	2	3	4	5
0	X			X		
1	X				X	
2			דמות			
3	X				X	
4		X		X		
5						

מטרת המשחק: בהינתן לוח משחק בגודל  $N \times N$  ומיקום התחלתי של הדמות  $(j,i)$ , על הדמות לבצע סדרת מהלכים המבקרים **בכל אחת ממשבצות הלוח בדיקוק פעם אחת**. סדרת מהלכים זו נקראת **פתרון**.

למשל, עבור לוח בגודל  $5 \times 5$  ומיקום התחלתי ב-(0,0), מוצג אחד הפתרונות האפשריים. המספר בכל משבצת מסמן את המספר הסידורי של המהלך בפתרון, כך הדמות מתחילה ב-(0,0), עוברת ל-(2,1) וכך הלאה עד ל-(4,0) במהלך .24-ה

מספר שורה	0	5	14	9	20
מספר עמודה	0	1	2	3	4
0	0				
1	13	8	19	4	15
2	18	1	6	21	10
3	7	12	23	16	3
4	24	17	2	11	22



סעיף א' (6 נקודות)

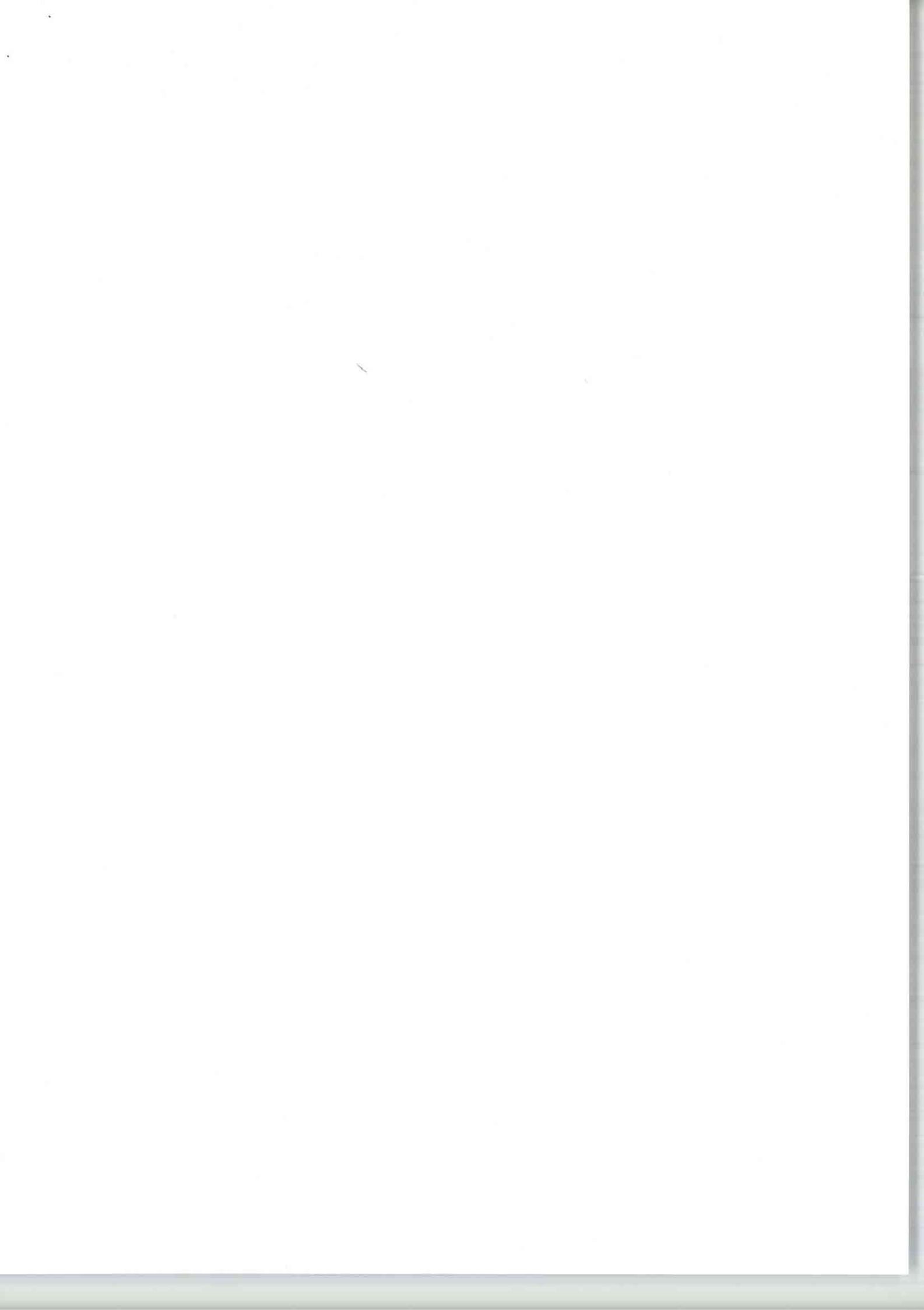
ממשו את הפונקציה get\_moves(row, col, board) אשר מקבלת כקלט שלושה ארגומנטים:

- משתנה מטיפוס int בשם row, המציין את האינדקס של השורה בה נמצאת הדמות.
  - משתנה מטיפוס int בשם col, המציין את האינדקס של העמודה בה נמצאת הדמות.
  - משתנה מטיפוס רשימה של רישומות בשם board המיצג את לוח המשחק. כל משבצת מכילה את הערך None אם ורק אם הדמות עדיין לא ביקרה במשבצת.
- הfonקציה תחזיר משתנה מטיפוס Generator אשר מחזיר את כל המהלך מהמשבצת (row,col) למשבצות שהדמות עדיין לא ביקרה בהן. כל מהלך זה יוצג באמצעות tuple שיכיל את אינדקס השורה והעמודה אליה המהלך יוביל את הדמות.

דוגמת ריצה:

להלן דוגמת ריצה המתאימה ללוח המשחק ולמיוקם ההתחלתי של הדמות לפיה הדוגמא מתחילה השאלה.

```
>>> board = [[None for i in range(6)] for i in range(6)]
>>> list(get_moves(2, 2, board))
[(4, 3), (3, 4), (1, 4), (0, 3), (0, 1), (1, 0), (3, 0), (4, 1)]
>>> board = [[None for i in range(4)] for i in range(4)]
>>> list(get_moves(0, 0, board))
[(2, 1), (1, 2)]
```





```
def get_moves(row, col, board):
```

```
    my_moves = [ ]
```

```
    my_place = board[row][col]
```

**חרינה מהלך (שלילי) -1 (3.1)**

```
    if row-2 > len(board) and col+1 < len(board[0]) and f == None:  
        my_moves.append(tuple([row-2, col+1]))
```

```
    if row+2 >= 0 and col+1 < len(board) and f == None:  
        my_moves.append(tuple([row+2, col+1]))
```

```
    if row-2 <= len(board) and col-1 > 0 and f == None:  
        my_moves.append(tuple([row-2, col-1]))
```

```
    if row-2 <= len(board) and col+1 < len(board[0]) and f == None:  
        my_moves.append(tuple([row-2, col+1]))
```

```
    if col-2 > 0 and row+1 >= 0 and f == None:  
        my_moves.append(tuple([row+1, col-2]))
```

```
    if col+2 < len(board[0]) and row+1 > len(board) and f == None:  
        my_moves.append(tuple([row+1, col+2]))
```

**גרטור שגוי -1**

(3.1)

```
    if col-2 > 0 and row-1 > len(board[0]):
```

```
        my_moves.append(tuple([row-1, col-2]))
```

**סעיף ב' (15 נקודות)**

```
yield my_moves
```

if col+2 < len(board[0]) and col+row+1 < len(board):  
 my\_moves.append(tuple([row, col+2]))

משוו את הפונקציה solve\_game(x, y, board) אשר מקבלת קלט שלושה ארגומנטים המייצגים את לוח המשחק  
ואת המיקום ההתחלתי של הדמות:

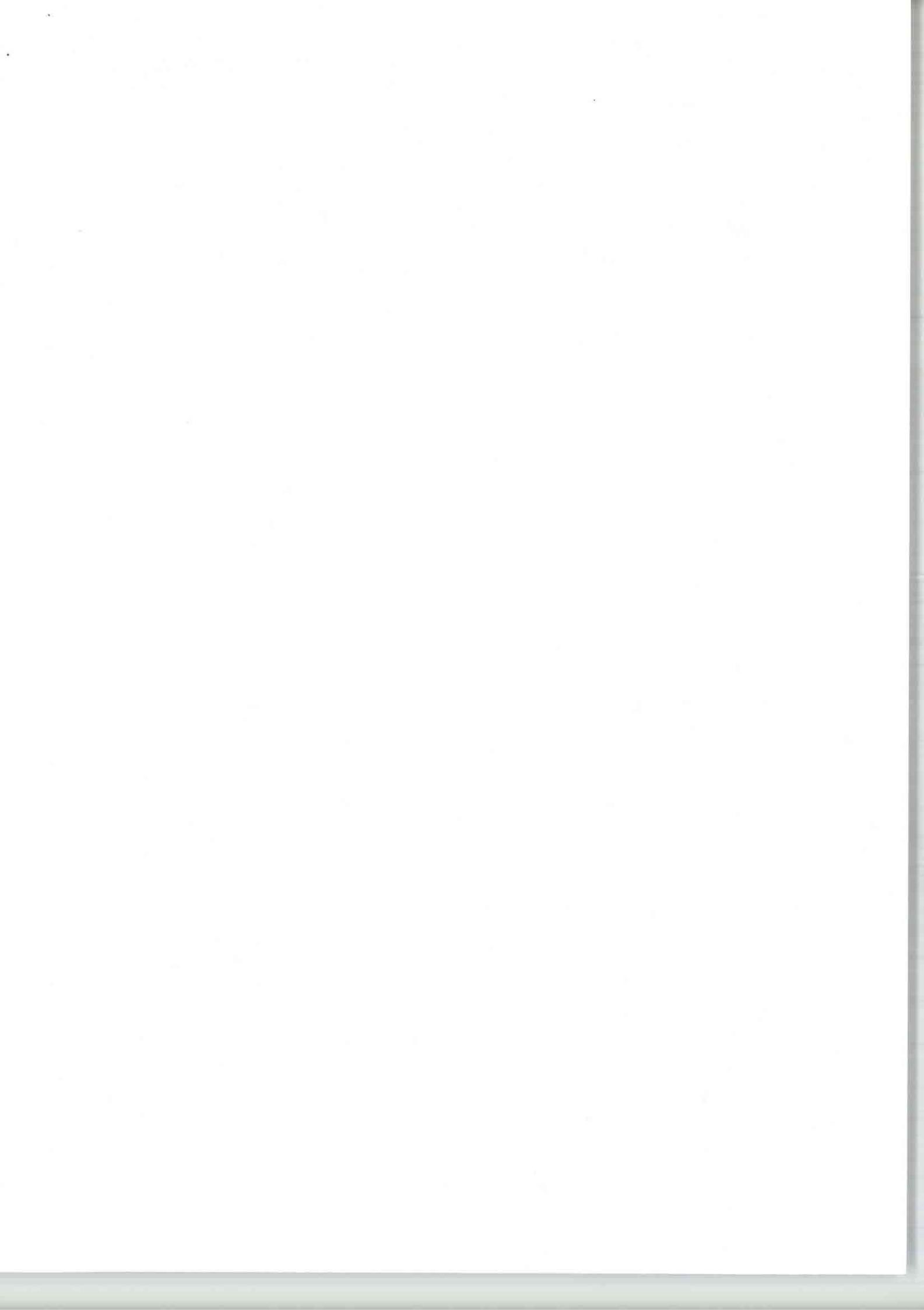
- משתנה מטיפוס int בשם row, המציין את האינדקס של השורה בה נמצאת הדמות.

- משתנה מטיפוס int בשם col, המציין את האינדקס של העמודה בה נמצאת הדמות.

- משתנה מטיפוס רשימה של רשימות בשם board המייצג את לוח המשחק. כל משבצת מכילה את הערך

- None אם ורק אם הדמות עדיין לא ביקרה במשבצת.

הfonktsiya Tchizir True אם קיים פתרון False אחרת. במקרה וקיים פתרון, לוח המשחק (board) ייעודן להציג את סדר מסלול בו הדמות יכולה לעבור בכל משבצת בלוח בדיקוק פעם אחת ואות אחד המסלולים האפשריים כרך שהמספרים ב-board מגדירים את המספר הסידורי של המהלך בפתרון (ראו את הדוגמה בתחילת השאלה). במקרה ואין פתרון אין חשיבות לערכים בלוח המשחק.





שימו לב: במידה ולא הצלחתם לעדכן את הלוח על פי ההוראות, יתקבל גם פתרון עבורו הערך המוחזר הוא האם קיים פתרון (True/False), אולם פתרון זה זוכה ב-8 נקודות לכל היתר.

דוגמת ריצה:

```
>>> board = [[None for i in range(4)] for i in range(4)]
```

```
>>> solve_game(0, 0, board)
```

False

```
>>> board = [[None for i in range(5)] for i in range(5)]
```

```
>>> solve_game(board)
```

True

```
>>> board
```

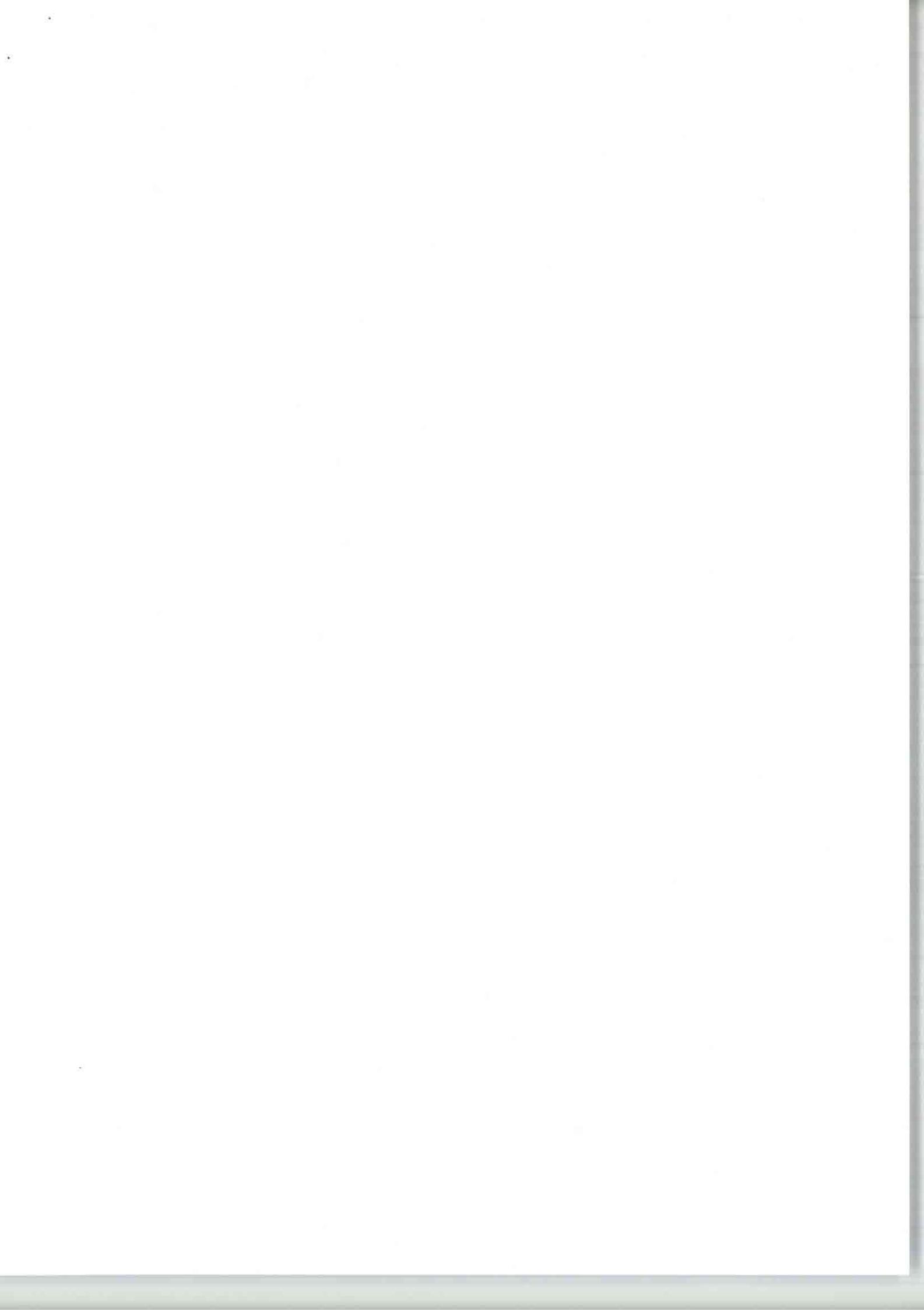
[[0, 5, 14, 9, 20],

[13, 8, 19, 4, 15],

[18, 1, 6, 21, 10],

[7, 12, 23, 16, 3],

[24, 17, 2, 11, 22]]



```
def solve_game(x, y, board):
    if len(board) == 0:
        count = 0
        for i in range(len(board)):
            for j in range(len(board[i])):
                if board[i][j] != None:
                    count += 1
        if count == len(board) * len(board[0]):
            return True
        return False
    זה מיד עוצר
```

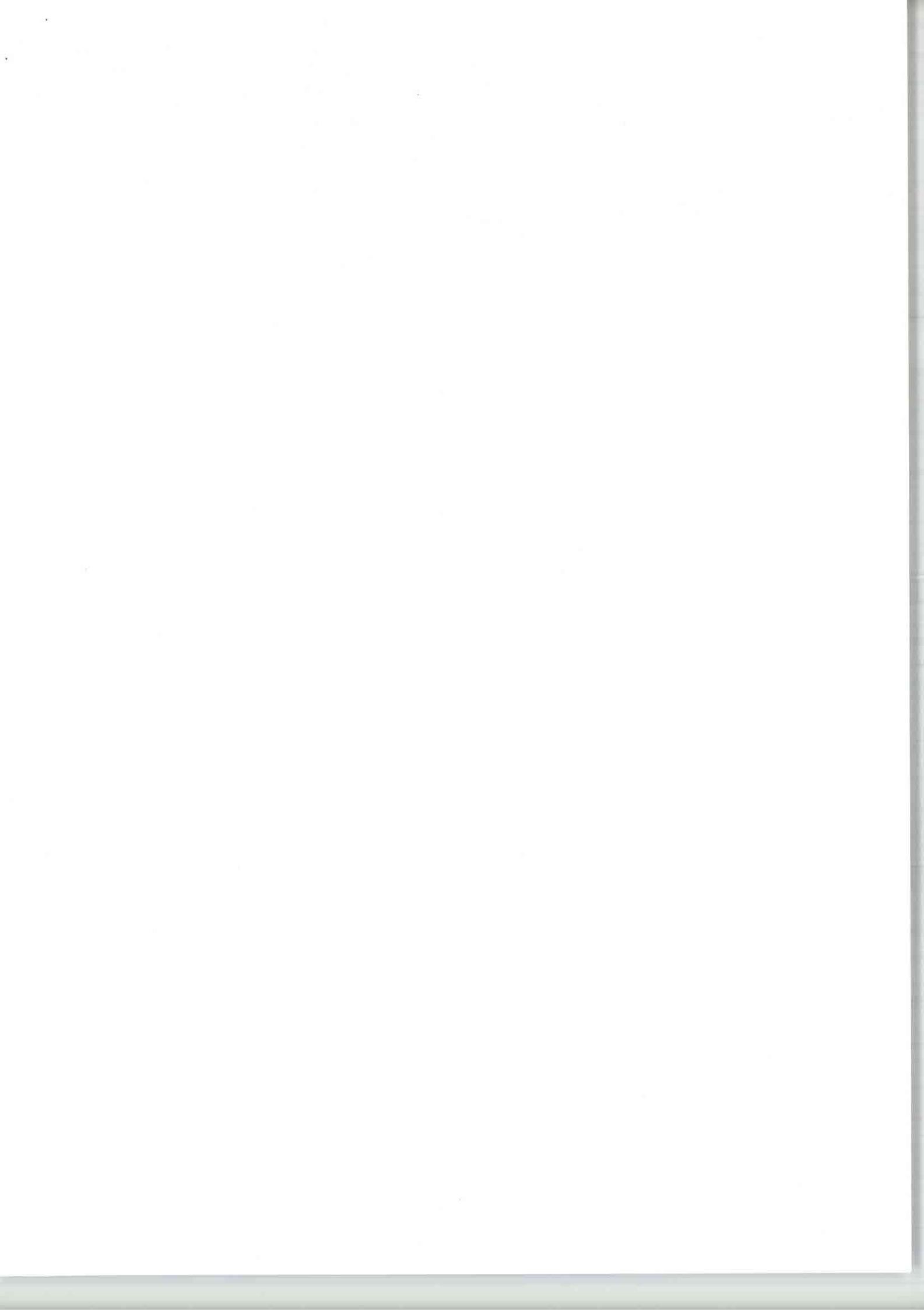
```
for move in range(got_moves(x, y, board)):
    new_board = do_move(x, y, board)
    return solve_game(x, y, new_board)
```

```
def do_move(row, col, board):
    new_board = get_moves(row, col, board).remove((row, col))
    return new_board
```

+2

(3.2)

פתרונות שני. הניקוד על תנאי עצירה  
ב hasilחה.





### שימוש לב - הטעיף הבא אינו תליי בסעיפים הקודמים.

#### **סעיף ג' (10 נקודות)**

טליה אהבת לשנות קולה. מאחר ולטליה חשוב לשמור על הסביבה היא ממחזרת את בקבוקי הקולח אותן היא שותה. חברות המשקאות יצאה במצע "מחזר בקבוקים וקבל בקבוק קולח מלא במתנה". בשאלת זו חסינוו לטליה למקSYM את המבצע.

משוו את הפונקציה **הרכורסיבית** count\_max\_bottle(*money, price, recycling*) המקבלת כקלט שלושה ארגומנטים:

- משתנה מטיפוס int בשם *money*, המגדיר את סכום כסף שיש ברשות טליה.
- משתנה מטיפוס int בשם *price*, אשר מצין את מחיר בקבוק הקולח.
- משתנה מטיפוס int בשם *recycling*, אשר מצין את מספר בקבוקי קולח הריקים שניתן למוחזר ועבורם לקבל בקבוק קולח אחד נוספים מלא.

כל המשתנים והפרמטרים שלמים וחובבים.

הfonקציה תחזיר משתנה מטיפוס int אשר מצין כמה בקבוקי קולח תוכל טליה לשנות באמצעות סכום הכספי שברשותה ועל ידי **ידי מסמוס** אפשרויות המוחזר. כموון שלא ניתן למוחזר חלקית בקבוקים ולכן על הפטרון להחזיר ערך **שלם**.

#### דוגמת ריצה:

```
>> count_max_bottle(money=15, price=1, recycling=3)
```

22

הסביר: בהנחה שברשות טליה 15 ש"ח, וכל בקבוק עולה 1 ש"ח ניתן לרכוש **15 בקבוקים מלאים**. מחזר 15 בקבוקים ריקים יאפשר לקבל **5 בקבוקי קולח מלאים**. החזרת 3 בקבוקים ריקים (מהחmissה) תאפשר לקבל **בקבוק קולח מלא**, שביחד עם שני הבקבוקים הנותרים משלימים 3 בקבוקים ריקים שניתנו **בקבוק קולח נוספים**. בסך הכל נקבל **22 בקבוקים** ( $15 + 5 + 1 + 1 = 22$ ).

```
>>> count_max_bottle(money=16, price=2, recycling=2)
```

15

```
>>> count_max_bottle(money=16, price=2, recycling=3)
```

11

הסביר: בהנחה שברשות טליה 16 ש"ח, וכל בקבוק עולה 2 ש"ח ניתן לרכוש **8 בקבוקים מלאים**. מחזר 6 בקבוקים ריקים אפשר לקבל **2 בקבוקי קולח מלאים**. החזרת 3 בקבוקים ריקים תאפשר לקבל **בקבוק קולח מלא**. בסך הכל נקבל **11 בקבוקים** ( $8 + 2 + 1 = 11$ ).



```

def count_max_bottle(money, price, recycling):
    return count_max_bottle_helper(money, price, recycling)

def count_max_bottle_helper(money, price, sum, recycling):
    if money == 0:
        if recycling // sum == 0:
            recycling = recycling // sum + 1
    else:
        recycling = recycling // sum
    return sum + recycling

r1 = count_max_bottle_helper(money - price, price,
                             sum + 1, recycling)
r2 = count_max_bottle_helper(money, price, sum, recycling)

return max(sum(r1), sum(r2))

```



לא נכון!

