**אוניברסיטת תל אביב - בית הספר למדעי המחשב**
**מבוא מורחב למדעי המחשב, אביב 2017**

# תרגיל בית מספר 3 - להגשה עד 7 במאי 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.
- השתמשו בקובץ השלד skeleton3.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 hw3_012345678.pdf ו- hw3_012345678.py.

- 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, אביב 2017

## שאלה 1

א. הוכיחו או הפריכו את הטענות הבאות. ציינו תחילה בברור האם הטענה נכונה או לא, ואח"כ הוכיחו / הפריכו באופן פורמלי תוך שימוש בהגדרת O.

הנחיה: **יש להוכיח / להפריך כל סעיף בלא יותר מ- 2 שורות**.
הפתרונות הם קצרים, ואינם דורשים מתמטיקה מתוחכמת. אם נקלעתם לתשובה מסורבלת וארוכה, כנראה שאתם לא בכיוון.
אם יש צורך, מותר להשתמש בעובדות הידועות הבאות: $\log n = O(n)$, $n^a = O(n^b)$ עבור a<=b, וכן בטענות שהוכחו בכיתה.
אלא אם צוין אחרת, log הוא לפי בסיס 2.

1. $(10n^2 + 7n)\log n = O(n^2)$
2. $\log((\sum_{k=1}^{n}(k^2))^3) = O(\log n)$
3. $2^{n^3} = O(3^{n^2})$
4. לכל פונקציה $f(n)$, אם $f(n) = O(\log n)$, אז $2^{f(n)} = O(n)$
5. לכל פונקציה $f(n)$, אם $2^{f(n)} = O(n)$, אז $f(n) = O(\log n)$.

ב. לכל אחת מהפונקציות הבאות, נתחו את סיבוכיות זמן ריצתה כתלות ב- n (אורך הרשימה lst). הניחו כי כל פעולה בודדת (ובכלל זה פעולת כתיבה של איבר ברשימה לזיכרון) דורשת O(1) זמן. ציינו את התשובה הסופית, ונמקו. על הנימוק להיות קולע, קצר וברור, ולהכיל טיעונים מתמטיים או הסברים מילוליים, בהתאם לצורך.

על התשובה להינתן במונחי O(…), ועל החסם להיות הדוק ככל שניתן. למשל, אם הסיבוכיות של פונקציה היא O(n) ובתשובתכם כתבתם O(nlogn), התשובה לא תקבל ניקוד (על אף שפורמלית O הוא חסם עליון בלבד).

1.
```
def f1(lst):
   n = len(lst)
   for i in range(n):
       lst.extend(range(2*n))
```

2.
```
def f2(lst):
   n = len(lst)
   for i in range(n):
       lst.extend(range(2*i))
```

3.
```
def f3(lst):
   n = len(lst)
   for i in range(n):
       lst.extend(range(len(lst), len(lst)+500))
```

4.
```
def f4(lst):
   n = len(lst)
   for i in range(n):
       lst = lst + list(range(len(lst)))
```

ג. הסבירו <u>בקצרה</u> מדוע הרצת קטע הקוד הבא תיכנס ללולאה אינסופית ולא תסתיים. היעזרו במה שלמדנו על מודל הזיכרון של פייתון.

```
l = [1,2,3]
for e in l:
   l += ["a"]
```

## <u>שאלה 2</u>

בהרצאה 8 נראה את שיטת ניוטון רפסון למציאת שורש של פונקציה. לפעמים, כאשר ניתן להניח הנחות מסויימות על הפונקציה שאת השורש שלה מחפשים, עדיף להשתמש בשיטות אחרות, שעשויות לתת תוצאות מדוייקות יותר או לעקוף את המגבלות של שיטת ניוטון רפסון. בשאלה זו נדון במקרה כזה (את הפסקה האחרונה יש לקרוא כ"אין להשתמש בשאלה זו בשיטת ניוטון רפסון").

תהי $f: \mathbb{N}^+ \to \mathbb{Z}$ פונקציה שמקבלת כקלט מספר טבעי (גדול מ-0) ומחזירה כפלט מספר שלם. נאמר ש- $f$ גדלה מונוטונית אם לכל ערך $x$ מתקיים ש $f(x+1) > f(x)$ . בהינתן פונקציה $f$ שגדלה מונוטונית, נרצה למצוא את הערך $n$ עבורו $f(n) = 0$ , אם יש כזה . (מאחר ש- $f$ גדלה מונוטונית אז מתקיים לכל $x > n$ ש- $f(x) > 0$ וכמו כן לכל $x < n$ מתקיים ש $f(x) < 0$.)
בשאלה זו הניחו כי סיבוכיות הזמן עבור הפעלת $f$ על קלט כלשהו הינה $O(1)$.

א.

I. השלימו את מימוש הפונקציה `find_root1` שמקבלת כקלט פונקציה $f: \mathbb{N}^+ \to \mathbb{Z}$ הגדלה מונוטונית ומחזירה את השורש של $f$ אם קיים כזה, אחרת מחזירה None. שימו לב כי בסעיף זה המימוש הוא נאיבי, ללא ניסיון לייעל את זמן הריצה, אלא פשוט ע"י בדיקת ערכים עוקבים של n, החל ב- 1 (כלומר n=1,2,3,…).
דוגמאות הרצה:

```
>>> f = lambda x : x - 10
>>> find_root1(f)
10
>>> g = lambda x : 2*x + 1
>>> find_root1(g) #returned None
>>>
```

II. ציינו בקובץ ה- pdf מהי סיבוכיות זמן הריצה של `find_root1` כפונקציה של הערך $k$ המינימלי עבורו $f(k) \geq 0$? תנו תשובה במונחי סדר גודל $O(...)$, הדוקה ככל שתוכלו ונמקו בקצרה.

ב. השלימו בקובץ השלד את הפונקציה `find_root_range` שמקבלת פונקציה גדלה מונוטונית $f: \mathbb{N}^+ \to \mathbb{Z}$ ושני מספרים חיוביים, $a \leq b$ ומחזירה את השורש של $f$ בקטע $[a, b]$ אם קיים כזה, אחרת מחזירה None.

דרישה מחייבת: הפונקציה צריכה לרוץ בסיבוכיות $O(\log(b-a))$.

דוגמאות הרצה:

```
>>> f = lambda x : x - 10
>>> find_root_range(f, 1, 12)
10
>>> find_root_range(f, 12, 20) #returned None
```

ג.

I. השלימו בקובץ השלד את הפונקציה `find_root2` שמקבלת כקלט פונקציה $f: \mathbb{N}^+ \to \mathbb{Z}$ שגדלה מונוטונית ומחזירה את השורש של $f$ אם קיים כזה, אחרת מחזירה None. על הפונקציה `find_root2` לרוץ בסיבוכיות זמן $O(\log k)$, כאשר $k$ הינו הערך המינימלי המקיים כי $f(k) \geq 0$.

על הפונקציה להשתמש בפונקציה `find_root_range` שכתבתם בסעיף ב'.

דוגמת הרצה:

```
>>> f = lambda x : x - 10
>>> find_root2 (f)
10
```

II. ציינו בקובץ ה- pdf מדוע הפונקציה שכתבתם עונה על דרישות הסיבוכיות.

## שאלה 3

בשאלה זו נעסוק בהילוכים אקראיים (random walks) על סריגים במרחב החד מימדי, במרחב הדו מימדי, במרחב התלת מימדי, ובמרחב הארבע מימדי (מונחים אלו יוסברו מייד). לתורה של הילוכים אקראיים יש שימושים רבים בתחומי ההסתברות המתמטית, הפיזיקה, הכימיה, הביולוגיה, הכלכלה, ועוד ועוד. זאת ועוד, הילוכים אקראיים יכולים לתאר את המצב הכספי של מהמר המבצע סדרה ארוכה של הימורים בזה אחר זה (אנחנו ממליצים להציץ בפרק בשם "מהמרים כושלים, שיכורים בביוב, ואלוהים" בבלוג הנפלא "לא מדויק" של גדי אלכסנדרוביץ').

בהילוך אקראי על סריג d-מימדי אנחנו מתחילים את ההילוך בראשית, כלומר בנקודה (0,0,…,0) בעלת d קואורדינטות. בכל צעד אנחנו מבצעים תזוזה של +1 או -1 בכל אחת מ-d הקואורדינטות, באופן בלתי תלוי. למשל עבור d=3, הילוך כזה מתחיל מ-(0,0,0), ויכול להתקדם, למשל, ל-(1,1,-1), משם ל- (2,0,-2) , משם ל- (3,-1,-1), וכך הלאה.

התזוזות בקואורדינאטות השונות הן (כאמור) בלתי תלויות, והתזוזות בין צעדים שונים הן בלתי תלויות. ההילוך נקרא בלתי מוטה אם ההסתברות לתזוזה של +1 בכל אחת מהקואורדינאטות היא בדיוק 1/2, והוא נקרא מוטה אם ההסתברות לתזוזה של +1 בכל אחת מהקואורדינאטות היא p, המקיים p>1/2. בתרגיל זה נבצע סימולציות של הילוכים מקריים בלתי מוטים, ונמדוד מספר תכונות שלהם. כל הילוך כזה יאופיין על ידי מימדי הסריג עליו הוא מתבצע (זה יקבל ערכים d=1,2,3,4), ומספר הצעדים שנבצע בהילוך, run_length.

אנחנו נתעניין בהיבטים שונים של שתי התכונות הבאות:

1. המרחק של המיקום בסוף ההילוך לראשית (מרחק אוקלידי). נסמן מרחק הזה על ידי dist (מספר ממשי אי שלילי). ניתן למצוא טענות בספרות כי המרחק הממוצע פרופורציוני לשורש מספר הצעדים, עם קבועי פרופורציה שונים למימדים השונים.
2. האם במשך ההילוך חזרנו (בנקודת זמן כלשהי) לראשית. נסמן זאת על ידי back_to_origin (משתנה בוליאני).

כיון שמדובר בתהליכים הסתברותיים עם שונות לא קטנה, לא נסתפק בתוצאה המתקבלת מסימולציה של הילוך בודד, אלא נבצע מספר הילוכים (number_runs) עבור כל קבוצת פרמטרים. נמדוד את הממוצע, המינימום, והמקסימום (עבור dist) ואת השכיחות לקבלת הערך True (עבור back_to_origin). כדי לחקות את התזוזות האקראיות בכל צעד של ההילוך ניתן להשתמש עקרונית בהטלות מטבע, במדידות של הקרינה הקוסמית, או במדידות של תופעות פיזיקליות אקראיות אחרות, אבל שיטות אלו אינן יעילות. במקום זאת, נעזר בחבילת random של Python. נקרא לה, כמקובל, באמצעות import random. כדי לבחור בין +1 ו-1- בהסתברות שווה, ניתן להשתמש במתודה random.choice([1,-1]) .

א. כתבו פונקציה הנקראת walk(run_length, d). הפונקציה תסמלץ הילוך בודד באורך נתון run_length בסריג ממימד נתון d. הפונקציה תחזיר שני פלטים:
   - המרחק האוקלידי dist מהמיקום בסוף ההילוך לראשית הצירים
   - ערך בוליאני back_to_origin שהינו True אם במשך ההילוך חזרנו לראשית (פעם אחת או יותר).

שורת הקוד הבאה מחזירה את שני הפלטים dist ו- back_to_origin:

return dist, back_to_origin

דוגמאות הרצה:

```
>>> walk(10,2)
(6.324555320336759, True)
>>> walk(10,2)
(5.656854249492381, False)
>>> dist, back_to_origin  = walk(10000, 4)
>>> dist
143.08039698015938
>>> back_to_origin
False
```

ב. כעת עליכם לכתוב פונקציה בשם rw_stats שתאסוף סטטיסטיקות על מספר הרצות של walk על מימד נתון d, ומספר צעדים נתון run_length. הפונקציה rw_stats תקבל שלושה ארגומנטים, run_length (אורך ההילוך), d (מימד הסריג בו מתבצע ההילוך) ו- number_runs (מספר ההילוכים שמסמלצים). הפונקציה תקרא ע"י:

rw_stats(run_length, d, number_runs=10**3)

הפונקציה תחזיר שישה פלטים: מספר ההילוכים (זה מוחזר למען נוחותנו בעת בדיקת הפלטים), ממוצע המרחקים, ממוצע המרחקים מחולק בשורש הריבועי של אורך ההילוך, מינימום המרחקים, מקסימום המרחקים, ושכיחות החזרה לראשית במשך ההילוכים (מספר ההילוכים בהם היתה חזרה לראשית, חלקי מספר ההילוכים הכולל) . הפונקציה rw_stats צריכה לקרוא לפונקציה walk. כמו כן היא צריכה להשתמש במספר קבוע של משתנים (מספר שאינו תלוי באורך ההילוך. בפרט אין להשתמש ברשימה שאורכה תלוי באורך ההילוך), דהיינו O(1) זכרון נוסף. הפונקציה תחזיר את הפלטים באופן הבא:

return run_length, average_dist, average_dist/(run_length**0.5),\
min_dist, max_dist, origin_frequency

דוגמאות הרצה:

```
>>> rw_stats(1000, 3, 50)
(1000,  47.9639725322366,  1.5167539883162158,  7.483314773547883,
90.64215354899729, 0.32)
>>> rw_stats(1000, 3, 50)
(1000, 48.74902772542371, 1.5415796133103634, 20.396078054371138,
85.25256594378845, 0.22)
```

ג. כעת עליכם לבצע את הניסוי הבא ולדווח על תוצאותיו (להדביק תמונה של הפלט) בקובץ ה pdf:
עליכם להריץ שתי לולאות מקוננות. בלולאה החיצונית, עברו על המימדים d מ-1 ועד 4. בתחילת כל לולאה חיצונית, הדפיסו print("\n Simulation results for dimension",d) . אח"כ יהיה עליכם לחשב סטטיסטיקות ע"י קריאה לפונקציה rw_stats(2**i, d, number_runs=10**3) עם אורכי הילוך 2**i כאשר i in range(7,17).

הקפידו לצרף לקובץ ה pdf את תוצאות כל ההרצות מוצגות בטבלה. בנוסף עליכם לענות על השאלות הבאות:

1. מה הסיכוי שבשתי הרצות שונות יוחזרו אותן תוצאות (0, חיובי אך זניח, סביר אך לא קרוב לודאי, קרוב לודאי אך לא בטוח, בטוח). הסבירו את תשובתכם בשורה או שתיים. אין צורך בהוכחה.
2. כיצד מתנהגת שכיחות החזרה למקור כפונקציה של המימד?
3. האם התוצאות שקיבלתם תומכות בטענה כי המרחק הממוצע אכן פרופורציוני לשורש מספר הצעדים? אם כן, מהו (בערך) קבוע הפרופורציה ומהי התלות שלו במימדים השונים?
4. האם אתם מזהים קשרים בין מינימום המרחק לבין המימד?

שימו לב כי עבור הילוכים ארוכים, זמני הריצה יהיו ארוכים למדי. מומלץ לפתח את הקוד ולבדוק אותו עם ערכים קטנים של run_length, למשל 128, ורק אחרי שהשתכנעתם בתקינות הקוד לבצע פעם אחת הרצות ארוכות כנדרש בשאלה.

## שאלה 4

א. אמיר ומיכל קנו כל אחד לפטופ חדש. הלפטופ של מיכל מהיר פי 2 מזה של אמיר, כלומר הוא מסוגל לבצע פי 2 פעולות ליחידת זמן. שניהם מריצים על המחשבים שלהם את אותו אלגוריתם למשך דקה אחת.
בכל אחד מהסעיפים הבאים מצויין חסם הדוק לסיבוכיות הזמן של האלגוריתם.
עבור כל אחד מהסעיפים למטה, נניח שהמחשב של אמיר מסוגל לסיים בדקה את ריצתו על קלט מקסימלי בגודל n=1000. בהינתן שחסם הדוק על סיבוכיות האלגוריתם הוא $T(n)$, רשמו מהו גודל הקלט המקסימלי עליו יסיים המחשב של מיכל לרוץ בדקה. ציינו את התשובה המספרית, ונמקו בשורה אחת.

a. $T(n) = O(\log n)$

b. $T(n) = O(n)$

c. $T(n) = O(n^2)$

d. $T(n) = O(2^n)$

ב. הגדרה: סיבוכיות זיכרון, או מקום של אלגוריתם, היא הכמות המקסימלית של תאי זיכרון שהאלגוריתם מקצה בכל שלב במהלך פעולתו, מעבר למקום שתופס הקלט שלו.
בכיתה ראינו שלושה אלגוריתמים בסיסיים במדעי המחשב: חיפוש בינארי, מיון בחירה, ומיזוג. לכל אחד מהאלגוריתמים, ציינו מהי סיבוכיות הזיכרון (space complexity) שלו, כתלות בגודל הקלט שלו (ובמקרה של מיזוג – גודל שני הקלטים שלו), במונחים אסימפטוטיים (כלומר באמצעות הסימון O(…) ).

## שאלה 5

בכיתה ראינו את האלגוריתם מיון-בחירה (selection sort) למיון רשימה נתונה. האלגוריתם כזכור רץ בסיבוכיות זמן $O(n^2)$ עבור רשימה בגודל $n$. בקורס נראה גם אלגוריתם מיון-מהיר יעיל יותר (quicksort), שרץ בסיבוכיות זמן ממוצעת $O(n\log n)$. לפעמים, כאשר יש לנו מידע נוסף על הקלט, אפשר למיין בסיבוכיות זמן טובה מזו. למשל, בשאלה זו, נעסוק במיון של רשימה שכל איבריה מוגבלים לתחום מצומצם יחסית: זוגות (a,b) כאשר a ו- b מוגבלים להיות מספרים שלמים בתחום [0, k-1], עבור k>0 נתון כלשהו. נייצג זוג כזה בפייתון ע"י אובייקט מסוג tuple.

נגדיר ש- (a1,b1) < (a2,b2) אם"ם a1<a2 או שמתקיים a1=a2 וגם b1<b2.

| |
|---|
| <u>הערה</u>: בניתוח הסיבוכיות בשאלה זו נניח שגודל כל מספר הינו קבוע. לכן פעולות כגון השוואה בין שני מספרים או העתקה של מספר ממקום אחד לאחר בזיכרון רצות בזמן קבוע. |

א. השלימו בקובץ השלד את הפונקציה sort_pairs1(lst, k) שמקבלת כקלט רשימה lst של זוגות מסוג tuple כמתואר ומספר חיובי k כך שבכל זוג ברשימה ערכו של כל איבר הוא בין 0 ל k-1. (הניחו כי הקלט תקין ואין צורך לבדוק את תקינותו.) על הפונקציה להחזיר רשימה <u>חדשה</u> ממויינת בסדר עולה (ולא לשנות את lst עצמה). בנוסף לרשימת הפלט שהיא בגודל n כמובן, על הפונקציה להשתמש ברשימת עזר (list) בעלת k*k איברים. רשימה זו יכולה להיות "חד מימדית" או "דו מימדית" (רשימה של רשימות פנימיות), לבחירתכם.

ב. ציינו בקובץ ה pdf ונמקו בקצרה מהי סיבוכיות זמן הריצה של הפונקציה מסעיף א' כתלות ב n ו/או k, כאשר כאשר $n$ הוא אורך הרשימה lst, ו- k הינו הערך השני שהועבר לפונקציה שכאמור מציין את טווח המספרים.

ג. השלימו בקובץ השלד את הפונקציה sort_pairs2(lst,k) שמקבלת קלטים כמו הפונקציה מסעיף א', ובדומה לפונקציה הקודמת עליה להחזיר רשימה חדשה ממויינת בסדר עולה (ולא לשנות את lst עצמה).
הפעם, בנוסף לרשימת הפלט שהיא בגודל n כמובן, על הפונקציה להיעזר לצורך פעולתה במספר קבוע O(1) (שאינו תלוי ב n ו/או k) של משתני עזר. בפרט, אסור להשתמש ברשימה בגודל שתלוי בפרמטרים n ו- k.
דרישה זו עלולה לגרום לסיבוכיות הזמן להיות גדולה יותר מאשר זו של sort_pairs1.

ד. ציינו בקובץ ה pdf ונמקו בקצרה מהי סיבוכיות זמן הריצה של הפונקציה מסעיף ב' כתלות ב n ו/או k, כאשר כאשר $n$ הוא אורך הרשימה lst, ו- k הינו הערך השני שהועבר לפונקציה שכאמור מציין את טווח המספרים.

דוגמת הרצה:

```
>>> import random
>>> k = 1000
>>> lst = [(random.choice(range(k)), random.choice(range(k))) for i in
range(8)]
>>> lst
[(258, 304), (264, 12), (324, 878), (981, 142), (49, 821), (68, 559), (479,
144), (180, 238)]
>>> sort_pairs1(lst, k)
```

```
[(49, 821), (68, 559), (180, 238), (258, 304), (264, 12), (324, 878), (479,
144), (981, 142)]
>>> sort_pairs2(lst, k)
[(49, 821), (68, 559), (180, 238), (258, 304), (264, 12), (324, 878), (479,
144), (981, 142)]
>>> sorted(lst) == sort_pairs1(lst,k)
True
>>> sorted(lst) == sort_pairs2(lst,k)
True
```

## שאלה 6

בשאלה זו נעסוק באלגוריתם ניוטון-רפסון.

להלן תזכורת לקוד של הפונקציות NR ו- diff_param שנלמדו:

```
1. import random
2.
3. def diff_param(f,h=0.001):
4.     return (lambda x: (f(x+h)-f(x))/h)
5.
6. def NR(func, deriv, epsilon=10**(-8), n=100, x0=None):
7.    """ Given a real valued func and its real value derivative,
8.    deriv, NR attempts to find a zero of function, using the
9.    Newton-Raphson method.
10.   NR starts w ith a an initial x0 (default value is None
11.   which is replaced upon execution by a random number distributed
12.   uniformly in (-100.,100.)), and performs n=100 iterations.
13.   If the absolute value function on some x_i is smaller
14.   than epsilon, None is the returned value """
15.
16.   if x0 is None:
17.      x0 = random.uniform(-100.,100.)
18.   x = x0; y = func(x)
19.   for i in range(n):
20.      if abs(y) < epsilon:
21.         print(x, y, "convergence in", i, "iterations")
22.         return x
23.      elif abs(deriv(x)) < epsilon:
24.         print("zero derivative, x0=", x0," i=", i, " xi=", x)
25.         return None
26.      else:
27.         print(x,y)
28.         x = x - func(x)/deriv(x)
29.         y = func(x)
30.   print("no convergence, x0=", x0," i=", i, " xi=", x)
31.   return None
```

א. בהינתן שתי פונקציות f1(x) ו- f2(x), שתיהן גזירות (כלומר עומדות בתנאי הנדרש עבור שיטת ניוטון רפסון), נרצה למצוא ערך x בו הפונקציות נפגשות, כלומר f1(x) == f2(x). למשל עבור:

```
f1 = lambda x:4*x
f2 = lambda x:x**2+3
```

הפונקציות נפגשות בנקודות x=1 ו- x=3.

השלימו בקובץ השלד את הפונקציה equal(f1, f2) המחזירה ערך x כזה, אם קיים. יש להשלים שתי שורות בלבד. הפונקציה תקרא ל- NR.

כל המגבלות החלות על פעולתה של NR כמובן חלות גם על equal. למשל, יכול להיות שקיימת נקודת מפגש, אבל equal לא תמצא אותה כי NR לא החזירה תשובה כנדרש. יש להשלים שתי שורות בלבד.

ב. מה יקרה כאשר יועברו ל- equal שתי פונקציות שאין להן כלל נקודת מפגש? להלן שני מקרים כאלו. עליכם להחליט עבור כל מקרה, האם:

1) equal תחזיר None ותודפס השורה מתוך NR שמתחילה ב- "zero derivative"
2) equal תחזיר None ותודפס השורה מתוך NR שמתחילה ב- "no convergence"
3) אף על פי שאין לפונקציות נקודת מפגש equal תחזיר ערך מספרי כלשהו (שכמובן אינו נכון).

```
>>> equal(lambda x:x, lambda x:x**2+1)

>>> equal(lambda x:x, lambda x:x+1)
```

את תשובתכם כיתבו בקובץ ה pdf והסבירו בקצרה.

ג. כעת נרצה לחשב את הפונקציה ההופכית של פונקציה נתונה, תוך שימוש באלגוריתם ניוטון רפסון. הפונקציה ההופכית של פונקציה f (מסומנת $f^{-1}$) מקיימת לכל x ו- y: $f^{-1}(y) = x$ אם ורק אם $f(x) = y$.

1. השלימו בקובץ השלד את הפונקציה source, שמקבלת כקלט פונקציה f וערך y, ומחזירה x עבורו מתקיים בקירוב טוב $f(x) = y$. תוכלו להניח כי קיים x יחיד כזה. על source לקרוא ל- NR (פתרונות אחרים לא יתקבלו). "קירוב טוב" ייקבע ע"י הערך epsilon של NR, ואין לשנות ערך זה, או ערכי ברירת מחדל אחרים. אין צורך לטפל במקרים בהם NR מחזירה None. שימו לב כי בגלל הניחוש ההתחלתי האקראי של NR, התוצאה יכולה להיות שונה מריצה לריצה (כפי שניתן לראות להלן):

```
>>> lin = lambda x: x+3
>>> source(lin,5)
2.0000000003798846
>>> source(lin,5)
1.9999999995163051
```

2. השלימו בקובץ השלד את הפונקציה inverse (יש להשלים שורה אחת בלבד), שמקבלת כקלט פונקציה f ומחזירה את ההופכית שלה $f^{-1}$ (עד כדי הדיוק שמצויין בפונקציה NR). הניחו כי ל- f אכן קיימת פונקציה הופכית. יש להשתמש בפונקציה מהסעיף הקודם. דוגמת הרצה:

```
>>> inverse(lin)(5)
1.9999999998674198
```

**סוף**