מבחן בקורס מבוא מורחב למדעי המחשב, CS1001.py

סמסטר א' 2013

מועד ב

מרצים: פרופ' בני שור, ד"ר דניאל דויטש

מתרגלים: אילן בן בסט, אמיר רובינשטיין, אדם ויינשטוק

משך הבחינה: 3 שעות.

חומר עזר מותר: שני דפי עזר (דו צדדיים) בגודל A4 כ"א.

- יש לכתוב את כל התשובות בטופס הבחינה. המחברת תשמש כטיוטה בלבד ולא תקרא.
- במבחן 12 עמודים (כולל עמוד ריק בסוף) – בידקו שכולם בידיכם. המבחן מכיל 4 שאלות "חצי סגורות" ושתי שאלות "פתוחות", אשר בכ"א מהן שני סעיפים. מומלץ לקרוא תחילה את כל השאלות.
- אנו ממליצים לא "להיתקע" על שום שאלה בודדת, להמשיך לשאלות אחרות, ולחזור לשאלה אח"כ.
- בכל סעיף בשאלות הפתוחות, ניתן לכתוב "איני יודע/ת" ולא לכתוב שום טקסט נוסף. במקרה זה יינתן 20% מציון הסעיף (מעוגל כלפי מעלה).
- יש לכתוב את כל התשובות במסגרת המקום המוקצב ובכתב קריא. חריגות משמעותיות מהמקום המוקצב, או הכתובות תוך שימוש בפונט זערורי, לא ייקראו. תשובות שמצריכות עזרים אופטיים מתקדמים לקריאתן, כאלה הכתובות בכתב חרטומים, או הדורשות כוחות על טבעיים להבנתן, עלולות לגרום להשפעה שלילית על הציון.
- לכתיבת קוד יש להקדים תיאור מילולי קצר שיסביר את הקוד במקום המיועד לכך.
- ניתן לצטט טענות שנטענו בהרצאה, בתרגול, או בשיעורי הבית.

טבלת ציונים: (לשימוש הבודקים)

שאלה	ניקוד
1	
2	
3	
4	
5	
6	
סה"כ	

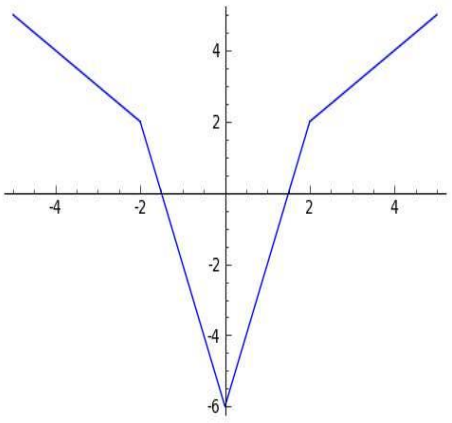
בהצלחה !

מס' ת"ז_____ מס' מחברת_____

<u>שאלה 1 (12 נק')</u>

הפונקציה f מתוארת גרפית על ידי התרשים הבא (למטה). מילולית, אם הערך המוחלט של x קטן או שווה 2, ערך הפונקציה הוא 4*abs(x)-6 ואחרת הוא abs(x).

שימו לב כי הפונקציה f היא רציפה, אך אינה גזירה. ליתר דיוק, לנגזרת שלה יש שלוש נקודות אי רציפות, ב- x=-2, x=0, x=2.



נגדיר בעזרת קוד בפייתון שתי פונקציות d ו- f. הפונקציה f היא הפונקציה שהוגדרה מילולית קודם.

הפונקציה d תייצג נגזרת "מימין" או "משמאל" של f, כאשר:

d(-2) = -4, d(0) = 4, d(2) = 4

```
def f(x):
    if abs(x)<=2:
        return 4*abs(x)-6
    else:
        return abs(x)
```

```
def d(x):
    if abs(x)<=2:
        if x>=0:
            return 4
        else:
            return -4
    else:
        if x>=0:
            return 1
        else:
            return -1
```

נריץ כעת את הקוד למציאת שורש בעזרת שיטת ניוטון-רפסון, עם הפונקציה f(x), ובתפקיד ה"נגזרת" שלה תשמש d(x). נקודת התחלה תהיה t כלשהי. כלומר נקרא ל- NR(f, d, x0=t).

איזו מהטענות הבאות נכונה? <u>הקיפו בעיגול</u>, והצדיקו את תשובתכם ע"י מתן הסבר קצר או דוגמא קצרה שיכתבו בשורות הבאות (הסברים ארוכים יותר יקוצצו בבדיקה):

1) לכל t יוחזר השורש x=1.5.

2) יש t עבורם יוחזר השורש x= 1.5 ויש t עבורם יוחזר השורש x= -1.5.

3) הפונקציה f אינה גזירה ולכן לכל t שיטת ניוטון-רפסון תיכנס לריצה אינסופית.

4) לכל t יוחזר השורש x= -1.5.

מס' ת"ז_____ מס' מחברת_____

הסבר קצר או דוגמא קצרה:

הפונקציה NR שנלמדה בכיתה:

```
def NR(func, deriv, epsilon=10**(-8), n=100, x0=None):
  if x0 is None:
        x0 = uniform(-100.,100.)
  x=x0; y=func(x)
  for i in range(n):
        if abs(y)<epsilon:
              return x
        elif abs(deriv(x))<epsilon:
              print ("zero derivative, x0=",x0," i=",i, " xi=", x)
              return None
        else:
              x=x- func(x)/deriv(x)
              y=func(x)
  print("no convergence, x0=",x0," i=",i, " xi=", x)
  return None
```

<u>שאלה 2 (14 נק')</u>

להלן שיטה לאיתור שגיאות (הנקראת שיטת Berger):

בהינתן 7 ביטים של אינפורמציה, נוסיף להם 3 ביטים שייצגו את מספר האפסים ב- 7 הביטים הללו, בכתיב בינארי.

דוגמאות:

- כדי לשלוח את 1001100, נוסיף 100, משום שיש ארבעה אפסים. בסה"כ נשלח אם כן 1001100<u>100</u>.
- כדי לשלוח את 1111111, נוסיף 000, משום שיש אפס אפסים. בסה"כ נשלח אם כן 1111111<u>000</u>.

א. מהו מרחק Hamming של הקוד? _____

ב. מהו מספר השגיאות שניתן לגלות (detect) ושניתן לתקן? לגלות_____ לתקן_____.

ג. נניח שנפל מספר לא ידוע של שגיאות, אבל ידוע שהשגיאות כללו אך ורק אפסים (0) שהפכו לאחדים (1). הקיפו בעיגול את הטענה הנכונה והצדיקו את תשובתכם ע"י מתן הסבר קצר או דוגמא קצרה שיכתבו בשורות הבאות (הסברים ארוכים יותר יקוצצו בבדיקה):

1. במצב כזה לא נוכל לגלות אפילו שגיאה אחת.
2. במצב כזה נוכל לגלות שגיאה בודדת, אך לא יותר מכך.
3. במצב כזה נוכל לגלות עד 2 שגיאות, אך לא יותר מכך.
4. במצב כזה נוכל תמיד לגלות כל כמות של שגיאות.

<u>שאלה 3 (12 נק')</u>

נתונה הפונקציה הבאה:

```
def mystery(x,y):
   if (y==0):
      return 1
   if (y % 2 == 0):
      return mystery(x**2, y//2)
   else:
      return x*mystery(x, y-1)
```

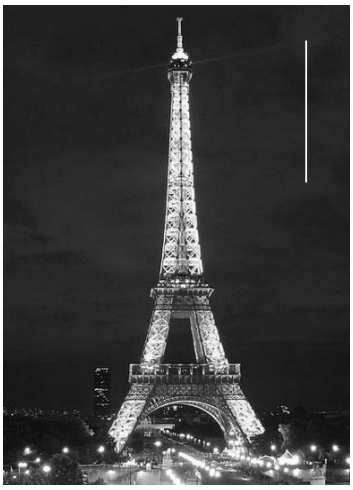
א. תארו בקצרה מה מחשבת הפונקציה, בהינתן לה x ו- y טבעיים.

ב. נסמן ב- n את מספר **<u>הביטים</u>** של y. מהי סיבוכיות זמן הריצה של הפונקציה, כתלות ב- n? הניחו כי פעולות אריתמטיות כגון העלאה בריבוע, כפל, חילוק וכו' לוקחות זמן ריצה קבוע O(1). הצדיקו את תשובתכם ע"י מתן הסבר קצר שיכתב בשורות הבאות (הסברים ארוכים יותר יקוצצו בבדיקה):

O( )

<u>שאלה 4 (12 נק')</u>

להלן תמונה ש"התלכלכה" – באזור שחור כלשהו נוצר מלבן לבן ברוחב 3 פיקסלים ובאורך 100 פיקסלים, כפי שניתן לראות להלן.



א. רוצים לנקות את הלכלוך לחלוטין תוך פגיעה מינימאלית באיכות התמונה. לשם כך מפעילים פעם אחת בלבד שיטת ניקוי כלשהי.

באיזו שיטה ניקוי – ממוצע מקומי או חציון מקומי – כדאי להשתמש, ובאיזה גודל סביבה (הפרמטר k)?
אין צורך בהסבר. לנוחותכם מופיעות שתי שיטות הניקוי למטה.

<u>הקיפו בעיגול</u>: ממוצע מקומי / חציון מקומי

<u>הקיפו בעיגול</u>: k=1, k=2, k=3, k=4, k=5

ב. כעת נניח כי מעוניינים להפעיל שיטת ניקוי כלשהי עם k=1, אבל אפשר להפעיל אותה יותר מפעם אחת.

באיזו שיטת ניקוי כדאי להשתמש, וכמה הפעלות שלה יש לבצע עד להיעלמות מוחלטת של הלכלוך? אין צורך להסביר.
<u>הקיפו בעיגול</u>: ממוצע מקומי / חציון מקומי

יש צורך ב- _____ הפעלות של שיטת הניקוי עם k=1.

```
def local_medians(A, k=1):
  n,m = A.dim()
  res = copy(A)  # brand new copy of A
  for i in range(k,n-k):
     for j in range(k,m-k):
           res[i,j] = median(items(A[i-k:i+k+1,j-k:j+k+1]))
  return res
```

```
def local_means(A, k=1):
  n,m = A.dim()
  res = copy(A)  # brand new copy of A
  for i in range(k,n-k):
     for j in range(k,m-k):
           res[i,j] = average(items(A[i-k:i+k+1,j-k:j+k+1]))
  return res
```

<u>שאלה 5 (25 נק')</u>

נתונה רשימה L המכילה n מספרים כלשהם. ידוע שקיים אינדקס 0<i<n-1 עבורו מתקיים:

L[0] < L[1] < … < L[i] **and** L[i+1] < … < L[n-1] **and** L[0] > L[n-1]

כלומר הרשימה ממויינת (בסדר עולה) עד אינדקס i כלשהו, וגם החל מאינדקס זה עד הסוף, אבל כל האיברים עד אינדקס i (כולל) גדולים מכל האיברים שאחריו עד הסוף.

אינדקס i כזה ייקרא **נקודת אי-רציפות**.

כיתבו בעמוד הבא פונקציה find_ discontinuity(L) שמקבלת רשימה L כנ"ל, ומוצאת בה את נקודת אי-הרציפות (כלומר מחזירה אינדקס i המקיים את התנאים הנ"ל). מותר להניח חוקיות הקלט.

על הפתרון שלכם להיות יעיל מבחינת סיבוכיות זמן. פתרון בזמן ליניארי יזכה לניקוד זניח.

תנו תחילה הסבר מילולי קצר, שמתאר באופן כללי מאוד את האלגוריתם שלכם.

```
def find_discontinuity(L):
```

שאלה 6 (25 נק')

כזכור, בשיעורי הבית ראינו מימוש פשוט למבנה הנתונים עץ:

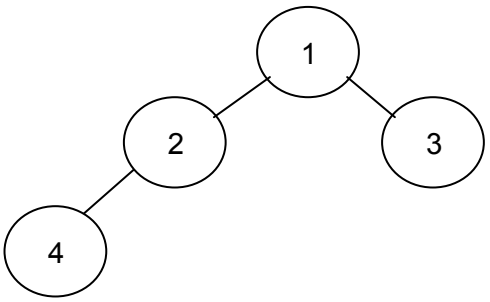
```
class Node():
    def __init__(self, val):
        self.value = val
        self.left = None
        self.right = None

    def addChild(self, side, child):
        if (side == "L"):
            self.left = child
        else:
            self.right = child

    def isLeaf(self):
        return (self.left == None and self.right == None)
```

נגדיר כעת פעולת mirror על עצים: בהינתן עץ T, הפעלת mirror על T משנה את העץ עצמו כך שהתנאים הבאים מתקיימים על העץ המתקבל לאחר הפעולה:

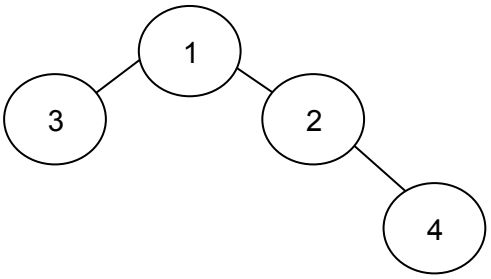
1. כל צומת שהיה בן ימני בעץ T המקורי, הינו כעת בן שמאלי (של אותו אב).
2. כל צומת שהיה בן שמאלי בעץ T המקורי, הינו כעת בן ימני (של אותו אב).
3. שורש העץ לא השתנה.

שימו לב שבעץ ייתכנו צמתים עם שני בנים, בן אחת ו- 0 בנים (עלים).

לדוגמה, אם בידינו העץ הבא:



אזי לאחר הפעלת mirror עליו נקבל את העץ הבא:



א. כתבו בעמוד הבא פונקציה רקורסיבית mirror(T) המקבלת את שורש העץ ומפעילה עליו את הפעולה שתוארה לעיל.

תנו תחילה הסבר מילולי קצר, שמתאר באופן כללי מאוד את האלגוריתם שלכם.

def mirror(T):

שימו לב: סעיף ב' בעמוד הבא.

ב. מהי סיבוכיות זמן הריצה של הפונקציה שלכם, כתלות ב- n, מספר הצמתים בעץ? הצדיקו את תשובתכם ע"י מתן הסבר קצר (הסברים ארוכים יותר יקוצצו בבדיקה):

O( )

סוף!

עמוד ריק