מס' ת"ז_____ מס' מחברת_____

מבחן בקורס מבוא מורחב למדעי המחשב, CS1001.py
עם שינויים קלים בהתאם להבהרות / תיקונים במהלך הבחינה

סמסטר א' 2014-5

מועד א, 8/2/2015

מרצים: פרופ' עמירם יהודאי, דר' דניאל דויטש

מתרגלים: מיכל קליינבורט, אמיר רובינשטיין

משך הבחינה: 3 שעות.

חומר עזר מותר: 2 דפי עזר (דו צדדיים) בגודל A4 כ"א.

- במבחן 11 עמודים – בידקו שכולם בידיכם. בנוסף, בסוף הבחינה ישנו דף נוסף ריק, לשימוש במקרה חירום בלבד.
- יש לכתוב את כל התשובות בטופס הבחינה. המחברת תשמש כטיוטה בלבד ולא תיבדק.
- יש לענות על כל חמש השאלות. מספר הנקודות של כל שאלה הוא 20, אך זה לא משקף בהכרח את רמת הקושי או את הזמן הנדרש לפתרון השאלה.
- בכל השאלות, אלא אם נכתב במפורש אחרת:
  - אם עליכם לכתוב פונקציה, אין צורך לבדוק בה את תקינות הקלט שלה
  - מותר להסתמך על סעיפים קודמים, גם אם לא עניתם עליהם
  - ניתן לצטט טענות שנטענו בהרצאה או בשיעורי התרגול
- אנו ממליצים לא "להיתקע" על אף שאלה, אלא 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.
- יש לכתוב את כל התשובות במקום המוקצב ובכתב קריא. חריגות משמעותיות מהמקום המוקצב, או תשובות הכתובות בכתב קטן מדי, לא ייקראו ; תשובות שדורשות מאמצים רבים להבנתן עלולות לגרור הורדת ציון.

בהצלחה !

טבלת ציונים: (לשימוש הבודקים)

שאלה	ערך	ניקוד
1	20	
2	20	
3	20	
4	20	
5	20	
סה"כ	100	

מס' ת"ז_____ מס' מחברת____

שאלה 1 (20 נק')

בשאלה זו נממש פעולת join בין שתי רשימות המוגדרת כך: הקלט לבעיה הוא 2 רשימות, כאשר כל איבר ברשימות הוא זוג (tuple בגודל 2) של מספרים (לאו דווקא שלמים).

נאמר שזוג p1 מרשימה אחת **מתאים** לזוג p2 מהרשימה השנייה אם האיבר הראשון של p1 זהה לאיבר הראשון של p2.

הפלט הוא רשימה של רביעיות (tuples בגודל 4) שמכילה בדיוק את כל הרביעיות שמתקבלות משרשור של איברי זוג כלשהו p1 מהרשימה הראשונה לאיברי זוג **מתאים** כלשהו p2. שימו לב שזוג מרשימה אחת יכול להתאים ליותר מזוג אחד ברשימה השנייה. **סדר הרביעיות ברשימת הפלט יכול להיות כלשהו**.

לדוגמא אם רשימות הקלט הן

```
[(1,2), (3,7), (2.5, 15), (3,4)]
```

```
[(2,2), (3,10), (1,4)]
```

אזי רשימת הפלט היא (אין חשיבות לסדר הרביעיות ברשימה):

```
[(1,2,1,4), (3,7,3,10), (3,4,3,10)]
```

א. ממשו את הפונקציה join בסיבוכיות O(m*n) כאשר m ו-n הם אורכי רשימות הקלט.

```
def join(L1, L2):
```

ב. נניח כעת שתי הנחות לגבי איברי הרשימות:

- לכל זוג (בשתי הרשימות) מתקיים שהאיבר הראשון בזוג הוא מספר גדול או שווה ל- 0 וקטן ממש מ- k (k הוא קלט נוסף לפונקציה), שהוא שלם **או** "חצי", כלומר x+0.5 ל- x שלם כלשהו.
- לכל ערך i, יש לכל היותר 10 זוגות שונים (בשתי הרשימות גם יחד) שמקיימים שהאיבר הראשון שלהם הוא i.

ממשו בעמוד הבא את פונקציית join בסיבוכיות **מקרה גרוע ביותר** O(k) (שימו לב שבהינתן ההנחות, m ו- n לינארים ב- k).

הנחייה: אין להשתמש ב- dictionary או set של פייתון.

```
def join(L1,L2,k):
    
    for p in L1:
        
    for p in L2:
        
    
```

שאלה 2 (20 נק')

השאלה עוסקת בעצי האפמן.

בכוכב מאדים התקבלה החלטה להחליף את הטכנולוגיה הספרתית הבינארית (המתבססת על שני ערכים 0 ו- 1) בטכנולוגיה ספרתית טרינארית: מידע ייוצג באמצעות שלושה ערכים שיסומנו 0, 1 ו- 2. ברצוננו להתאים את האלגוריתם לבניית עצי האפמן לשיטה זו. בכל שלב האלגוריתם יאחד שלושה צמתים (במקום שניים) בעלי השכיחויות המינימליות. במקרה של יותר ממינימום אחד הבחירה ביניהם שרירותית. בנוסף, בחירת הסדר בין הבנים של צומת (שמאל, מרכז, ימין) היא שרירותית ואין לה חשיבות בשאלה זו. מכאן שלכל צומת בעץ שאיננו עלה יהיו כעת 3 בנים. יוצא דופן אחד הוא שורש העץ: אם מספר התווים השונים זוגי, לשורש יהיו רק שני בנים.

המשך השאלה בעמוד הבא.

א. נתון קורפוס המכיל את k התווים $\{x_1, x_2, x_3 \ldots x_k\}$, כאשר k מספר אי-זוגי כלשהו ומתקיים $k \geq 3$. נתון כי התפלגות התווים בקורפוס מקיימת את הכלל הבא: תדירות התו x_i היא 2^{i-1}. ציירו את המבנה הכללי של עץ האפמן (טרינארי) המתקבל, וציינו את השכיחויות **בעלים** (אין צורך לתאר את שלבי בניית העץ).

ב. נתון קורפוס בעל 9 תווים שתדירויותיהם הן: $a_1 < a_2 < \cdots < a_8 < a_9$ ומתקיים בנוסף ש: $a_9 < 3a_1$. ציירו עץ האפמן טרינארי שמתקבל מקורפוס זה.

ג. נתון קורפוס שמכיל 80 תווים שונים, ביניהם התו A. שכיחות התו A הינה k, ושכיחות כל תו אחר (שאינו A) הינה 1. בונים מקורפוס זה עץ האפמן טרינארי.

מהו k המינימלי שמבטיח שהתו A יקבל קידוד באורך ביט אחד בלבד? _____

מהו אורך הקידוד המקסימלי לתו בקורפוס במקרה זה? _____

שאלה 3 (20 נק')

שאלה זאת עוסקת במציאת מספרים ראשוניים בעזרת אלגוריתם הנפה (מסננת). האלגוריתם מוצא את המספרים הראשוניים הקטנים מ **n** (שניתן כקלט) באופן הבא. בתחילה מיוצגים כל השלמים מ **2** עד **n-1** כמועמדים, כלומר מספרים שייתכן שהם ראשוניים. בשלב ראשון מזוהה 2, המספר הראשון בין המועמדים, כראשוני, וכל כפולותיו **4,6,8,10,...** מוסרות מרשימת המועמדים. בהמשך, חוזרים על התהליך -- המספר הבא מבין המועמדים מזוהה כראשוני, וכל כפולותיו מוסרות מרשימת המועמדים.

נתייחס למספר גירסאות של האלגוריתם.

<u>הנחייה</u>: בכל הסעיפים של שאלה זו אין להשתמש בפונקציות שנלמדו לבדיקת ראשוניות, כגון `is_prime`.

א. נתונה הפונקציה **`sieve1`** שמקבלת כפרמטר מספר שלם **n** ומחזירה את רשימת המספרים הראשוניים הקטנים מ **n**. לדוגמא:

```
>>> sieve1(20)
[2, 3, 5, 7, 11, 13, 17, 19]
```

```
1. def sieve1(n):
2.   """ returns a list of all prime numbers smaller than n"""
3.    candidates = [m for m in range(2,n)]
4.    primes = []
5.    while candidates !=[]:
6.          k = candidates.pop(0)
7.          primes.append(k)
8.          j = 0
9.          while j < len(candidates):
10.               if candidates[j]% k ==0:
11.                    candidates.pop(j)
12.               else:
13.                    j += 1
14.   return primes
```

מה מספר פעולות **`pop`** ו- **`append`** שמתבצעות ע"י **`sieve1`** כפונקציה של **n** ו/או של **p** (מספר המספרים הראשוניים הקטנים מ **n**)? תנו ביטויים מדויקים ככל האפשר, ו<u>לא</u> ביטויי סדר גודל אסימפטוטי $O(\ldots)$.

1. מספר פעולות **`pop`**: _____
2. מספר פעולות **`append`**: _____

ב. **sieve1Iter** היא פונקצית גנרטור, שמקבלת כפרמטר מספר שלם **n** ומחזירה גנרטור שיוצר את רשימת המספרים הראשוניים הקטנים מ **n**. לדוגמא:

```
>>> g = sieve1Iter(20)
>>> list(g)      #convert the generator elements into a list
[2, 3, 5, 7, 11, 13, 17, 19]
```

ציינו **בפירוט** את השינויים הנדרשים בפונקציה **sieve1** כדי לממש את **sieve1Iter**: ציינו אם יש שורות שצריך להשמיט ואילו, שורות שצריך לשנות ומה השינוי, ו/או שורות חדשות שצריך להוסיף

_____

_____

_____

_____

ג. הפונקציה **sieve2** היא מימוש שונה **של אותו האלגוריתם,** שמשתמש ברשימה **primes** של ערכים בוליאניים במקום שתי הרשימות ב **sieve1**.

הנחיה: הפונקציה אינה מוסיפה או משמיטה אברים מהרשימה לאחר האתחול שלה, אבל היא משנה את ערכם, כך שבסוף ריצת הפונקציה מתקיים ש- **j** ראשוני אם ורק אם **primes[j]== True**.

לדוגמא:

```
>>> sieve2(20)
[2, 3, 5, 7, 11, 13, 17, 19]
```

השלימו את הפונקציה:

```
def sieve2(n):
      """ returns a list of all prime numbers smaller than n """
      primes = [True for m in range(n)]
      primes[0]=False
      primes[1]=False
      for k in _____:
          if primes[k]:     # k is prime
```

```
      return [_____]
```

ד. **allPrimesIter** היא פונקצית גנרטור ללא פרמטרים, שמחזירה גנרטור שיוצר את כל המספרים הראשוניים. לדוגמא:

```
>>> g = allPrimesIter()
>>> [next(g) for i in range(10)]
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

הפונקציה שומרת ברשימה **primes** את כל המספרים הראשוניים שהגנרטור כבר יצר. הפונקציה לוקחת בכל שלב את המספר השלם הבא מגנרטור בשם **naturals** שמייצר את כל המספרים הטבעיים החל ב-1 (גנרטור זה נתון לכם ואין צורך לממשו), ובודקת אם הוא מתחלק באחד הראשוניים הקודמים, כדי לקבוע אם הוא ראשוני. הבדיקה נעשית בעזרת פונקצית העזר **does_not_divide(m,primes)** הנתונה:

```
def does_not_divide (m,primes):
        for k in primes:
            if m % k == 0:
                return False
        return True
```

השלימו את הפונקציה **allPrimesIter**:

```
def allPrimesIter():
    """ a generator for all prime numbers"""
    primes = []
    nat = naturals()
    n = next(nat) # skip 1
    while True:
```

שאלה 4 (20 נק')

השאלה עוסקת באחת הגרסאות לקוד אינדקס לאיתור ותיקון שגיאות אותה ראינו בכיתה.

<u>תזכורת</u>: אנו רוצים לשלוח הודעה $msg = b_1b_2...b_k$, כאשר $b_i \in \{0,1\}$ ו- $k = 2^m-1$ עבור m טבעי כלשהו. נסתכל על הביטים ה"דולקים" ($b_i=1$, עבור $1 \le i \le k$), ונחשב xor בין האינדקסים שלהם (בכתיב בינארי). נסמן את תוצאת החישוב ב- EC. אם אין בכלל ביטים "דולקים" אז $EC=000$. נוסיף ל- msg את EC פעמיים (נסמנם $EC1$ ו- $EC2$). את השדר כולו נסמן $trans$.

דוגמה עבור $m=3$, אותה גם ראינו בתרגול:

```
msg        0110110          (k = 2^3-1)
indices    1234567

        2 = 010 xor
        3 = 011 xor
        5 = 101 xor
        6 = 110
       EC= 010

trans      0110110 010 010
            msg    EC1 EC2
```

כזכור, המרחק המינימלי של הקוד הזה הוא $d=3$, ולכן ניתן לתקן שגיאה אחת.

להלן האלגוריתם שכתבנו בתרגול לשחזור ההודעה msg, בהנחה שנפלה לכל היותר שגיאה אחת:

```
Decode (trans = msg+EC1+EC2):
1. compute EC ' from msg (as done by the sender)
2. if EC ' = EC1 or EC ' = EC2   # if both True, then 0 errors
3.    return msg                 # no error in ,msg
4. else:                         # single error in msg
5.    i = EC ' xor EC1
6.    return msg with the i'th index flipped
```

שימו לב כי בשורה 5 הנחנו שהחישוב נעשה מול $EC1$.

מס' ת"ז_____ מס' מחברת____

הטבלה הבאה מתייחסת למקרה שבו נפלו שתי שגיאות בשדר. סמנו V במשבצות המייצגות מצבים אפשריים (את השאר השאירו ריקות). מצב אפשרי הוא כל מצב שההסתברות שיתרחש גדולה מאפס.

נפלו שתי שגיאות בשדר, אחת ב- ***msg*** והשנייה ב- ***EC1***	נפלו שתי שגיאות בשדר, אחת ב- ***msg*** והשנייה ב- ***EC2***	
1	5	האלגוריתם יחזיר את ***msg*** הנכון (המקורי) בשורה 3
2	6	האלגוריתם יחזיר את ***msg*** הנכון (המקורי) בשורה 6
3	7	האלגוריתם יחזיר ***msg*** לא נכון בשורה 3
4	8	האלגוריתם יחזיר ***msg*** לא נכון בשורה 6

לכל משבצת שסימנתם בטבלה למעלה, ציינו בטבלה הבאה, בשורה המסומנת עם מספר המשבצת, דוגמה למיקומים בהם נפלו שתי השגיאות בהודעה 0110110, כך שהמצב המתואר מתרחש. (סמנו X בשני המקומות בהם נפלו השגיאות, עבור כל מצב אפשרי):

משבצת מס'	***msg***							***EC1***			***EC2***		
	0	1	1	0	1	1	0	0	1	0	0	1	0
1													
2													
3													
4													
5													
6													
7													
8													

שאלה 5 (20 נק')

להלן מימוש חלקי לשתי מחלקות: המחלקה **File**, המייצגת קובץ, והמחלקה **Folder** המייצגת תיקיה במערכת הקבצים. להזכירכם, תיקיה יכולה להכיל קבצים או תיקיות פנימיות.

שימו לב כי לשם פשטות, במימוש שיוצג כאן אין ייצוג ממשי לתוכן הקובץ, אלא רק למבנה התיקיות ומאפייני הקבצים.

בכל סעיפי השאלה אין להוסיף שדות או לשנות מתודות קיימות שכבר מומשו עבורכם.

```
class File:
    def __init__(self, name, size, ftype):
        self.size = size
        self.name = name
        self.ftype = ftype

    def __repr__(self):
        return self.name + "." + self.ftype

class Folder:
    def __init__(self, name):
        self.items = []
        self.name = name

    def add_item(self, item):
        assert isinstance(item, (File, Folder))
        self.items.append(item)

    def __repr__(self):
        … #some implementation
```

הניחו שהמתודה **__repr__** של המחלקה **Folder** כבר מומשה עבורכם, כך שהפלט שלה הינו בהתאם לדוגמאות ההרצה הבאות. סדר הופעת הקבצים והתיקיות הפנימיות הינו בהתאם לסדר הופעתם ברשימה שמיוצגת ע"י השדה **items**.

```
>>> docs =  Folder("my_docs")
>>> songs = Folder("my_songs")
>>> songs.add_item(File("derech_hashalom", 50, "mp3"))
>>> docs.add_item(File("hw5", 200, "pdf"))
>>> docs.add_item(File("hw5", 100, "py"))
>>> docs.add_item(songs)
>>> docs
{hw5.pdf, hw5.py, {derech_hashalom.mp3}}
>>> songs
{derech_hashalom.mp3}
```

א. ממשו בעמוד הבא את המתודה **all_files** של המחלקה **Folder**. מתודה זו תחזיר רשימה (**list** של פייתון) המכילה את כל הקבצים (אובייקטים מהמחלקה **File**) שמוכלים בתיקיה **self**, ישירות או בתיקיות פנימיות. המימוש חייב להיות רקורסיבי. דוגמת הרצה:

```
>>> docs.all_files()
[hw5.pdf, hw5.py, derech_hashalom.mp3]
```

```
def all_files(self):
     files = []
     for item in self.items:
          
    return files
```

ב. ממשו את המתודה **all_files_by_size** של המחלקה **Folder**. מתודה זו תחזיר את אותה רשימה כמו בסעיף הקודם, אך **הפעם רשימה זו תהיה ממוינת (בסדר עולה) על פי גודל הקובץ** (השדה **size** במחלקה **File**). יש להשלים שורה אחת בודדת בלבד. למשל:

```
>>> docs.all_files_by_size()
[derech_hashalom.mp3, hw5.py, hw5.pdf]
```

```
def all_files_by_size(self):

  return  _____
```

ג. ממשו את המתודה **delete_file** של המחלקה **Folder.** מתודה זו תמחק את הקובץ (אובייקט מהמחלקה **File)** ששמו **filename,** אם הוא מוכל בתיקיה **self**, ישירות או בתיקיות פנימיות. אם קובץ בשם זה לא מוכל ב- **self** המתודה לא תעשה דבר. אם קיימים שני קבצים או יותר בעל שם זה, המתודה תמחק אחד מהם שרירותית. למשל:

```
>>> docs.delete_file("hw5")
>>> docs
{hw5.py, {derech_hashalom.mp3}}
>>> docs.delete_file("derech_hashalom")
>>> docs
{hw5.py}
```

שימו לב כי עליכם למחוק גם תיקיה פנימית שהתרוקנה מקבצים, כמו בהרצה האחרונה.

```
def delete_file(self, filename):
     for item in self.items:
          
```

דף נוסף למקרה הצורך