מס' ת"ז_____ מס' מחברת_____

מבחן בקורס מבוא מורחב למדעי המחשב, CS1001.py

סמסטר א' 2014

מועד ב, 14/3/2014

מרצים: פרופ' עמירם יהודאי, דר' דניאל דויטש

מתרגלים: מיכל קליינבורט, אמיר רובינשטיין

משך הבחינה: 3 שעות.

חומר עזר מותר: 4 עמודים בגודל A4 כ"א.

- במבחן 11 עמודים – בידקו שכולם בידיכם.
- יש לכתוב את כל התשובות בטופס הבחינה. המחברת תשמש כטיוטה בלבד ולא תיבדק.
- יש לענות על כל חמש השאלות. מספר הנקודות של שאלה אינו 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
  - אין צורך להוסיף הערות (#comments) בגוף הקוד שאתם כותבים
- 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.

טבלת ציונים: (לשימוש הבודקים)

שאלה	ניקוד
1	
2	
3	
4	
5	
סה"כ	

בהצלחה !

שאלה 1 (25 נק')

בשאלה זו אנו עוסקים ברשימות בגודל n המכילות מספרים כלשהם. המספרים לא בהכרח שונים זה מזה.

א. השלימו את הפונקציה counts(lst) הבאה, המקבלת רשימה lst כנ"ל, ומחזירה מילון (dictionary של פייתון), עם איברים num **:** cnt, כך שהמספר num נמצא ב- lst בדיוק cnt פעמים.

דוגמת הרצה:

```
>>> counts([1.3, 2, 1, 1, 2, 1])
{1: 3, 2: 2, 1.3: 1}
```

דרישת סיבוכיות: על הפונקציה לרוץ בסיבוכיות זמן O(n) בממוצע (תוחלת). אנו מניחים כי תוחלת זמן הריצה של כל פעולת הכנסה / חיפוש במילון של פייתון היא O(1).

```
def counts(lst):
    d = {}









    return d
```

ב. להלן הפונקציה mergesort שראינו בכיתה (כולל הפונקציה merge), עם השינוי הבא:

לפונקציה מועבר פרמטר נוסף key, בדומה לזה שאפשר להעביר לפונקציית המיון sorted של פייתון. פרמטר זה יגדיר מהו האופן בו יושוו איברים בעת המיון. השינויים מודגשים בקו.

השלימו במקומות המסומנים בקו את הפונקציה merge שבה משתמשת mergesort.

דוגמת הרצה:

```
>>> mergesort([3,5,2,1,4], key = lambda n:-n)
[5, 4, 3, 2, 1]
```

```
def mergesort(lst, key = lambda x:x):
    """ recursive mergesort """
    n=len(lst)
    if n <= 1:
        return lst
    else:
        return merge(mergesort(lst[0:n//2], key),
                     mergesort(lst[n//2:n], key), key)
                #two recursive calls

def merge(lst1, lst2,_____):
    """ merging two ordered lists using
        the two pointer algorithm """
    n1 = len(lst1)
    n2 = len(lst2)
    lst3 = [0 for i in range(n1 + n2)]  # allocates a new list
    i = j = k = 0  # simultaneous assignment
    while i < n1 and j < n2:
        if _____:
            lst3[k] = lst1[i]
            i = i +1
        else:
            lst3[k] = lst2[j]
            j = j + 1
        k = k + 1  # incremented at each iteration
    lst3[k:] = lst1[i:] + lst2[j:]  # append remaining elements
    return lst3
```

ג. השלימו את הפונקציה sorted_counts(lst) הבאה, המקבלת רשימה lst כפי שמוגדר בתחילת השאלה, ומחזירה רשימה של tuples מהצורה (num : cnt), כאשר המספר num נמצא ב- lst בדיוק cnt פעמים, והרשימה המוחזרת ממוינת על פי num. השתמשו בקריאה ל- mergesort הנ"ל, וכן בפונקציה count מסעיף א'. יש להשלים שורה אחת בלבד.

דוגמת הרצה:

```
>>> sorted_counts([1.3, 2, 1, 1, 2, 1])
[(1, 3), (1.3, 1), (2, 2)]
```

הכוונה נוספת:

```
>>> list({1: 3, 2: 2, 1.3: 1}.items())
[(1, 3), (2, 2), (1.3, 1)]
```

```
def sorted_counts(lst):

    return _____
```

ד. ממשו את הפונקציה sort(lst), שמקבלת רשימה lst כפי שמוגדר בתחילת השאלה, ומחזירה רשימה ממוינת של אותם מספרים בדיוק.

יש להשתמש בפונקציה sorted_counts מסעיף ג'. אין להשתמש בשום פונקצית מיון אחרת שלמדנו (גם לא sorted של פייתון).

דוגמת הרצה:

```
>>> sort([1.3, 2, 1, 1, 2, 1])
[1, 1, 1, 1.3, 2, 2]
```

```
def sort(lst):
```

<u>שאלה 2 (20 נק')</u>

השאלה עוסקת בטבלאות hash שבהן מספרים טבעיים, וטיפול בהתנגשויות נעשה בשיטה של open addressing שתוסבר בהמשך. גודל הטבלה יסומן ב- m, ואילו מספר האיברים שנמצאים בטבלה יסומן ב- n. יצירת טבלה חדשה ריקה תתבצע כך:

```
def hash_table(m):
   return [None for i in range(m)]   #initial hash table, m empty entries
```

כאשר נרצה להכניס איבר k לטבלה, נשתמש בפונקצית ה- hash הבאה: h(k) = k%m. כמו כן נשתמש במנגנון הבא לטיפול בהתנגשויות: אם התא אליו מנסים להכניס תפוס כבר, נבדוק האם התא ה- (k+1)%m פנוי, וכך הלאה. בשלב ה- i (i≥0) נבדוק אם התא ה- (k+i)%m פנוי להכנסה. אם לאחר m ניסיונות כנ"ל לא נמצא תא פנוי – יודפס "Table is full".

דוגמאות הרצה:

```
>>> ht = hash_table(3)
>>> ht
[None, None, None]
>>> insert(ht,5)
>>> ht
[None, None, 5]
>>> insert(ht,2)
>>> ht
[2, None, 5]
>>> insert(ht,8)
>>> ht
[2, 8, 5]
>>> insert(ht,1)
Table is full
```

א. ממשו את הפונקציה insert(table, k) שמכניסה את המספר הטבעי k לטבלה table בהתאם למנגנון הנ"ל.

```
def insert(table, k):
```

ב. מהי סיבוכיות הזמן של הכנסת מספר לטבלה, שבה התנגשויות מטופלות כנ"ל, במקרה הגרוע, כפונקציה של m ו/או n? תנו תשובה במונחי $O(\ldots)$, כאשר החסם העליון יהיה נמוך ככל שתוכלו. למשל, עבור אלגוריתם שזמן ריצתו $O(l)$, החסם $O(l^2)$ איננו נמוך מספיק.

O(_____)

ג. מכניסים לטבלה <u>ריקה</u> את אותו המספר פעם אחר פעם, m פעמים. מהי סיבוכיות הזמן הכוללת של סדרת ההכנסות הזו? תנו תשובה במונחי $O(\ldots)$, כאשר החסם העליון יהיה נמוך ככל שתוכלו.

O(_____)

ד. להלן טבלה בגודל m=6 אליה הוכנסו 5 איברים: [6, 1, 2, 7, 8, None] .

מי מהבאים יכול להיות סדר הכנסת האיברים (משמאל לימין): סמנו את כל התשובות הנכונות.

1. 1,6,7,8,2
2. 1,6,7,2,8
3. 2,1,6,8,7
4. 2,1,6,7,8
5. אף אחת מהתשובות הנ"ל לא אפשרית.

ה. הועלתה הצעה לבדוק בשלב ה- i את התא ה- (k+2i)%m.

נניח כי בטבלה מסוימת יש בדיוק תא פנוי אחד (m-1 תפוסים כבר). מנסים להכניס לטבלה הזו איבר נוסף, בהתאם להצעה הנ"ל. סמנו את התשובה הנכונה, ונמקו בקצרה:

1. בהכרח יימצא המקום הפנוי והאיבר החדש יוכנס בהצלחה
2. בהכרח לא יימצא המקום הפנוי ויודפס Table is full.
3. ייתכן שיימצא המקום הפנוי וייתכן שלא יימצא.

נימוק:

<u>שאלה 3 (20 נק')</u>

שאלה זאת עוסקת במבני רשימות, שמוגדרים רקורסיבית כך:

<u>הגדרה</u>: מבנה רשימה (list structure) הוא מספר שלם או רשימה של 0 או יותר מבני רשימות.
דוגמאות יינתנו בהמשך (הארגומנטים של הפונקציות).

א. השלימו בעמוד הבא את הפונקציה flatten שמקבלת מבנה רשימה, ומשטחת אותו לרשימה פשוטה, כלומר מחזירה רשימה שמכילה את כל המספרים השלמים שהופיעו במבנה לפי סדר הופעתם. לדוגמא:

```
>>> flatten(1)
[1]
>>> flatten([])
[]
>>> flatten([[[1],[2,[1]]]])
[1, 2, 1]
```

```
def flatten(lst):
    if type(lst) == int:
        return _____
    elif lst==[]:
        return _____
    else:
        return _____
```

ב. גודל של מבנה רשימות הוא מספר המספרים שמופיעים בו בתוספת מספר זוגות הסוגריים.
כיתבו פונקציה רקורסיבית size שמקבלת מבנה רשימה, ומחזירה את גודל המבנה. לדוגמא:

```
>>> size([[[14],[23,[14]]]])      # 3 מספרים ו 5 זוגות סוגריים
8
>>> size([])
1
>>> size(3)
1
```

```
def size(lst):
```

ג. כיתבו (בעמוד הבא) פונקציה רקורסיבית transform שמקבלת פונקציה f (שמקבלת מספר שלם יחיד כקלט ומחזירה מספר שלם כפלט), ומבנה רשימה lst, ומחזירה מבנה רשימה חדש, שהמבנה שלו זהה לזה של lst, והמספרים שבו הם תוצאה של הפעלת f על המספרים ב lst בהתאמה. לדוגמא:

```
>>> transform(lambda x: 2*x+1, [[1], [2, []],3])
[[3], [5, []], 7]
```

```
def transform(f,lst):
```

שאלה 4 (20 נק')

שימו לב: **בכל הסעיפים בשאלה זו אסור להשתמש בלולאות for או while**.

א. השלימו בעמוד הבא את הפונקציה הרקורסיבית aggregate שמקבלת כקלט:

- רשימה (אולי ריקה) של פונקציות f1,…fn, שכל אחת מהם מקבלת מספר יחיד כקלט ומחזירה מספר כפלט
- פונקציית agg שמקבלת כקלט שני מספרים ומחזירה מספר יחיד (באופן כללי יש חשיבות לסדר הארגומנטים של agg)
- מספר neutral שהוא איבר נייטרלי מימין של agg, כלומר מקיים שלכל x, agg(x,neutral)=x

aggregate מחזירה פונקציה f שמחשבת עבור כל קלט x את תוצאת ההפעלה החוזרת ונשנית של האגרגציה, על תוצאות הפעלת הפונקציות f1,…fn על x.

כלומר לכל x מתקיים f(x)=agg(f1(x),agg(f2(x),agg(f3(x),agg(….agg(fn(x),neutral))))).

למשל, אם agg היא פונקצית החיסור, neutral=0 ורשימת הקלט כוללת 3 פונקציות f1,f2,f3 אזי מתקיים ש: f(x) = f1(x) - (f2(x)-(f3(x)-0)).

אם הרשימה ריקה, הפלט של aggregate יכול להיות כלשהו, לפי החלטתכם.

דוגמא:

```
>>> funclist = [lambda x: x**2, lambda x: x**3, lambda x:x]
>>> f = aggregate(funclist , lambda x,y: x-y,0)
>>> f(3)
-15
```

```
def aggregate(funclist, agg, neutral):
    if _____ :
        return _____
    else:
```

ב. האם הפונקציה שכתבתם בסעיף א' מבצעת רקורסית זנב (tail recursion)? הסבירו בקצרה:

_____

_____

_____

ג. בסעיף זה נכתוב גירסה אחרת של הפונקציה aggregate. השלימו את הקוד כך שסיבוכיות זמן הריצה של פונקציית aggregate תהיה O(n) תחת ההנחות שלעיל.

```
def aggregate_helper(funclist ,agg, neutral, mystery):
    if _____ :
        return _____
    else:

def aggregate(funclist ,agg, neutral):
    return aggregate_helper(funclist, agg, neutral, 0)
```

שאלה 5 (15 נק')

בשאלה זו ננתח את מספר פעולות הכפל שמתבצעות בחישוב a^b.

הפונקציה הבאה (שראינו בכיתה) מקבלת שני פרמטרים a,b ומחשבת את a^b.

```
def power(a,b):
    """ computes a**b using iterated squaring """
    result=1
    while b>0: # b is nonzero
        if b % 2 == 1: #  b is odd
            result = result*a   # (פעולת כפל אחת)
        a=a*a  # (פעולת כפל אחת)
        b = b//2
    return result
```

א. בהינתן a, b שמיוצגים ע"י n, m ביטים בהתאמה, מהו המספר (המדויק) הקטן ביותר האפשרי של פעולות הכפל שמתבצעות ע"י power? התשובה צריכה להינתן כפונקציה של n ו/או m.
הסבירו וציינו מה צריך להתקיים עבור a ו/או b כדי שנקבל מספר כזה של פעולות.

_____

_____

_____

_____

_____

ב. בהינתן a, b שמיוצגים ע"י n, m ביטים בהתאמה, מהו המספר (המדויק) הגדול ביותר האפשרי של פעולות הכפל שמתבצעות ע"י power? התשובה צריכה להינתן כפונקציה של n ו/או m.
הסבירו וציינו מה צריך להתקיים עבור a ו/או b כדי שנקבל מספר כזה של פעולות.

_____

_____

_____

_____

_____

ג. השלימו את הפונקציה power_new שבהינתן a, b מחשבת את a^b באותה סיבוכיות זמן כמו הפונקציה power שלמדנו.

תזכורת:

```
>>> x = 14
>>> bin(x)
'0b1110'
>>> type(bin(x))
<class 'str'>
```

```
def power_new(a,b):

    """ computes a**b using iterated squaring """

    result=1

    b_bin = bin(b)[2:]

    reverse_b_bin = b_bin[: :-1]

    for  bit in reverse_b_bin:

      _____

      _____

      _____

    return result
```

סוף!