מס' ת"ז_____ מס' מחברת_____

מבחן בקורס מבוא מורחב למדעי המחשב, CS1001.py
עם תיקונים קלים והבהרות שניתנו במהלך המבחן

ביה"ס למדעי המחשב, אונ' תל אביב

סמסטר ב' 2015-16, מועד ב, 18/8/2016

מרצים: ד"ר דניאל דויטש, אמיר רובינשטיין

מתרגלות: יעל ברן, מיכל קליינבורט, אמיר גלעד

משך הבחינה: 3 שעות.

חומר עזר מותר: 2 דפי עזר (דו צדדיים) או 4 עמודים (חד צדדיים) בגודל A4 כל אחד.

- במבחן 12 עמודים מודפסים – בידקו שכולם בידיכם. בנוסף, בסוף הבחינה ישנו דף נוסף, ריק, לשימוש ב"מקרה חירום" בלבד.
- יש לכתוב את כל התשובות בטופס הבחינה. המחברת תשמש כטיוטה בלבד ולא תיבדק.
- יש לענות על כל השאלות.
- בכל השאלות, אלא אם נכתב במפורש אחרת:
  - אם עליכם לכתוב פונקציה, אין צורך לבדוק את תקינות הקלט שלה.
  - מותר להסתמך על סעיפים קודמים, גם אם לא עניתם עליהם.
  - ניתן לצטט טענות שנטענו בהרצאה או בשיעורי התרגול. במקרה זה יש לכתוב "בהרצאה/תרגול ראינו כי...".
- אנו ממליצים לא "להיתקע" על אף שאלה, אלא להמשיך לשאלות אחרות ולחזור לשאלה אח"כ.
- בכל סעיף ניתן לכתוב "איני יודע/ת" ולא לכתוב שום טקסט נוסף. במקרה זה יינתן 20% מציון הסעיף (מעוגל כלפי מטה).
- יש לכתוב את כל התשובות במקום המוקצב ובכתב קריא. תשובות ובהן חריגות משמעותיות מהמקום המוקצב, או תשובות הכתובות בכתב קטן מדי, לא ייקראו ולא יקבלו ניקוד, או שיקבלו ניקוד חלקי בלבד. תשובות שדורשות מאמצים רבים להבנתן גם כן 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')

להלן תיאור של אלגוריתם מיון חדש, שלא נלמד בכיתה – מיון דרגה (rankSort), שיוסבר מייד. בשאלה זו נשתמש במיון הזה לצורך מיון של אובייקטים ממחלקה (class) Rectangle1 שראיתם בתרגול. מחלקה זו מייצגת מלבנים במישור, והמיון צריך להתבצע לפי שטח המלבן. כל שצריך לדעת על המחלקה לצורך שאלה זו, הוא שיש לה מתודה בשם area, שמחזירה את שטח המלבן.

בהינתן רשימה L המכילה n אובייקטים מטיפוס Rectangle1, האלגוריתם ירוץ על איברי הרשימה לפי הסדר, ועבור כל איבר e יחשב כמה איברים יש ברשימה שקטנים ממש מ- e (כלומר בעלי שטח קטן ממש משטחו של המלבן e). ערך זה ייקרא הדרגה (rank) של e. המיקום של e ברשימת הפלט הוא בדיוק הדרגה שלו. למשל, עבור רשימת המלבנים L = [r1, r2, r3] ששטחם בהתאמה 7,4,9, הדרגה של r1 היא 1, של r2 היא 0, ושל r3 היא 2. האלגוריתם לא משנה את הקלט, ומחזיר רשימה חדשה, של מלבנים, ממויינים לפי שטחם.

א. להלן מימוש לאלגוריתם זה, אך הוא לא עובד נכון בכל המקרים:

```
def rankSort_wrong(L):
    out = [None]*len(L)
    for i in range(len(L)):
        rank = len([e for e in L if e.area() < L[i].area()])
        out[rank] = L[i]
    return out
```

i) ידוע כי המיון הנ"ל הופעל על רשימה L2 = [r1, r2, r3, r4], והחזיר פלט שגוי. רישמו דוגמה לערכי השטחים של המלבנים עבורה מצב כזה יכול לקרות.

מלבן	r4	r3	r2	r1
שטח				

ציינו מהו הפלט של האלגוריתם במקרה שציינתם. לצורך כך, הניחו כי המתודה __repr__ של המחלקה Rectangle1 פשוט מדפיסה את שטח המלבן:

```
>>> rankSort_wrong([r1, r2, r3, r4])
[ _____, _____, _____, _____ ]
```

ii) הסבירו במילים בקצרה מה שורש הבעיה במימוש הנ"ל.

_____

_____

ב. השלימו את המימוש הבא, כך שיעבוד נכון בכל המקרים. על הפתרון שלכם לרוץ בסיבוכיות זמן של $O(n^2)$ במקרה הגרוע עבור רשימה בגודל n.

```
def rankSort(L):
    out = [None]*len(L)
    for i in range(len(L)):
        rank = len([e for e in L if e.area() < L[i].area()])









    return out
```

הסבירו בקצרה מדוע הפתרון שלכם רץ בסיבוכיות $O(n^2)$ במקרה הגרוע.

_____

_____

_____

מס' ת"ז_____ מס' מחברת____

שאלה 2 (20 נק')

חמדני הוא בעל דירה בשטח n מ״ר בתל אביב, והוא מעוניין לחלק את דירתו כך שתניב לו רווח מירבי. חמדני חישב את המחיר בו יוכל למכור כל חלק של הדירה בהתאם לגודלה במ״ר. כלומר לכל גודל דירה $i = 1,2,\ldots,n$ במ״ר יש מחיר מסויים $\boldsymbol{value_i}$ (כל תת-דירה חייבת להיות בשטח מ״ר שלם, למשל לא ניתן לחלק ל- 1.5 מ״ר).

לדוגמה, להלן כל הדרכים לחלק דירה בגודל 4 מ״ר לדירות בגדלים 1, 2, 3, 4 עבור המחירים [1, 5 ,8 , 9]:

- מחיר כל הדירה בשטח 4 מ״ר: 9
- מחיר הדירה לאחר חלוקה לשטח 1,1,1 מ״ר: 4
- מחיר הדירה לאחר חלוקה לשטח 1,2 מ״ר: 7
- מחיר הדירה לאחר חלוקה לשטח 2,2 מ״ר: 10
- מחיר הדירה לאחר חלוקה לשטח 1,3 מ״ר: 9

א. עזרו לחמדני למכור את דירתו במחיר המירבי, והשלימו את הפונקציה הרקורסיבית הבאה אשר מחשבת את המחיר המירבי עבור דירה בשטח *size* ורשימת מחירים $value$ ($value[i]$ הוא המחיר עבור דירה בשטח i+1). הניחו כי הקלט תקין, ובפרט כי אורך הרשימה $value$ אינו קטן מ- *size*.

למשל:

```
>>> profit([2, 3, 7, 8, 9], 5)
11
```

```
def profit(value, size):
    if _____:
        return _____
    _____
    for i in range(_____):
        _____
    return _____
```

ב. שנו את הפונקציה כך שתשתמש בממואיזציה:

```
def profit_mem(value, size, d):
    if _____:
        return _____
    _____:
    _____
    _____
    for i in range(_____):
    _____
    _____

    return _____
```

ג. ציירו את עץ הרקורסיה של הפונקציה `profit_mem` עבור size = 5 כאשר בכל צומת רישמו את ערכו של size בקריאה הנוכחית (אין צורך לרשום את $value$ בעץ).

שאלה 3 (20 נק')

בשאלה זו נממש גרסה נוספת של אלגוריתם הפריטה הרקורסיבי שראינו בתרגול, הפעם באמצעות גנרטורים.
להלן הקוד מהתרגול:

```
def countChange(amount, coins):
   if (amount == 0):
       return 1
   elif (amount < 0 or coins == []):
       return 0
   else:
       return (countChange(amount, coins[:-1]) +      \
               countChange(amount - coins[-1], coins))
```

כמו בתרגול, אנו מניחים כי amount הוא מספר שלם לא-שלילי, וכי coins היא רשימה (אולי ריקה) של מספרים שלמים חיוביים.

א. השלימו את פונקצית הגנרטור הבאה, אשר מייצרת את כל הפריטות האפשריות של amount באמצעות coins. כל פריטה מיוצגת כרשימה של מספרים. אין חשיבות לסדר בו מיוצרות הפריטות, או לסדר המטבעות בתוך פריטה. קריאה ל next כאשר תמו הפריטות תגרום לזריקת exception, ואין צורך לטפל בזה.
לדוגמה:

```
>>> for e in change_gen(5,[1,2,3]):
    print(e)
[1, 1, 1, 1, 1]
[1, 1, 1, 2]
[1, 2, 2]
[1, 1, 3]
[2, 3]
```

דרישות:

1. הפונקציה שתשלימו היא רקורסיבית.
2. הפונקציה קוראת רק לעצמה, ולא לפונקציות אחרות.

```
def change_gen(amount, coins):
   if (amount == 0):
       _____________________________________
   elif not (amount < 0 or coins == []):
       g = change_gen(amount, coins[:-1])
```

ב. השלימו את הפונקציה הבאה, אשר מחזירה פריטה **יחידה כלשהי** של amount באמצעות המטבעות שב coins. על הפריטה לכלול לכל היותר max_coins מטבעות, כאשר max_coins הוא פרמטר נוסף של הפונקציה. אם לא קיימת פריטה כזאת, הפונקציה מחזירה None.

לדוגמה:

```
>>> change_gen_maxlim(5,[1,2,3],2)
[2, 3] # the function returned a list
>>> change_gen_maxlim(5,[1,2,3],1)
>>> # the function returned None
```

```
def change_gen_maxlim(amount, coins, max_coins):
   for e in change_gen(amount,coins):
```

שאלה 4 (20 נק')

להלן תזכורת למחלקה "עץ בינארי" שראינו בכיתה, וכן לפונקציה lookup (זו לא מתודה של המחלקה), שמחזירה את הערך (val) של מפתח (key) נתון, או None אם מפתח זה לא קיים בעץ:

```
class Tree_node():
     def __init__(self, key, val):
         self.key = key
         self.val = val
         self.left = None
         self.right = None

     def __repr__(self):
         return "[" + str(self.left) + \
              " (" + str(self.key) + "," + str(self.val) + ") " \
              + str(self.right) + "]"

def lookup(root, key):
     if root == None:
        return None
     elif key == root.key:
        return root.val
     elif key < root.key:
        return lookup(root.left, key)
     else:
        return lookup(root.right, key)
```

א. השלימו את המימוש האלטרנטיבי הבא ל- lookup, שלא עושה שימוש ברקורסיה (מימוש איטרטיבי):

```
def lookup_iterative(root, key):
```

ב. התבוננו כעת בקוד הבא, שמשתמש במחלקה באופן שחורג (מאוד!) מהשימוש הסטנדרטי שלה:

```
Node1 = Tree_node(5,"a")
Node2 = Tree_node(4,"b")
Node3 = Tree_node(9,"c")
Node1.left = Node2
Node1.right = Node3
Node3.left = Node1
Node3.right = Tree_node(11,"d")
```

(המשך סעיף ב)

(i) מה יקרה אם נבצע את הקריאה lookup_iterative(Node1, 4)?

(ii) מה יקרה אם נבצע את הקריאה lookup_iterative (Node1, 7)?

(iii) מה יקרה אם נבצע את הקריאה lookup_iterative (Node1, 12)?

ג. כתבו מחדש את הקוד של lookup_iterative מסעיף א, כך שיחזיר תשובה נכונה גם במקרים הבעייתיים מבין אלו שהודגמו לעיל. כלומר יש לקיים את התכונה הבאה:

אם קיים רצף של Tree_nodes , נסמנו $N_0,\ldots N_m$, כך ש- $N_m.key=key$, ולכל $i = 1\ldots m$ מתקיים $N_i = N_{i-1}.left$ או $N_i = N_{i-1}.right$, אז על הפונקציה להחזיר את $N_m.val$. אחרת על הפונקציה להחזיר None.

<u>דרישה</u>: על הקוד להיות ללא רקורסיה (איטרטיבי).

ניתן להניח שכל מפתח מופיע לכל היותר פעם אחת, וכן שלכל צומת, המפתח של בנו השמאלי <u>המיידי</u> קטן יותר, והמפתח של בנו הימני <u>המיידי</u> גדול יותר. אין להניח הנחות נוספות לגבי המבנה או סדר הופעת המפתחות (זאת בניגוד למצב לגבי עצי חיפוש בינאריים כפי שראינו בכיתה).

```
def lookup_iterative_corrected(root, key):
```

שאלה 5 (20 נק')

השאלה עוסקת בשינוי באלגוריתם למפל-זיו לדחיסת טקסט.

<u>תזכורת</u>:

הפונקציה lz77_compress2 (שלמדנו בכיתה) מחזירה את ייצוג הביניים של דחיסת למפל-זיו של המחרוזת text.

למשל:

```
>>> lz77_compress2("abcdabc")
['a', 'b', 'c', 'd', [4, 3]]
>>> lz77_compress2("abab")
['a', 'b', 'a', 'b']
>>> lz77_compress2("abababab")
['a', 'b', [2, 4]]
```

בנוסף, ראינו בכיתה את הפונקציה:

```
def inter_to_bin(lst, w=2**12-1, max_length=2**5-1)
```

שבהינתן רשימה lst שמייצגת ייצוג ביניים של מחרוזת דחוסה, מחזירה מחרוזת של ביטים, המייצגת את המחרוזת הבינארית הדחוסה. נזכיר, שתו שלא נדחס ייוצג ע"י הביט 0 ואחריו 7 ביטים עבור התו עצמו (סה"כ 8 ביטים), ואילו מקטע שנדחס ייוצג ע"י הביט 1 ואחריו 12 ביטים עבור ההיסט אחורה, ו- 5 ביטים עבור אורך המקטע שנדחס (סה"כ 18 ביטים). שימו לב שבחישוב זה לקחנו בחשבון את ערכי ברירת המחדל של הפרמטרים w, max_length של שתי הפונקציות.

דוגמאות הרצה:

```
>>> inter_to_bin(lz77_compress2("abcdabc"))
'01100001011000100110001101100100100000000010000011'
>>> len(inter_to_bin(lz77_compress2("abcdabc")))
50      # 4*8 + 18
>>> inter_to_bin(lz77_compress2("abab"))
'01100001011000100110000101100010'
>>> len(inter_to_bin(lz77_compress2("abab")))
32      # 4*8
```

ליסה החליטה לשנות מעט את הקוד של אלגוריתם למפל-זיו לדחיסת מחרוזות.

ליסה כתבה פונקציה חדשה בשם lz77_compress2_modified שמופיעה מטה. הפונקציה שלה משתמשת בפונקציה maxmatch_new שמופיעה גם היא. maxmatch_new שונה מ maxmatch המקורית (מצורפת לסוף השאלה) בשתי השורות המודגשות בקו. בנוסף היא מצפה לפרמטר אחד נוסף (שמודגש גם הוא בקו).

```
def lz77_compress2_modified(text,w=2**12-1,max_length=2**5-1, what = False):
  result = []
  out_string = ""
  n = len(text)
  p = 0
  while p<n:
     if ord(text[p])>=128: continue
     m,k = maxmatch_new(text,p,w,max_length, what)
     if k<3:   # modified from k<2
        result.append(text[p]) #  a single char
        p+=1
     else:
        result.append([m,k])   # three or more chars in match
        p+=k
  return(result)  # produces a list composed of chars and pairs
```

```
def maxmatch_new (T,p,w=2**12-1,max_length=2**5-1, what = False):
  """ finds a maximum match of length k<=2**5-1 in a w long window, T[p:p+k]
      with T[p-m:p-m+k]. Returns m (offset) and k (match length) """

  assert isinstance(T,str)
  n = len(T)
  maxmatch=0
  offset=0
  for m in range(1,min(p+1,w)):
     k = 0
     res = min(m,max_length,n-p) if what else min(max_length,n-p)
     while k<res and T[p-m+k] == T[p+k]:
       # at this point, T[p-m:p-m+k]==T[p:p+k]
       k = k+1
     if maxmatch<k:
        maxmatch=k
        offset=m
  return offset,maxmatch
```

שימו לב שכאשר מפעילים את lz77_compress2_modified ומעבירים ערך False לפרמטר what מקבלים תוצאות זהות לאלו של lz77_compress2 המקורית.

א. טענה: קיימת מחרוזת s לא ריקה שמקיימת

```
len(inter_to_bin(lz77_compress2_modified(s, what=True ))) ==
len(inter_to_bin(lz77_compress2_modified(s, what=False)))
```

תנו דוגמא למחרוזת s כזו בצירוף ייצוג הביניים המתקבל עבור שתי ההרצות השונות, או הסבירו מדוע אין מחרוזת s כזו.

ב. טענה: קיימת מחרוזת s לא ריקה שמקיימת

```
len(inter_to_bin(lz77_compress2_modified(s, what=True ))) >
len(inter_to_bin(lz77_compress2_modified(s, what=False)))
```

תנו דוגמא למחרוזת s כזו בצירוף שני ייצוגי הביניים המתקבלים ע"י שתי ההרצות הנ"ל, או הסבירו מדוע אין מחרוזת s כזו.

ג. טענה: קיימת מחרוזת s לא ריקה שמקיימת

```
len(inter_to_bin(lz77_compress2_modified(s, what=True ))) <
len(inter_to_bin(lz77_compress2_modified(s, what=False)))
```

תנו דוגמא למחרוזת s כזו בצירוף שני ייצוגי הביניים המתקבלים ע"י שתי ההרצות הנ"ל, או הסבירו מדוע אין מחרוזת s כזו.

_____

_____

_____

תזכורת ל- maxmatch המקורית שראינו בכיתה:

```
def maxmatch(T,p,w=2**12-1,max_length=2**5-1):
  """ finds a maximum match of length k<=2**5-1 in a w long window, T[p:p+k]
      with T[p-m:p-m+k]. Returns m (offset) and k (match length) """

  assert isinstance(T,str)
  n =len(T)
  maxmatch = 0
  offset = 0
  for m in range(1,min(p+1,w)):
     k = 0
     while k<min(max_length,n-p)and T[p-m+k] == T[p+k]:
       # at this point, T[p-m:p-m+k]==T[p:p+k]
       k = k+1
     if maxmatch<k:
        maxmatch=k
        offset=m
  return offset,maxmatch
```

דף נוסף למקרה הצורך