מס' ת"ז_____ מס' מחברת_____

מבחן בקורס מבוא מורחב למדעי המחשב, CS1001.py
עם הבהרות שניתנו במהלך הבחינה / לאחריה

ביה"ס למדעי המחשב, אונ' תל אביב

סמסטר ב' 2014-15, מועד א, 16/7/2015

מרצים: אמיר רובינשטיין, פרופ' בני שור

מתרגלות: יעל ברן, מיכל קליינבורט

משך הבחינה: 3 שעות.

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

- במבחן 11 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 חמש 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.
- 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	30	
2	20	
3	15	
4	20	
5	15	
סה"כ	100	

שאלה 1 (30 נק')

בשאלה זו נממש מחלקה Binary המייצגת מספר טבעי (גדול או שווה 0) בכתיב בינארי. המספר ייוצג כמחרוזת.
להלן מימוש של כמה מתודות:

```
class Binary():
    def __init__(self, s):
        """ represent a binary number as a string """
        assert type(s)==str
        assert s.count("0") + s.count("1") == len(s)
        self.s = s

    def __repr__(self):
        return "0b" + self.s

    def length(self):
        return len(self.s)

    def __eq__(self, other):
        return self.s == other.s
```

לשם פשטות, לאורך כל השאלה נניח כי אין אפסים מובילים במספרים הבינאריים, למעט המספר "0". אין צורך לבדוק זאת במתודות שתכתבו.

<u>שימו לב</u>: בכל סעיפי שאלה זו אין להשתמש בהמרה לעשרוני בכל דרך שהיא (כולל int של פייתון).

<u>דוגמאות הרצה</u>:

```
>>> Binary("0")
0b0
>>> Binary("1101") #decimal 13, in binary
0b1101
>>> Binary("1101").length()
4
>>> Binary("1101") == Binary("1110")
False
```

א. מיכל העלתה בפגישת צוות הקורס בעיה: "אין זה תקין כי למחלקה כפי שהוגדרה לא יהיה מימוש למתודה __lt__ (less than)". צוות הקורס הסכים פה אחד לשנות את המצב.
השלימו את המתודה __lt__, שמופעלת ע"י self ומקבלת אובייקט נוסף other מהמחלקה Binary, ומחזירה True אם self < other (self קטן ממש מ- other), אחרת מחזירה False. יש להשלים קוד במלבן הריק בלבד.

```
class Binary():
    ...
   def __lt__(self, other):
       n1 = self.length()
       n2 = other.length()
       if n1<n2:
           return True
       elif n1>n2:
           return False
       else:
```

ב. אמיר מימש מתודה של המחלקה בשם inc (קיצור של increment) שמוסיפה 1 למספר self. למשל:

```
>>> b = Binary("1101")
>>> b.inc()
>>> b
0b1110
>>> b.inc()
>>> b
0b1111
>>> b.inc()
>>> b
0b10000
```

כעת הוא מעוניין לממש את המתודה __add__ (מימוש לאופרטור +). המתודה מופעלת ע"י self ומקבלת אובייקט נוסף other מהמחלקה Binary. היא מחזירה אובייקט חדש מהמחלקה Binary, שמייצג את תוצאת החיבור בין שני המספרים שבקלט. self ו- other לא ישתנו. יש לממש את __add__ תוך שימוש במתודה inc(). בקוד שתכתבו, אין לגשת ישירות לשדות s של self או של other. יש להשלים קוד במלבן הריק בלבד. הערה: ניתן (וכדאי) לפתור גם ללא גישה ל- __repr__.

```
class Binary():
    ...
    def __add__(self, other):
        """ operator +, using inc() """
        res = Binary(self.s) #a new copy of self
        
        return res
```

ג. אם self ו- other הם מספרים בינאריים בעלי n ביטים כל אחד, ובהנחה שסיבוכיות הזמן של inc עבור מספר בעל n ביטים היא $O(n)$ במקרה הגרוע, מהי סיבוכיות הזמן של __add__ הנ"ל במקרה הגרוע? אין צורך להסביר.

O(_____)

ד. בני נוזף באמיר וטוען כי השימוש ב- inc() כדי לממש חיבור אינו יעיל, ושם את צוות הקורס ללעג ולקלס. הוא מציע את המימוש הבא, המאפשר גישה לשדות s של self ושל other:

```
class Binary():
    ...
    def __add__(self, other):
        """ operator +, without inc() """
        maxlen = max(self.length(), other.length())
        s1 = self.s.zfill(maxlen) #see explanation later
        s2 = other.s.zfill(maxlen)
        bits = [0]*maxlen
        carry = 0
        for i in range(1,maxlen+1):
            summ = 0
            if s1[-i] == "1":
                summ +=1
            if s2[-i] == "1":
                summ +=1
            summ += carry
            print(summ)
            bits[-i] = "1" if summ%2==1 else "0"
            carry = 0 if summ<2 else 1
        if carry == 1:
            bits = ["1"] + bits
        return Binary("".join(bits))
```

<u>הערה</u>: הפונקציה zfill מוסיפה למחרוזת שמפעילה אותה אפסים משמאל, עד שאורכה הכולל של המחרוזת מגיע ל- maxlen. לדוגמה:

```
>>> "11".zfill(5)
'00011'
```

שימו לב כי אחת השורות במימוש זה היא שורת ההדפסה המודגשת `print(summ)`.
עבור הפקודה הבאה בפייתון, ציינו מה יודפס למסך:

```
>>> b = Binary("100011") + Binary("1011")
```

ה. אם self ו- other הם מספרים בינאריים בעלי *n* ביטים כל אחד, מהי סיבוכיות הזמן של __add__ מסעיף ד' במקרה הגרוע? אין צורך להסביר.

O(_____)

מס' ת"ז_____ מס' מחברת____

שאלה 2 (20 נק')

נגדיר קוד זוגיות דו-מימדי "חסר פינה", הדומה לזה שראינו בכיתה: עבור 9 ביטים של מידע D1, …, D9, נוסיף לכל שורה ביט זוגיות (R1, R2, R3) ולכל עמודה ביט זוגיות (C1, C2, C3).

D1	D2	D3	R1
D4	D5	D6	R2
D7	D8	D9	R3
C1	C2	C3	--

שימו לב כי בניגוד לקוד שראינו בכיתה, כאן אין ביט זוגיות בפינה הימנית התחתונה (מכאן השם "חסר פינה").

ניתן לתאר מילת קוד ע"י D1 D2 D3…D9 R1 R2 R3 C1 C2 C3, כלומר 15 ביטים.

עוד נזכיר כי קוד נקרא קוד (n,k,d) אם n הוא מספר הביטים במילת קוד (כלומר מילה חוקית), k הוא מספר ביטי המידע, ו- d הוא המרחק המינימלי בין שתי מילות קוד.

א. עבור הקוד הנ"ל, קיבעו מהם n, k, d. לקביעת המרחק המינימלי, ניתן להסתמך על הטענה הבאה: בקוד זה יש מילת קוד (כלומר מילה חוקית) שמרחקה ממילת האפס (מילת הקוד שבה כל הביטים הם 0) הוא d. מכאן, שלמציאת המרחק המינימלי של הקוד, d, מספיק למצוא מילת קוד במרחק מינימלי ממילת האפס.

n = ____ k = ____ d = ____

הראו מילת קוד במרחק מינימלי ממילת האפס (מספיק למלא את המקומות בהם יש 1, מקומות ריקים ייחשבו כ- 0):

			--

ב. הראו מילת קוד במרחק **מקסימלי** ממילת האפס (מספיק למלא את המקומות בהם יש 1, מקומות ריקים ייחשבו כ- 0):

			--

(המשך השאלה בעמוד הבא)

ג. התשדורת הבאה התקבלה לאחר מעבר בערוץ תקשורת רועש. האם היא נמצאת במרחק 1 ממילת קוד כלשהי?

0	0	0	0
0	0	0	1
0	0	0	1
0	0	0	--

הוכיחו בקצרה, **או** ע"י הצגת מילת קוד במרחק 1 מהתשדורת הנ"ל:

			--

או ע"י הסבר מדוע אין כזו: _____

_____

_____

_____

_____

ד. תהי $A \subseteq \{0,1\}^{15}$ קבוצת כל המילים במרחב $\{0,1\}^{15}$ שנמצאות במרחק הקטן או שווה ל- 2 ממילת קוד כלשהי. האם הקבוצה A כוללת את כל המילים ב- $\{0,1\}^{15}$? הסבירו תשובתכם.

<u>תשובה (הקיפו בעיגול)</u>: כן / לא

<u>הסבר</u>:

שאלה 3 (15 נק')

נתונה רשימה של n מחרוזות $\{s_1, s_2, \ldots, s_n\}$ מעל האי"ב הלטיני (אותיות קטנות, סה"כ 26 אותיות באי"ב). כל אחת מן המחרוזות היא באורך 100 בדיוק, וכל המחרוזות שונות זו מזו. בהינתן k חיובי קטן שווה מ-100, אנו מעוניינים למצוא את כל הזוגות הסדורים של מחרוזות שונות (s, t), כך שקיימת חפיפה באורך k בדיוק בין סיפא (סיומת) של s לרישא (התחלה) של t.

לדוגמה, אם האוסף מכיל את המחרוזות

s_1 = "a"*100

s_2 = "a"*60+"b"*40

s_3 = "a"*10+"b"*40+"c"*50

אז עבור k=50 יש חפיפה באורך k בין הסיפא של s_1 לרישא של s_2, ויש חפיפה באורך k בין הסיפא של s_2 לרישא של s_3. שימו לב שאנו לא מתעניינים בחפיפות אפשריות של מחרוזות עם עצמן, כמו למשל החפיפה באורך 50 בין סיפא של s_1 לרישא שלה עצמה. לכן הפלט במקרה זה יהיה שני הזוגות (s_1, s_2) ו- (s_2, s_3).

א. יעל מציעה את השיטה הבאה למציאת כל החפיפות: לכל מחרוזת נבדוק את הסיפא באורך k שלה אל מול כל הרישות באורך k של כל המחרוזות האחרות. מיכל טוענת כי השיטה הזו לא יעילה, וכי אם n הוא בסדר גודל של מליונים, הבדיקה לא תסתיים בזמן סביר.
ציינו מהי סיבוכיות הזמן של השיטה שיעל מציעה, כתלות ב- n וב- k במונחים של $O(\ldots)$. הניחו כי השוואה בין שתי תת מחרוזות באורך k דורשת $O(k)$ פעולות.

O(_____)

ב. בני טוען כי ניתן לבצע את המשימה בצורה יעילה יותר, תוך שימוש בפונקציות hash. תארו במילים פתרון אפשרי כזה. בתשובתכם: ציינו באיזה מבנה נתונים built-in של פייתון תשתמשו (למשל list, dict וכד'), מה בדיוק תכניסו אליו, כמה איברים יוכנסו אליו, ומה תהיה סיבוכיות הזמן של הפתרון שלכם. ניתן להניח כי חישוב hash על מחרוזת באורך k לוקח $O(k)$ פעולות, וכי פונקציית ה- hash היא חד-חד-ערכית על המחרוזות ב- $\{s_1, s_2, \ldots, s_n\}$ ועל הרישות והסיפות הרלוונטיות.

שאלה 4 (20 נק')

יעל מציעה גרסה חדשה של האלגוריתם quicksort (מיון מהיר). בגרסה זו, חלוקת הרשימה נעשית באמצעות פונקציה partition שנתונה להלן. הפונקציה מחלקת את הרשימה L על פי איבר הציר L[0]. בסיום, הרשימה L עצמה מסודרת מחדש, כך שהיא מחולקת לשני איזורים: האיזור השמאלי ובו איברים שקטנים או שווים לציר, והאיזור הימני, ובו איברים שגדולים ממש מהציר. השורה הלפני אחרונה של הפונקציה דואגת לכך שהציר ימוקם בדיוק בין שני החלקים הללו. כפי שניתן לראות, בסיום, הפונקציה מחזירה את מיקום הציר.

```
def partition(L):
    pivot = L[0]
    left = right = -1
    while right < len(L)-1:
        right += 1
        if L[right] <= pivot:
            left += 1
            L[left], L[right] = L[right], L[left]
    L[0], L[left] = L[left], L[0] #now pivot is between the two parts
    return left #pivot's position
```

א. מה יהיה הפלט של הפקודות הבאות? השלימו בשני המקומות הדרושים.

```
>>> L = [4,5,6,1,3,7,2]
>>> partition(L)
____
>>> L
[_____________________________________]
```

ב. עבור רשימה באורך n, מהו הערך המינימלי ומהו הערך המקסימלי ש- partition יכולה להחזיר כתלות ב- n?
תנו דוגמה לרשימה מתאימה באורך $n=5$ עבור כל אחד מהמקרים שציינתם.

ערך מינימלי: _____ דוגמה לרשימה עבורה יתקבל ערך זה: [_____]

ערך מקסימלי: _____ דוגמה לרשימה עבורה יתקבל ערך זה: [_____]

ג. השלימו את המימוש לפונקציה quicksort, העושה שימוש ב- partition הנ"ל. quicksort תחזיר רשימה ממוינת חדשה ולא תשנה את הקלט L, ולשם כך היא מייצרת העתק של L ונדרשת לעבוד עליו.

```
def quicksort(L):
    if len(L) <= 1:
        return L
    copy = L[:] #work with a new copy of L
```

שאלה 5 (15 נק')

בשאלה זו עליכם לכתוב פונקציה רקורסיבית המייצרת את כל הבחירות עם חזרות של k איברים מתוך רשימת איברים elements. הפונקציה תחזיר רשימה, שאיבריה הם רשימות. מותר להניח כי ב- elements אין חזרות.

להלן שתי דוגמאות הרצה של הקוד הנדרש:

```
>>> a = repetitions([1,2],3)
>>> for r in a:
      print(r)

[1, 1, 1]
[1, 1, 2]
[1, 2, 2]
[2, 2, 2]
>>> a = repetitions([1,2,3,4],3)
>>> for r in a:
      print(r)

[1, 1, 1]
[1, 1, 2]
[1, 1, 3]
[1, 1, 4]
[1, 2, 2]
[1, 2, 3]
[1, 2, 4]
[1, 3, 3]
[1, 3, 4]
[1, 4, 4]
[2, 2, 2]
[2, 2, 3]
[2, 2, 4]
[2, 3, 3]
[2, 3, 4]
[2, 4, 4]
[3, 3, 3]
[3, 3, 4]
[3, 4, 4]
[4, 4, 4]
```

א. השלימו את הקוד הנתון במלבן בלבד. שימו לב כי סדר יצירת הרשימות אינו חייב להיות זהה לסדר שלמעלה, ובלבד שכל בחירה תיוצר פעם אחת בדיוק. לעומת זאת, סדר האיברים הפנימי בתוך כל בחירה זהה לסדרם ברשימה המקורית. לדוגמה, בדוגמה האחרונה אחת הבחירות היתה [1,2,3], ולא למשל [1,3,2] או [3,2,1].

הערה: אין להשתמש במודולים של פייתון, כמו itertools.

```
def repetitions(elements,k):
    """ Produces subsets containing exactly k elements, where
    repetitions are allowed. Retains order of elements """
    if elements == []:
        return []
    if k == 0:
        return [[]]
    else:
        result = []
        
        return result
```

ב. הסבירו מילולית את אופן הפעולה של הפתרון שלכם.

דף נוסף למקרה הצורך