מס' ת"ז_____ מס' מחברת_____

מבחן בקורס מבוא מורחב למדעי המחשב, CS1001.py
עם הבהרות / תיקונים שניתנו בעת הבחינה

ביה"ס למדעי המחשב, אונ' תל אביב

סמסטר א' 2015-16, מועד א, 31/1/2016

מרצים: פרופ' עמירם יהודאי, אמיר רובינשטיין

מתרגלות: יעל ברן, מיכל קליינבורט

משך הבחינה: 3 שעות.

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

- 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 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.

טבלת ציונים: (לשימוש הבודקים)

שאלה	ערך	ניקוד
1	15	
2	20	
3	15	
4	20	
5	30	
סה"כ	100	

בהצלחה !

שאלה 1 (15 נק')

השאלה עוסקת בשיטת הצפנה הידועה בשם "צופן החלפה". בשיטה זו, כל אות מהא"ב מוחלפת באות כלשהי מאותו א"ב. את המיפוי בין האותיות לאותיות שמחליפות אותן ניתן לייצג בפייתון למשל ע"י מילון (dictionary). לדוגמה, עבור המילון הבא:

```
d = {'a': 'a', 'c': 'd', 'b': 'c', 'd': 'b'}
```

האות a תישאר a, האות c תוחלף ב- d וכן הלאה. אם כן ההודעה "abcd" תוצפן ל- "acdb".

המילון צריך כמובן לייצג מיפוי חד חד ערכי ועל מהא"ב לעצמו: כל תו ממופה לתו אחד, ואין תו שממופים אליו כמה תווים.

א. השלימו את הפונקציה הבאה encrypt, שמקבלת מחרוזת text ומילון d המייצג מיפוי כנ"ל מא"ב לעצמו (ייתכן שיש ב- text תווים שלא מופיעים במיפוי הזה). הפונקציה תחזיר את המחרוזת המוצפנת המתקבלת מ- text לאחר הפעלת המיפוי שמיוצג ב- d. תווים ב- text שלא מופיעים במילון יישארו ללא שינוי.
דוגמת הרצה:

```
>>> d = {'c': 'a', 'd': 'c', 'a': 'd', 'b': 'b'}
>>> encrypt("abcd123", d)
'dbac123'
```

```
def encrypt(text, d):
    cipher = ""
    for
    return cipher
```

ב. נתונה לכם פונקצית גנרטור בשם permutations, המקבלת רשימה ומחזירה את כל התמורות (פרמוטציות) שלה. למשל:

```
>>> for perm in permutations(["a", "b", "c"]):
    print(perm)

['a', 'b', 'c']
['b', 'a', 'c']
['b', 'c', 'a']
['a', 'c', 'b']
['c', 'a', 'b']
['c', 'b', 'a']
```

השלימו את הפונקציה `all_mappings`, המקבלת רשימה של תווי א״ב כלשהו, ומחזירה את כל המיפויים (מילונים) האפשריים בין תווי האא״ב לעצמו. `all_mapping` צריכה להיות בעצמה פונקצית גנרטור (כלומר לעשות שימוש ב- yield). למשל:

```
>>> for map in all_mappings(["a", "b", "c"]):
     print(map)

{'c': 'c', 'b': 'b', 'a': 'a'}
{'c': 'c', 'b': 'a', 'a': 'b'}
{'c': 'a', 'b': 'c', 'a': 'b'}
{'c': 'b', 'b': 'c', 'a': 'a'}
{'c': 'b', 'b': 'a', 'a': 'c'}
{'c': 'a', 'b': 'b', 'a': 'c'}
```

```
def all_mappings(alphabet):
    for perm in permutations(alphabet):
```

שאלה 2 (20 נק')

שאלה זאת עוסקת בקוד לזיהוי/תיקון שגיאות, לפי מודל ערוץ התקשורת שהוצג בהרצאות, שמניח שביט מוחלף (מ 0 ל 1 או מ 1 ל 0) במהלך השידור באופן אקראי ובלתי תלוי בביטים האחרים, ושההסתברות להחלפה קטנה, כך שהסיכוי ל m שגיאות גדול מהסיכוי ל $m+1$ שגיאות.

נתון קוד עבור **הודעות** באורך 2 ביטים. **התשדורות** באורך 4 נוצרות לפי ההגדרה הבאה:

$00 \rightarrow 0000,\ 01 \rightarrow 0111,\ 10 \rightarrow 1011,\ 11 \rightarrow 1110$

(זאת לא טעות, הביטים שנוספים בשניים מהמקרים זהים).

א. בכל אחד מהמקרים הבאים עליכם לציין מהו התרחיש הסביר ביותר מבין שלושת הבאים:

1. נפלה שגיאה אחת, ואפשר לתקן אותה. במקרה זה ציינו גם מהי ההודעה המקורית.
2. נפלה שגיאה אחת, אך לא ניתן לתקן אותה.
3. נפלו שתיים או יותר שגיאות, ולא ניתן לתקן אותן.

a) 1010 תשובה: _____

b) 1001 תשובה: _____

c) 1101 תשובה: _____

d) 0101 תשובה: _____

ב. מיטל טוענת שאפשר להגדיר קוד אחר עבור הודעות באורך 2 ותשדורות באורך 4, שיאפשר תמיד לתקן שגיאה בודדת. רויטל טוענת שזה בלתי אפשרי. מי צודקת? נמקו את תשובתכם באופן תמציתי וקולע על מנת לא לאבד נקודות.

_____

_____

_____

_____

ג. הקוד שהוגדר בתחילת השאלה לא מסוגל לתקן שתי שגיאות. אילו רצינו לבנות קוד שכן יוכל לתקן שתי שגיאות, מה הערך הקטן ביותר האפשרי של d (המרחק המינימלי) של הקוד? אין צורך להסביר.

$d =$ _____

שאלה 3 (15 נק')

האלגוריתמים לצמצום רעש בתמונות על פי ממוצע מקומי או חציון מקומי השתמשו בפונקציה `local_operator` שחישבה ערך חדש לכל פיקסל שאינו קרוב מדי לשולי התמונה על ידי הפעלת אופרטור op על הפיקסלים השכנים. כמו כן נעשה שימוש בפונקציה `items`:

```
def local_operator(A, op, 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] = op(items(A[i-k:i+k+1,j-k:j+k+1]))
  return res

def items(mat):
    '''flatten mat elements into a list'''
    n,m = mat.dim()
    lst = [mat[i,j] for i in range(n) for j in range(m)]
    return lst
```

כדי לאפשר הגדרות חלופיות לסביבה של פיקסל נתון, ולאפשר תיקון של כל הפיקסלים, כולל אלו שבשולי התמונה, נכתבה הפונקציה `local_operator` מחדש, כך שחישוב רשימת הפיקסלים בסביבה ייעשה בפונקציה נפרדת `neighbors`.

`neighbors` מקבלת כפרמטרים את המטריצה A, את ממדיה n ו m, את האינדקסים i ו j של הפיקסל הנוכחי ופרמטר אופציונלי k, גודל הסביבה. `neighbors` מחזירה את רשימת ערכי הפיקסלים שעליהם יש להפעיל את האופרטור. נתונה הגירסה החדשה של `local_operator`:

```
def local_operator(A, op, k=1) :
    n,m = A.dim()
    res = copy(A)  # brand new copy of A
    for i in range(0,n):          # מעבר על כל השורות
        for j in range(0,m):      # והעמודות
            # using the function neighbors
            res[i,j] = op(neighbors(A,n,m,i,j,k))
    return res
```

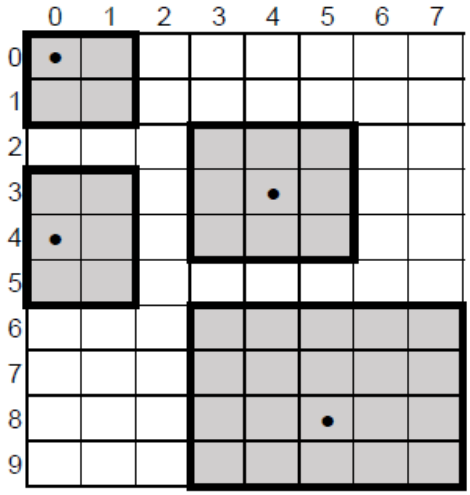
א. השלימו את הפונקציה `neighbors` באופן שהערך שתחזיר `local_operator` יהיה זהה לזה שהוחזר בגירסה המקורית (כלומר הפיקסלים הקרובים לשולי התמונה יישארו ללא שינוי).

הדרכה: ניתן להניח שכאשר op (לדוגמה ממוצע או חציון) פועלת על אבר בודד, מוחזר ערכו של האבר עצמו.

```
def neighbors(A,n,m,i,j,k=1):

    if _____ :

        return _____

    else:

        return items(A[i-k:i+k+1,j-k:j+k+1])
```

ב. השלימו גירסה חדשה של הפונקציה `neighbors` כך שכל הפיקסלים יוחלפו על ידי הפעלת האופרטור על הסביבה, אשר תכלול את כל השכנים במרחק קטן או שווה k שקיימים.

לדוגמא, במטריצה A, עבור $k=1$, ראינו בהרצאות שהסביבה של האלמנטים שאינם קרובים לשוליים (כדוגמת A[3,4]) כוללת 9 איברים. לעומת זאת, הסביבה של A[0,0] כוללת 4 איברים: A[0,0], A[0,1], A[1,0], A[1,1], והסביבה של A[4,0] כוללת את האלמנטים A[3,0], A[3,1], A[4,0], A[4,1], A[5,0],A[5,1] . הציור מראה גם את הסביבה של A[8,5] עבור $k=2$.




```
def neighbors(A,n,m,i,j,k=1):

    return items(A[_____ : _____ ~~+~~,

                 ~~A[~~_____ : _____ ] )
```

(המחיקות הן תיקון בעת הבחינה)

מס' ת"ז_____ מס' מחברת____

שאלה 4 (20 נק')

השאלה עוסקת בקודים לדחיסה, כדוגמת קוד האפמן.

תזכורת: נתון אלפבית $\Sigma = \{a_1, a_2, \dots, a_n\}$ וקורפוס שבו a_i מופיעה w_i פעמים. קידוד $D = \{d_1, d_2, \dots, d_n\}$ מגדיר את המחרוזת הבינארית שמתאימה לכל תו מהאלפבית. כלומר $a_1 \to d_1$, $a_2 \to d_2$ וכו'. בכיתה הגדרנו גם את האורך הממושקל (weighted length) של קידוד ביחס לקורפוס נתון:

$$L_W(D) = \sum_{i=1}^{n} w_i \cdot len(d_i)$$

א. נתון הקורפוס הבא: "a"*5 + "b"*2 + "c"*2 + "d"*1 (כלומר "aaaaabbccd") מעל האי"ב {"a", "b", "c", "d"}. עבור כל אחד מהקידודים הבאים (שאינם בהכרח קידודי האפמן) ציינו מהו האורך הממושקל שלו ביחס לקורפוס.

(i) $D_1 = \{0, 100, 101, 11\}$ אורך ממושקל: _____

(ii) $D_2 = \{00, 01, 10, 11\}$ אורך ממושקל: _____

ב. עמירם טוען שמצא קוד עם אורך ממושקל קטן יותר מכל אחד מאלו של סעיף א': $D_3 = \{0, 1, 01, 11\}$

הסבירו לעמירם בקצרה מדוע הקוד הזה בעייתי ולא ניתן להשתמש בו.

ג. האם ייתכן קידוד שלא סובל מהבעיה של סעיף ב', בעל אורך ממושקל קטן יותר (ביחס לקורפוס הנתון) מכל אחד מאלו של סעיף א'? אם לדעתכם לא, נמקו, אחרת הראו קידוד כזה.

ד. תנו דוגמה לקורפוס אחר מעל האי"ב {"a", "b", "c", "d"}, שביחס אליו לשני הקידודים מסעיף א' אותו אורך ממושקל. ציינו מהו האורך הממושקל עבור הדוגמה שבחרתם. הבהרה בעת הבחינה: על הקורפוס להכיל כל אחת מאותיות האי"ב הנתון לפחות פעם אחת.

"a"*____ + "b"*____ + "c"*____ + "d"*____

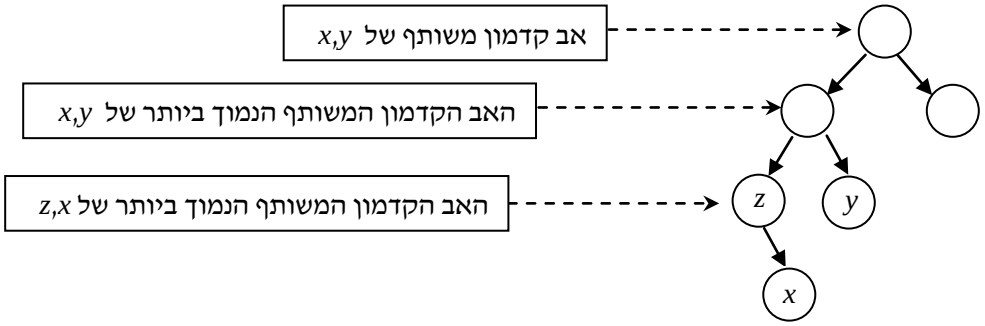
אורך ממושקל: _____

שאלה 5 (30 נק')

שאלה זו עוסקת בעצי חיפוש בינריים. תזכורת: המפתחות בעץ חיפוש בינארי שונים זה מזה.

נאמר שצומת w הוא **אב קדמון משותף** (common ancestor) של זוג צמתים נתונים x, y (לא בהכרח עלים) אם הצומת w נמצא על המסלול מהשורש ל-x ועל המסלול מהשורש ל-y, כלומר x,y שניהם צאצאים של w. שימו לב כי כל צומת בעץ הוא צאצא של עצמו.

נגדיר **אב קדמון משותף נמוך ביותר** (least common ancestor) של שני צמתים נתונים x,y להיות הצומת w שהינו אב קדמון משותף של x,y שרחוק ביותר מהשורש מבין כל האבות הקדמונים המשותפים ל- x,y.



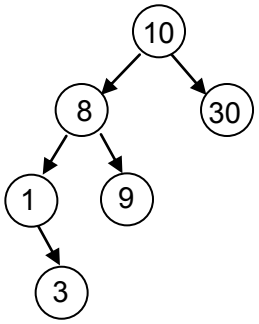
לנוחיותכם, מצורפת המתודה `__init__` של המחלקה `Tree_node` שראיתם בכיתה עבור צומת בעץ חיפוש בינרי.

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

א. השלימו בעמוד הבא את הפונקציה הרקורסיבית `lca(root, key1, key2)` שמקבלת את הצומת root (משתנה מטיפוס `Tree_node`) וכן זוג מפתחות key1, key2 של צמתים קיימים בעץ ששורשו root. הפונקציה מחזירה את הצומת בעץ שהוא האב הקדמון המשותף הנמוך ביותר של הצמתים בעץ בעלי המפתחות key1, key2. בפרט, טיפוס האובייקט המוחזר יהיה `Tree_node`. הניחו כי הקלט לפונקציה תקין.
שימו לב: על סיבוכיות הזמן של הפונקציה במקרה הגרוע להיות $O(h)$ כאשר h הינו עומק העץ ששורשו root.

דוגמת הרצה: שורות הקוד הבאות בונות את העץ שמופיע בשרטוט מימין (המפתחות מופיעים בצמתים).

```
>>> root = None
>>> root = insert(root, 10, "hello")
>>> root = insert(root, 8, "biscuit")
>>> root = insert(root, 1, "tea")
>>> root = insert(root, 3, "dear")
>>> root = insert(root, 9, "muffin")
>>> root = insert(root, 30, "sherlock")
```



עבור העץ שנבנה:

```
>>> lca(root, 1, 3).key
1
>>> lca(root, 9, 3).key
8
```

```
def lca(root, key1, key2):
    # נניח כי שני המפתחות הנתונים נמצאים בעץ
    key1, key2 = min(key1, key2), max(key1, key2)
    if key2 < root.key:
        return _____
    elif _____:
        return _____
    else:
        return _____
```

ב. השלימו את הפונקציה הרקורסיבית `all_ca(root, key1, key2)` שמקבלת את הצומת root (משתנה מטיפוס `Tree_node`) וכן זוג מפתחות key1, key2 של צמתים קיימים בעץ ששורשו root.
הפונקציה מחזירה רשימה של פייתון שמכילה את כל האבות הקדמונים המשותפים לצמתים בעלי המפתחות הנתונים, מסודרים מהשורש לאב הקדמון המשותף הנמוך ביותר. גם בסעיף זה הניחו כי הקלט לפונקציה תקין.
שימו לב: על סיבוכיות הזמן של הפונקציה במקרה הגרוע להיות $O(h)$ כאשר h הינו עומק העץ ששורשו root.

דוגמת הרצה עבור העץ הקודם:

```
>>> lst = all_ca(root, 1, 3)
>>> [item.key for item in lst]
[10, 8, 1]
```

```
def all_ca(root, key1, key2):
    key1, key2 = min(key1, key2), max(key1, key2)
```

ג. גדליהו מימש את הפונקציה `lca_many_power_of_2(root, keys_lst)` שמקבלת בנוסף לשורש העץ root גם רשימה keys_lst של k מפתחות ומחזירה את האב הקדמון המשותף הנמוך ביותר לכל k המפתחות הנתונים. לשם פשטות, הפונקציה של גדליהו מניחה שאורך רשימת המפתחות k הינו חזקה שלמה של 2. הניחו שהקלט לפונקציה תקין.
להלן הפונקציה שכתב גדליהו:

```
def lca_many_power_of_2(root, keys_lst):
    k = len(keys_lst)    # מניחים שאורך הרשימה k הוא חזקה שלמה של 2
    if k==1:
        # אם הרשימה מכילה מפתח אחד נחזיר את הצומת בעל המפתח
        print("Returning a node with key:" , keys_lst[0])
        return lca(root, keys_lst[0], keys_lst[0])
    lca_candidates = []
    for i in range(k//2):
        print("key1:", keys_lst[2*i], ", key2:" , keys_lst[2*i+1])
        lca_candidates.append(lca(root, keys_lst[2*i], \
                                        keys_lst[2*i+1]))
    lca_cands_keys = [node.key for node in lca_candidates]
    return lca_many_power_of_2(root, lca_cands_keys)
```

שימו לב לשתי שורות ההדפסה בפונקציה הנ"ל. השלימו בתיבה הריקה מה יודפס כאשר נפעיל את `lca_many_power_of_2` על העץ מסעיף א' ועל רשימת המפתחות `[8,9,3,30]` (אורך הרשימה הוא 4, כלומר חזקה שלמה של 2, ולכן קלט חוקי לפונקציה זו):

```
>>> lca_many_power_of_2(root, [8,9,3,30])
```

ד. מהי סיבוכיות הזמן במקרה הגרוע של הפונקציה `lca_many_power_of_2` מסעיף ג' כתלות בעומק העץ h ו/או אורך רשימת המפתחות k? תנו תשובה במונחים אסימפטוטיים, הדוקה ככל שתוכלו.

O(_____)

ה. השלימו את הפונקציה `lca_many(root, keys_lst)` שמקבלת את הצומת root (משתנה מטיפוס `Tree_node`) ורשימת מפתחות keys_lst של k מפתחות. הפונקציה מחזירה את האב הקדמון המשותף הנמוך ביותר לכל k המפתחות הנתונים. הפונקציה מבצעת <u>קריאה אחת בודדת</u> ל- `lca` מסעיף א'. הניחו כי הקלט לפונקציה `lca_many` תקין.

```
def lca_many(root, keys_lst):
    x = _____________________________________________________
    y = _____________________________________________________
    return lca(root, x, y)
```

ו. מהי סיבוכיות הזמן במקרה הגרוע של הפונקציה `lca_many` שמימשתם בסעיף ה', כתלות בעומק העץ h ו/או אורך רשימת המפתחות k? תנו תשובה במונחים אסימפטוטיים, הדוקה ככל שתוכלו.

O(_____)

דף נוסף למקרה הצורך