מס' ת"ז_____ מס' מחברת_____

מבחן בקורס מבוא מורחב למדעי המחשב, CS1001.py

סמסטר ב' 2013

מועד א

מרצים: פרופ' חיים וולפסון, פרופ' עמירם יהודאי

מתרגלים: יואב רם, אמיר רובינשטיין

משך הבחינה: 3 שעות.

חומר עזר מותר: 4 עמודים בגודל A4 כ"א.

- יש לכתוב את כל התשובות בטופס הבחינה. המחברת תשמש כטיוטה בלבד ולא תיבדק.
- במבחן 10 עמודים – בידקו שכולם בידיכם.
- 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.
- אנו ממליצים לא "להיתקע" על אף שאלה בודדת, 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 (מעוגל כלפי מעלה).
- בכל השאלות מותר להסתמך על סעיפים קודמים, גם אם לא עניתם עליהם.
- 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.
- ניתן לצטט טענות שנטענו בהרצאה או בשיעורי התרגול.

בהצלחה !

טבלת ציונים: (לשימוש הבודקים)

שאלה	ניקוד
1	
2	
3	
4	
5	
סה"כ	

מס' ת"ז_____ מס' מחברת_____

<u>שאלה 1 (20 נק')</u>

א. נתון קורפוס המכיל 4 תווים בתדירויות הבאות: $1, 3, 9, 27$ (התו הראשון מופיע פעם אחת, השני 3 פעמים, וכו'). ציירו מבנה אפשרי לעץ האפמן הנבנה על-פי קורפוס זה. הניחו כי בעת איחוד שני צמתים, הקטן יותר יהיה הבן השמאלי.

ב. נתון קורפוס המכיל m תווים בתדירויות הבאות: $1, 3, 9, \ldots, 3^{m-1}$ (התו הראשון מופיע פעם אחת, השני 3 פעמים, וכו').
בונים קוד האפמן מקורפוס זה, ומקודדים באמצעותו טקסט המכיל את אותם תווים, כאשר כל תו מופיע פעם אחת בדיוק. מהו מספר הביטים בטקסט הדחוס, כתלות ב-m? רישמו <u>ביטוי מדויק</u>, ונמקו.

הביטוי:

נימוק:

ג. נתון קורפוס המכיל 17 תווים: התו 'a' מופיע בדיוק k פעמים, וכל אחד מ-16 התווים האחרים 'b', 'c',…, 'q' מופיע בדיוק פעם אחת.

מהו ה-k המינימלי, שמבטיח שקידוד האפמן של 'a' על-פי קורפוס זה יהיה באורך 1? הסבירו בקצרה.

k =

נימוק:

שאלה 2 (20 נק')

להלן שיטה לאיתור שגיאות שמהווה הרחבה של קוד המינג (7,4,3) שנלמדה בשיעור:

בהינתן 4 ביטים של אינפורמציה, נוסיף להם 3 ביטים כמו בקוד המינג (7,4,3) שלמדנו, וביט נוסף, שמהווה ביט זוגיות (parity) ל-7 הביטים הקודמים. כלומר הביט x8 הוא XOR של שבעת הביטים הקודמים (סכום מודולו 2 שלהם).

הפונקציה `hamming_encode` שנלמדה בכתה היא:

```
def hamming_encode(x3,x5,x6,x7):
    """ Hamming encoding of the 4 bits input """
    x1= (x3+x5+x7) % 2
    x2= (x3+x6+x7) % 2
    x4= (x5+x6+x7) % 2
    return (x1,x2,x3,x4,x5,x6,x7)
```

דוגמאות:

- עבור ההודעה **0010** נשלח את השדר 01010101
- עבור ההודעה **0110** נשלח את השדר 11001100

(הביטים שנוספו מודגשים בקו)

א. מהו מספר השדרים חוקיים (codewords) ? _____

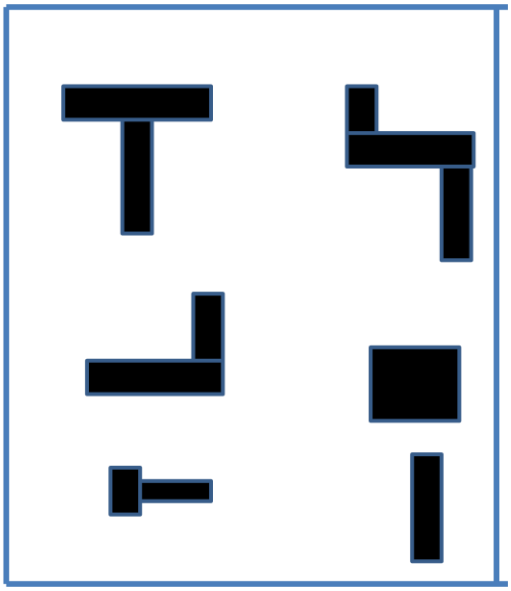
ב. מהו d, מרחק Hamming המינימלי של הקוד (המרחק המינימלי בין שני שדרים חוקיים כלשהם)? _____

ג. בהינתן שדר, ניתן לבדוק את הזוגיות שלו (האם מספר ה-1 ים בכל 8 הביטים זוגי או איזוגי), וכמו כן ניתן לבדוק האם 7 הביטים של קוד המינג מהווים שדר חוקי. עבור כל אחד מהצרופים האפשריים בטבלה, ציינו את מספר השגיאות בשדר כולו, האם ניתן לתקן אותו (כלומר לשחזר את ההודעה המקורית), ואם כן, באיזה חלק יהיה התיקון (ב-7 הביטים הראשונים המהווים קוד המינג, בביט הזוגיות השמיני, או בשני החלקים). כרגיל, אנחנו מניחים כי שגיאות בביטים שונים הן בלתי תלויות וכי ההסתברות לשגיאה בכל ביט נמוכה מאוד, ומתייחסים תמיד להודעה המקורית הסבירה ביותר.

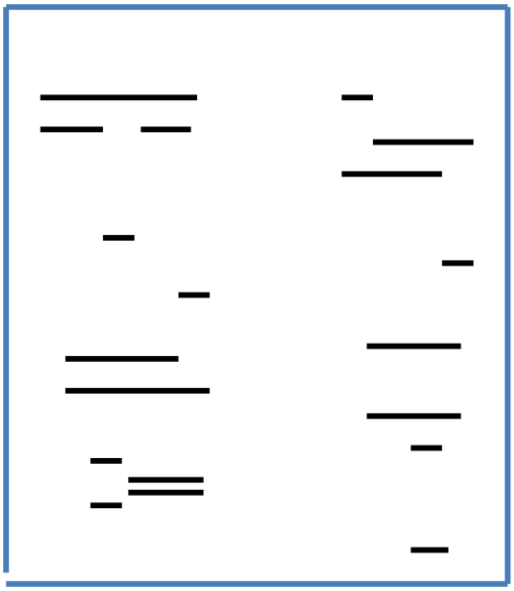
מספר 1-ים	יש שגיאה בקוד המינג (7 ביטים ראשונים)	מספר השגיאות	ניתן לתיקון? (כן/לא)	באיזה חלק יהיה התיקון?
זוגי	יש			
זוגי	אין			
איזוגי	יש			
איזוגי	אין			

<u>שאלה 3 (20 נק')</u>

תלמידי סדנא בראיה ממוחשבת קיבלו פרויקט שבמסגרתו עליהם לנתח צורות מהסוג שמוצג בתמונה א'. המאפיין של צורות אלו שהן בעלות פינות ישרות, צבועות בצבע שחור ומונחות על רקע לבן בצורה כזאת שקווי הגבול התוחמים אותן יכולים להיות רק אנכיים (vertical) או אופקיים (horizontal) בתמונת הקלט המצולמת.



<u>תמונה א'</u>: דוגמא לצורות מצולמות.



<u>תמונה ב'</u>: פעולת מגלה שפות אופקיות על החלקים של תמונה א'.

א. בחלק הראשון של הפרויקט יש לכתוב קוד למגלה שפות (edge detector) בתמונות כמו תמונה א'. בפרויקט הנ"ל ניתן להסתפק בגילוי שפות אופקיות ואנכיות (כאן נעסוק רק בשפות אופקיות). פיקסל יוגדר כשייך <u>לשפה אופקית</u>, אם יש הפרש ניכר בין רמות האפור של הפיקסלים מעליו לבין אלה מתחתיו בסביבה "קטנה" k (בדרך כלל k יהיה 1 או 3). כיוון שבתמונות שלנו כל ערכי רמות האפור הן בין 0 (שחור) ל-255 (לבן), הוחלט ש"הפרש ניכר" הוא מעל **80**.

<u>בצורה מדויקת</u>, פיקסל (i,j) במטריצת תמונה A יוגדר כפיקסל שפה אופקית אם:

$$\frac{1}{2k+1}\sum_{t=-k}^{t=k}|A(i+1,j+t)-A(i-1,j+t)| > 80$$

הקלט לתוכנה המבוקשת הוא מטריצה **A** של ערכי אפור שבה **n** שורות ו-**m** עמודות, ואילו הפלט הוא מטריצה **edge** שגם בה **n** שורות ו-**m** עמודות. בכל פיקסל (i,j) <u>שנמצאה</u> בו שפה אופקית יוצב הערך **edge[i,j] = 0** ואילו בכל פיקסל <u>שלא נמצאה</u> בו שפה אופקית יוצב הערך **edge[i,j] = 255** (תמונה ב' מציגה את תמונת מטריצת השפות האופקיות **edge** עבור המטריצה **A** שבתמונה א').

(המשך הסעיף בעמוד הבא)

השלימו את שורות הקוד החסרות במגלה השפות <u>האופקיות</u> **horiz_edge** הנתון למטה.
פונקציה זו מקבלת מטריצה A מטיפוס Matrix שלמדנו וגודל סביבה k. בגמר האלגוריתם תוחזר המטריצה **edge** שמימדיה **n x m** כמוגדר למעלה.
אפשר להשתמש בפונקציה abs(x) שמחזירה ערך מוחלט של מספר.

```
def horiz_edge(A, k=1):
    n,m = A.dim()
    edge = Matrix(n,m,255) #a new nXm matrix with value  255 everywhere
    for i in range(1, n - 1):
        for j in range(k, m - k):
```

_____

_____

_____

_____

_____

_____

_____

_____

```
    return edge
```

ב. מה היא סיבוכיות הזמן, במונחים של O(...), של **horiz_edge** כפונקציה של מימדי מטריצת התמונה n, m (ניתן להתעלם מ- k, כי הוא מאד קטן) ? נמקו.

_____

_____

_____

ג. ייצוג אפשרי נוסף לשפות האופקיות הוא רשימה (list) של כל הפיקסלים שהתגלו כשפות אופקיות, כלומר רשימה (נקרא לה edgelist) שכוללת את זוגות הקואורדינטות של הפיקסלים שזוהו כשפות. לדוגמא, אם (20,6) הוא איבר ב- edgelist, אז יש שפה אופקית בשורה 20 ועמודה 6 של מטריצת התמונה. סדר הופעת האיברים ברשימה יכול להיות כלשהו. נסמן ב **p** את אורך הרשימה.

כיתבו פונקציה **pixel_count** שקולטת את הרשימה edgelist ומספר השורות -**n**, ומחזירה לכל אחת מ-**n** השורות בתמונה את מספר פיקסלי השפה הנמצאים עליה (לא כולם חייבים להיות רצופים, ראו תמונה ב'). הפלט יהיה רשימה באורך **n**, ששמה **rows**, כאשר **rows[i]** הוא מספר פיקסלי השפה בשורה **i**.

לדוגמא:

```
>>> pixel_count([(2,1),(1,1),(2,7),(1,3),(1,0)],4)
[0, 3, 2, 0]
```

<u>השתדלו לכתוב אלגוריתם יעיל ככל הניתן</u>. מירב הנקודות יינתן לפיתרון שסיבוכיות הזמן שלו היא רק **O(p+n)**, שהם גודל הקלט + גודל הפלט.

```
def pixel_count(edgelist, n):
```

_____

_____

_____

_____

_____

_____

_____

_____

```
    return rows
```

<u>שאלה 4 (20 נק')</u>

הפונקציה sum בפייתון מחשבת את סכום האברים ברשימה של מספרים.

לדוגמא:

```
>>> sum([2, 3, 7])
12
>>> sum([])
0
```

הפונקציה פועלת על ידי הפעלת פעולת החיבור שוב ושוב לקבלת סכום האברים:

$$(...((0 + x_0) + x_1) + ... x_k)$$

sum היא <u>פונקציה צוברת</u> שמפעילה פונקציה אסוציאטיבית על שני ארגומנטים (במקרה זה פעולת חיבור).

ניתן לכתוב <u>פונקציה צוברת כללית</u> acc, שמקבלת פונקציה f, ערך התחלתי v, ורשימה lst ומחזירה את תוצאת הצבירה.

לדוגמא, ניתן להגדיר בעזרת acc פונקציה mySum שפועלת על רשימה של מספרים ומחשבת את סכומם, כמו sum, באופן הבא:

```
def mySum(lst):
  return acc((lambda x,y: x + y), 0, lst)
```

א. השלימו הגדרה רקורסיבית של הפונקציה acc:

```
def acc(f, v, lst):
  if _____:
    return  _____
  else:
    return _____
```

ב. השלימו את הגדרת הפונקציה myProd אשר מקבלת רשימת מספרים ומחשבת את מכפלתם:

```
def myProd(lst):
  return acc( _____ )
```

ג. בסעיף זה נכתוב מימוש חדש לפונקציה compose שראינו בתרגיל בית ומוגדרת כך:

compose מקבלת רשימה של פונקציות ומחזירה פונקציה שמהווה הרכבה של הפונקציות ברשימת הקלט. למשל, עבור הקלט [f1, f2, f3] הפונקציה תחזיר את הפונקציה f1∘f2∘f3, שפועלת על x כך: f1(f2(f3(x))). הפונקציות ברשימה הן פונקציות של משתנה אחד – מקבלות משתנה אחד כקלט ומחזירות משתנה אחד כפלט.

תיקון בזמן המבחן: הפונקציה תחזיר את הפונקציה f3∘f2∘f1, שפועלת על x כך: f3(f2(f1(x)))

לדוגמא:

```
>>> compose([lambda x: x-1, lambda x: x*2, lambda x: x+1])(5)
11
>>> compose([])(5)
5
```

נתונה הגדרת הפונקציה compose שמשתמשת בפונקציות עזר g ו-h. השלימו את הגדרת הפונקציות g ו-h, כולל בחירת הארגומנטים:

```
def g(_____) :
    return _____


def h(_____):
    return _____


def compose(lst):
    return acc(g, h, lst)
```

שאלה 5 (20 נקודות)

מיון דלי – bucket sort – הוא מיון מיוחד למקרים בהם אנו יודעים מראש מהו הטווח בו מפוזרים המספרים. עבור קלט של n מספרים ממשיים בטווח [0, a), כלומר גדולים או שווים ל-0 וקטנים ממש מ-a, מיון דלי מתבצע לפי השלבים הבאים:

1. מייצרים רשימה חדשה באורך n שמאותחלת ל-n רשימות ריקות שהן ה-"דליים"
2. עוברים על המספרים בקלט ומחלקים אותם בין הדליים כך שהדלי הראשון יכיל מספרים בטווח [0, a/n), הדלי השני יכיל מספרים בטווח [a/n, 2a/n), וכך הלאה עד הדלי האחרון, שכיל מספרים בטווח [(n-1)a/n, a)
3. כעת ממיינים כל דלי ע"י מיון quicksort שנלמד בכיתה
4. לבסוף, משרשרים את הדליים הממוינים לפי הסדר לקבלת רשימה אחת ממויינת

שימו לב: בסעיפים א' ו-ב', נתחו את הסיבוכיות כפונקציה של n והתייחסו ל-a כאל קבוע.

א. נתחו את סיבוכיות הזמן של האלגוריתם עבור <u>המקרה הגרוע ביותר</u>. ציינו מהו מקרה זה ובחרו את הסיבוכיות ההדוקה ביותר מבין האפשרויות:

המקרה הגרוע ביותר: _____

1. $O(n)$ 2. $O(n\log n)$ 3. $O(n^2)$ 4. $O(n^2\log n)$ 5. $O(n^3)$

ב. נתחו את סיבוכיות הזמן של האלגוריתם עבור המקרה הטוב ביותר. ציינו מהו מקרה זה ובחרו את הסיבוכיות ההדוקה ביותר מבין האפשרויות:

המקרה הטוב ביותר: _____

1. $O(n)$ 2. $O(n\log n)$ 3. $O(n^2)$ 4. $O(n^2\log n)$ 5. $O(n^3)$

ג. השלימו את הקוד למימוש האלגוריתם בעמוד הבא. הקלט נמצא ברשימה lst, כל המספרים הם ממשיים בין 0 ל-a, לא כולל את a.

```
def bucket_sort(lst, a=100):
      n = len(lst)
      # init buckets
      buckets = ____________________________________________
      # scatter
      for ____________________ in ____________________:
            buckets[____________________].append(____________________)
      # sort and gather
      sorted_list = []
      for ____________________ in ____________________:
            if len(____________________) > ____________________:
                  # quicksort(lst) returns a sorted copy of lst
                  ____________________ = quicksort(____________________)
            sorted_list.extend(____________________)
      return ____________________________________________
```

סוף!