מס' ת"ז_____ מס' מחברת_____

מבחן בקורס מבוא מורחב למדעי המחשב, CS1001.py

סמסטר ב' 2013

מועד ב

מרצים: פרופ' חיים וולפסון, פרופ' עמירם יהודאי

מתרגלים: יואב רם, אמיר רובינשטיין

משך הבחינה: 3 שעות.

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

- יש לכתוב את כל התשובות בטופס הבחינה. המחברת תשמש כטיוטה בלבד ולא תיבדק.
- במבחן 11 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.
- אם לא צוין אחרת, אפשר להניח כי הקלט והארגומנטים לכל פונקציה תקינים.

טבלת ציונים: (לשימוש הבודקים)

שאלה	ניקוד
1	
2	
3	
4	
5	
סה"כ	

בהצלחה !

<u>שאלה 1 (20 נק')</u>

השאלה עוסקת בדחיסת זיו-למפל, עם ערכי ברירת המחדל שראינו בשיעור:

אורך חזרה מירבי 31, גודל חלון מירבי 4095. כמו כן דוחסים רק חזרות <u>באורך 3</u> או יותר.

א. להלן שני ייצוגי ביניים של דחיסת זיו-למפל. לכל אחד מהם עליכם לציין האם ייצוג ביניים כזה יכול להתקבל מהאלגוריתם שלמדנו. אם לא – עליכם להסביר מדוע לא. אם כן – עליכם לרשום את הטקסט המקורי לפני הדחיסה.

1. ['a', 'b', 'c', 'd', (3,5), 'x', (6,5), 'x']

_____

_____

_____

2. ['a', 'b', 'c', 'd', (3,20)]

_____

_____

_____

ב. השלימו את המחרוזת הבאה באורך 7 תווים, כך שיחס הדחיסה של אלגוריתם זיו-למפל עבורה יהיה 6/7.

a b c

____ ____ ____ ____ ____ ____ ____

הבהרה: יחס הדחיסה = מספר הביטים במחרוזת הדחוסה חלקי מספר הביטים במחרוזת ללא דחיסה.
הניחו כי ללא דחיסה כל תו מיוצג ע"י 7 ביטים.

<u>שאלה 2 (20 נק')</u>

השאלה עוסקת בקוד גילוי שגיאות המשלב קוד זוגיות (parity bit) עם קוד חזרות (repetition).

בהינתן הודעה באורך 3, נוסיף בסוף ביט זוגיות (XOR) עבור שלושת הביטים, ואז נשכפל את כל 4 הביטים.

בסה"כ, עבור הודעה של 3 ביטים משדרים 8 ביטים.

למשל (הביטים שנוספו מודגשים בקו):

הודעה		שדר
0 1 0	→	0 1 0 <u>1 0 1 0 1</u>
1 1 0	→	1 1 0 <u>0 1 1 0 0</u>

א. כמה שדרים חוקיים (codewords) יש? _____

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

_____

ג. להלן 3 שדרים שנתקבלו. באחד מהם לא נפלו שום שגיאות, באחד מהם נפלה שגיאה אחת, ובאחד נפלו שתי שגיאות. ציינו ליד כל שדר כמה שגיאות נפלו בו, ואם ניתן לשחזר את ההודעה המקורית - רישמו גם את ההודעה המקורית (3 ביטים).

כרגיל, אנחנו מניחים כי שגיאות בביטים שונים הן בלתי תלויות וכי ההסתברות לשגיאה בכל ביט נמוכה מאוד, ומתייחסים תמיד להודעה המקורית הסבירה ביותר.

שדר שנתקבל	מספר השגיאות	הודעה מקורית (אם אפשר לשחזר)
1 1 1 0 1 1 1 0	_____	_____
0 1 0 1 1 1 0 1	_____	_____
0 0 1 1 0 0 1 1	_____	_____

<u>שאלה 3 (20 נק')</u>

להלן תיאור של משחק פשוט בין שחקן אנושי למחשב:

נתונים שני שלמים חיוביים N1 ו-N2, כאשר N1<N2. השחקן האנושי בוחר מספר בין N1 ל-N2, <u>כולל</u> (מספר זה ייקרא "סוד"). המחשב מנסה לנחש את הסוד שוב ושוב, עד שהוא מצליח. כל ניחוש מתבצע ע"י קריאה לפונקציה guess שנתונה להלן. הארגומנט g הוא הניחוש של המחשב, והשחקן משיב למחשב האם הסוד שלו שווה לניחוש (ע"י הקלדת "="), גדול מהניחוש (ע"י הקלדת ">"), או קטן ממנו (ע"י הקלדת "<"). כאמור, המשחק ממשיך כל עוד המחשב לא הצליח לנחש את הסוד.

```
def guess(g):
    print("My guess is:", g)
    return input("Enter '=', '>' or '<' ")
```

מותר להניח שהשחקן לא מרמה, כלומר נותן תמיד תשובות נכונות, והמספר שבחר הוא אכן בטווח שצוין לעיל.

א. עליכם להשלים את הפונקציה find_secret(N1,N2), שמחזירה את הסוד שבחר השחקן, ועושה זאת בעזרת O(log(N2-N1)) ניחושים (כלומר קריאות לפונקציה guess).

```
def find_secret(N1,N2):
```

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

return _____

ב. נניח כעת כי לא ידועים N1 ו-N2, וכי הסוד אותו בוחר השחקן הוא מספר שלם חיובי כלשהו. עליכם להשלים את הפונקציה find_secret2 הבאה כך שתפתור בעיה זו.

הנחיות:
1. השתמשו בפונקציה find_secret מסעיף א'.
2. על התוכנית לבצע O(log S) ניחושים, כאשר S הוא המספר אותו בחר השחקן (הסוד). על הפתרון לבצע כמה שפחות ניחושים.

```
def find_secret2():
    low = 1
    high = 2
    answer = guess(_____)
    while _____:
        _____
        _____
        answer = guess(_____)

    if answer == "=":
        return _____
    return _____
```

<u>שאלה 4 (20 נק')</u>

שאלה זאת עוסקת במספרים שלמים, הגורמים הראשוניים שלהם, והמחלקים שלהם.

א. בסעיף זה נכתוב את הפונקציה factors שמקבלת מספר שלם n <u>גדול מ-1</u> ומחזירה את רשימת <u>הגורמים הראשוניים</u> שלו, לדוגמא:

```
>>> factors(12)
[2, 2, 3]
>>> factors(600)
[2, 2, 2, 3, 5, 5]
```

כפי שניתן לראות מהדוגמא, הרשימה מכילה מספרים ראשוניים שמכפלתם היא n (ולכן יכולות להיות חזרות), מסודרת בסדר עולה.
כדי לסייע למימוש הפונקציה, נוסיף לה פרמטר אופציונלי start שמציין שניתן להניח שאין ל-n גורמים קטנים מ-start.
השלימו את כתיבת המימוש <u>הרקורסיבי</u> של הפונקציה factors. תזכורת - הניחו ש-n גדול מ-1.

```
def factors(n,start=2):
    # assert - n does not have a factor that is smaller than start

    while _____ :
        start +=1

    if start>n:
        return _____

    else:
        return _____
```

בסעיפים הבאים נתייחס <u>לסכום המחלקים של מספר שלם</u>. אלה כל המספרים השלמים שהמספר מתחלק אליהם (מלבד המספר עצמו). נתונה הפונקציה sumDivisors שמקבלת כפרמטר מספר שלם גדול מ-1, ומחזירה את סכום המחלקים שלו. לדוגמא:

```
>>> sumDivisors(12)   # 1+2+3+4+6
16
>>> sumDivisors(13)   # 1
1
```

זוג מספרים שלמים שונים הם <u>ידידותיים</u> (amicable) אם כל אחד מהם שווה לסכום המחלקים של השני. זוג המספרים הידידותיים הקטן ביותר הוא (284 , 220):

```
>>> sumDivisors(220)   # 1+2+4+5+10+11+20+22+44+55+110
284
>>> sumDivisors(284)   # 1+2+4+71+142
220
```

ב. כיתבו פונקציה areAmicable שמקבלת שני מספרים שלמים ומחזירה ערך אמת אם הם ידידותיים, וערך שקר אם לא. השתמשו בפונקציה sumDivisors.

```
def areAmicable(m, n):
    _____
    _____
    _____
```

ג. בסעיף זה נגדיר גנרטור allAmicable שמייצר את הקבוצה האינסופית של זוגות מספרים ידידותיים. לדוגמא:

```
>>> p = allAmicable()
>>> next(p)
(220, 284)
>>> next(p)
(1184, 1210)
```

השלימו את המימוש של allAmicable:

```
def allAmicable():
    n = _____
    while _____:
        m = _____
        if sumDivisors(m) == n and _____:
            yield _____
        # increment
        _____
```

שאלה 5 (20 נקודות)

א.
בשאלה זו נגדיר מחלקה חדשה בשם Circle שתממש מספר תכונות של עיגול. העיגול יוגדר לפי נקודת המרכז שלו (a,b) והרדיוס שלו R. להזכירכם, עיגול הוא קבוצת הנקודות (x,y) שמרחקן מנק' המרכז (a,b) קטן או שווה ל-R – כלומר, שמקיימות את המשוואה $(a - x)^2 + (b - y)^2 \leq R^2$. בנוסף, היקף העיגול הוא $2\pi R$ ושטחו πR^2.

השלימו את הקוד בעמוד הבא. יש לממש את המתודות הבאות, ניתן להניח שהקלט תקין, מלבד במתודה __mul__, ניתן להשתמש בקבוע pi מתוך המחלקה math לפי הצורך:

- **__init__** – מקבלת ארגומנטים לפי הצורך ומציבה אותם בשדות R (רדיוס העיגול), a (קורדינטת ה- X של מרכז העיגול) ו-b (קורדינטת ה- Y של מרכז העיגול).
- **area** – אינה מקבלת ארגומנטים. מחזירה ערך מספרי ששווה לשטח העיגול (בקירוב).
- **circumference** – אינה מקבלת ארגומנטים. מחשבת ומחזירה את היקף העיגול (בקירוב).
- **__mul__** – מקבלת מספר כארגומנט (יש לוודא כי הקלט הוא אכן מספר שלם או ממשי בעזרת assert) ומחזירה עיגול חדש בעל אותו מרכז אך עם רדיוס גדול פי הארגומנט של המתודה.
- **move** – מקבלת tuple בשם point ובו שני מספרים אשר מייצגים קורדינטות x ו-y של נקודה ומזיזה את מרכז העיגול לנקודה זו. המתודה אינה מחזירה דבר.
- **contains** – מקבלת tuple בשם point ובו שני מספרים אשר מייצגים קורדינטות x ו-y של נקודה ומחזירה ערך אמת אם הנקודה נמצאת בתוך העיגול, וערך שקר אם הנקודה אינה בתוך העיגול. נקודה על שפת העיגול נחשבת כאילו היא בתוך העיגול.
- **intersect** – מקבלת עיגול נוסף כארגומנט, ומחזירה True אם יש לשני העיגולים שטח משותף, ו-False אחרת. השקה (מגע בין שפות העיגולים) אינה נחשבת לשטח משותף, הכלה נחשבת לשטח משותף.

דוגמאות לשימוש:

```
>>> A = Circle(R=3, a=0, b=0)
>>> A
Circle center (0.000000,0.000000) with radius 3.000000
>>> A.area(), A.circumference()
28.274333882308138, 18.84955592153876
>>> A.contains(0,4)
False
>>> A.move(0,1)
>>>A.contains(0,4)
True
>>> A.intersect(A*2)
True
>>> A.intersect(Circle(1,4,0) )
False
```

```
import math
class Circle:
    # Constructor:
    def __init__(self, ____________________________________________):

        self.R = ____________________________________________

        self.a = ____________________________________________

        self.b = ____________________________________________

    def __repr__(self):
        return "Circle center " + self.a + ", " + self.b + " with radius " + self.radius

    def area(self):

        return ____________________________________________

    def circumference(self):

        return ____________________________________________

    def __mul__(self, factor):

        assert ____________________________________________

        return ____________________________________________

    def move(self, point):

        ____________________________________________

        ____________________________________________


    def contains (self, point):

        ____________________________________________

        ____________________________________________

        ____________________________________________


    def intersect(self, other):

        ____________________________________________

        ____________________________________________

        ____________________________________________

        ____________________________________________
```

(המשך שאלה 5)

כפי שראינו בשיעור, ניתן לייצג תמונה ע"י מטריצה A בעלת n שורות ו-m עמודות. ערכי המטריצה מייצגים פיקסלים בעלי רמות אפור הנעות מערך מינימלי 0 (שחור) לערך מקסימלי 255 (לבן).

עליכם לכתוב מתודה חדשה paint למחלקה Circle, אשר תקבל מטריצה בשם img בגודל nxm (ניתן להניח שגודל המטריצה מספיק כדי להכיל את העיגול – n,m>2R), תמקם את מרכז העיגול במרכז המטריצה ותשחיר את הפיקסלים המתאימים - לאחר הפעלת המתודה כל הפיקסלים שבתוך העיגול יקבלו ערך 0. המתודה אינה מחזירה דבר אלא משנה את img.

מירב הניקוד יינתן לאלגוריתם שסיבוכיותו $O(R^2)$.

ב.

ציינו מה סיבוכיות הפתרון שלכם, ונמקו בקצרה.

_____

_____

_____

_____

ג.

השלימו את הקוד של paint בעמוד הבא.

```
def  paint (self, img):
    n,m = img.dim()
```

סוף!