מס' סידורי:_____ ת.ז.:_____

בחינה במבוא מורחב למדעי המחשב
בשפת פייתון
0368.1105.07

סמסטר ב', מועד ב' ,תשע"ב

12/08/2012

אוהד ברזילי, אמיר רובינשטיין

הוראות (נא לקרוא!)

- משך הבחינה **שלוש שעות**, חלקו את זמנכם ביעילות.
- בבחינה 11 עמודים והיא כוללת 5 שאלות (מרובות סעיפים)
- חומר עזר מותר: 2 דפי A4 כתובים משני צידיהם.
- אנא נצלו את מחברת הבחינה לרישומי טיוטה, והעתיקו את התשובות הסופיות לטופס בחינה זה. תשובות במחברת הבחינה לא תיבדקנה. יש לענות על כל השאלות בטופס זה במקום המיועד לכך. המקום המיועד מספיק לתשובות מלאות, ויותר, ואולם במידת הצורך ניתן לכתוב בגב טופס הבחינה. יש לצרף את טופס המבחן למחברת הבחינה.
- יש למלא מספר סידורי (מס' מחברת) ומספר ת.ז על כל דף של טופס הבחינה.

- ניתן להניח לאורך הבחינה שכל החבילות הדרושות יובאו, ואין צורך לכתוב שורות `import`.
- במקומות בהם תתבקשו לכתוב מתודה או פונקציה, ניתן (ומומלץ!) לכתוב גם פונקציות עזר, אלא אם צוין במפורש אחרת.
- ניתן להסתמך על פונקציות אשר מומשו בשאלות אחרות ובסעיפים אחרים גם אם לא עניתם על אותו הסעיף

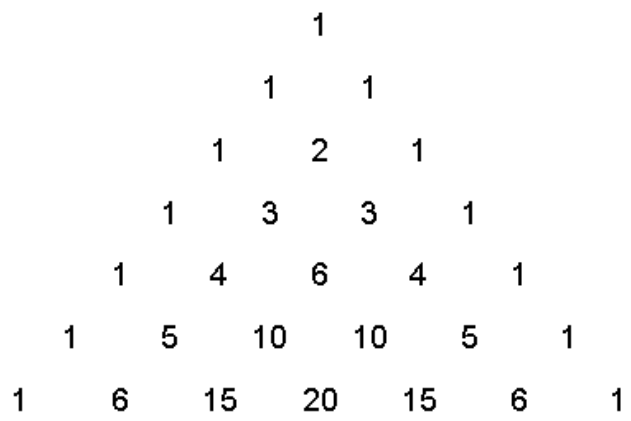
לשימוש הבודקים:

שאלה	א	ב	ג	ד	סה"כ
1					
2					
3					
4					
5					

שאלה 1 (20 נקודות)

משולש פסקל הוא משולש אינסופי המורכב משורות של מספרים, כך שבשורה ה- n, האיבר ה- r מוגדר כ:
$a_{nr} = \binom{n}{r} = \frac{n!}{r!(n-r!)}$.

7 השורות הראשונות במשולש הן כדלהלן:



נשים לב שניתן לקבל את השורה ה-i במשולש פסקל מהשורה ה-i-1 באופן הבא:

1. האיבר הראשון והאיבר האחרון בשורה הם תמיד 1.
2. פרט לכך, האיבר ה-j בשורה ה-i הוא סכום האיבר ה-j-1 והאיבר ה-j בשורה ה-i-1.

נייצג כל שורה במשולש על ידי רשימה כך שהשורה ה-0 היא הרשימה [1], השורה ה-1 היא הרשימה [1,1] וכו'.

א. (6 נקודות) ממשו את הפונקציה **next_row(row)**, המקבלת את השורה ה-i-1 במשולש ומחזירה את השורה ה-i:

```
def next_row(row):
```

ב. (6 נקודות) נייצג את משולש פסקל כזרם אינסופי של רשימות בצורה הבאה:

```
[[1], [1,1], [1,2,1], [1,3,3,1],…
```

ממשו את הפונקציה `generate_pascal()` המחוללת את משולש פסקל (שימו לב זהו generator):

```
def generate_pascal():
```

ג. (8 נקודות) משולש ברנולי הוא המשולש שבו כל שורה היא רשימת הסכומים החלקיים של המקדמים הבינומיים, כלומר $a_{nk} = \sum_{i=0}^{k} \binom{n}{i}$.

במילים אחרות, כל שורה במשולש ברנולי היא **רשימת <u>הסכומים החלקיים</u> של השורה המתאימה במשולש פסקל**. כלומר, המספר ה- i בשורה ה- j של משולש ברנולי הוא סכום i המספרים הראשונים של השורה ה-j במשולש פסקל.

7 השורות הראשונות במשולש ברנולי הן כדלהלן:

1
1 2
1 3 4
1 4 7 8
1 5 11 15 16
1 6 16 26 31 32
1 7 22 42 57 63 64

נייצג את משולש ברנולי בצורה הדומה למשולש פסקל, כזרם אינסופי של רשימות:

```
[[1], [1,2], [1,3,4], [1,4,7,8] …
```

מס' סידורי:_____ ת.ז.:_____

ממשו את הפונקציה `generate_bernoulli()` המחוללת את משולש ברנולי (שימו לב זהו generator):

```
def generate_bernoulli():
```

שאלה 2 (20 נקודות)

חברת ההזנק "לחישות פזיזות" מתמחה בייצור מבני נתונים המבוססים על פונקציות גיבוב (hash functions) פורצות דרך.

הפונקציות **hash1** ו- **hash2** מייצרות ערך hash מספרי עבור רשימות של מחרוזות (שימו לב כי שתי הפונקציות משתמשות בפונקצית ה-**hash** הסטנדרטית של Python כפונקצית עזר):

```
def hash1(lst):
      hash_list = [hash(word) for word in str(lst)]
      return sum(hash_list) // len(hash_list)

def hash2(lst) :
      concat = "".join(lst)
      return hash(concat)
```

א. (5 נקודות) בהינתן כי פונקציות הגיבוב הנתונות ישמשו לאחסון n מחרוזות באורך ממוצע m, העריכו מה תהיה סיבוכיות זמן הריצה של hash1 (עבור ריצה בודדת). אם לצורך החישוב ברצונכם להתייחס לנתונים נוספים, ציינו זאת במפורש:

ב. (5 נקודות) בהינתן כי פונקציות הגיבוב הנתונות ישמשו לאחסון n מחרוזות באורך ממוצע m, העריכו מה תהיה סיבוכיות המקום של hash2 (עבור ריצה בודדת). אם לצורך החישוב ברצונכם להתייחס לנתונים נוספים, ציינו זאת במפורש:

ג. (10 נקודות) איזו משתי פונקציות הגיבוב עדיפה לצורך אחסון רשימות של מחרוזות במבני נתונים? תנו דוגמא המצדיקה את תשובתכם:

שאלה 3 (30 נקודות)

בשאלה זו נממש פונקציה בשם find_steady שמקבלת כקלט רשימה L של מספרים שלמים, שונים זה מזה, ממוינת בסדר עולה, ומחזירה ערך $i \geq 0$ המקיים $L[i] == i$. אם אין i כזה יוחזר None. אם יש יותר מאינדקס אחד המקיים את התנאי אתם רשאים להחליט איזה מהערכים להחזיר.

לדוגמא:

```
find_steady([3,5,9,17]) => None
find_steady([-3,0,2,10]) => 2
```

א. (5 נקודות) הסבירו את האלגוריתם **במילים**, וציינו מהי סיבוכיות זמן הריצה (בסימון $O(\ldots)$, כפונקציה של אורך הרשימה n). על הפונקציה לרוץ בזמן יעיל. זמן ריצה לינארי יזכה בניקוד חלקי בלבד.

ב. (5 נקודות) תנו דוגמה לרשימה ממוינת בסדר עולה שבה המספרים לא בהכרח שונים זה מזה, ועבורה הפתרון שלכם בסעיף א' לא עובד נכון. ציינו מה יחזיר האלגוריתם ומדוע פלט זה אינו נכון.

ג. (10 נקודות) ממשו את הפונקציה בפייתון.
אתם רשאים לשנות את חתימת הפונקציה, כך שתקבל ארגומנטים נוספים עם ערכי ברירת מחדל.

הפונקציה count_steadies מקבלת כקלט רשימה L של מספרים שלמים שונים זה מזה, ממוינת בסדר עולה, ומחזירה את מספר האינדקסים i, עבורם מתקיים $i \geq 0$ ו- $L[i] == i$.

לדוגמא:

```
count_steadies([3,5,9,17]) => 0        # no steadies
count_steadies ([-3,0,1,3,4,11]) => 2  # found 2 steadies
```

ד. (10 נקודות) ממשו את הפונקציה **count_steadies** בפייתון.

על הפונקציה לרוץ בזמן יעיל. זמן ריצה לינארי (ביחס לאורך הקלט) יזכה בניקוד חלקי בלבד.

אתם רשאים להיעזר בפונקציה שהוגדרה בסעיף הקודם (לא חובה).

שאלה 4 (15 נקודות)

לפניכם הפונקציה is_prime אשר נלמדה בהרצאה:

```
def is_prime(m, show_witness=False):
    for i in range(0,100):
        a = random.randint(1,m-1) # a is a random integer in [1..m-1]
        if pow(a,m-1,m) != 1:
            if show_witness:  # caller wishes to see a witness
                print(m,"is composite","\n",a,"is a witness")
            return False
    else:
        return True
```

א. (5 נקודות) ציינו לגבי הטענה הבאה האם היא אמת או שקר. אם ציינתם שהמשפט הוא שקרי יש צורך לנמק:

הפונקציה is_prime עשויה להחזיר שהמספר הנתון m הוא פריק אף על פי שהוא בעצם ראשוני, בהסתברות של $\left(\frac{1}{4}\right)^{100}$

ב. (10 נקודות) דני פצחני יודע שחוזקן של מערכות הצפנה רבות תלוי בקושי לפרק מספר לגורמים ראשוניים, והוא זומם להשתמש בפונקציה is_prime כדי לפצח הצפנות שכאלה. להלן תוכניתו הזדונית:

בהינתן מספר m אשר יש לפרקו לגורמים – הוא יקרא לפונקציה is_prime(m,True).
אם המספר אינו ראשוני הפונקציה תדפיס גם את העד (witness).
כאשר דני יחלק את המספר m ב-witness שהודפס הוא יקבל את הגורם השני של המספר m וכך יפצח את הקוד.

האם תוכניתו של דני תצליח לפרק מספר פריק $m > 2^{1000}$ לגורמים? אם לא, נמקו.

שאלה 5 (15 נקודות)

נתונה מחרוזת טקסט s המכילה את התווים $\{x_1, x_2, x_3 ... x_k\}$.

נתון כי התפלגות התווים בטקסט s מקיימת את הכלל הבא: התו x_i מופיע בטקסט 2^i פעמים.

יהי T עץ האפמן אשר נבנה מהתפלגות התווים ב-s (בעמוד הבא מצורף הקוד לבניית העץ שראינו בהרצאה).

א. ציירו את עץ האפמן T (אין צורך לתאר את שלבי בנייתו)

ב. מהו אורך הקידוד הקצר ביותר המוקצה לתו בודד שמגדיר T? לאיזה תו או תווים יש קידוד באורך זה?

ג. מהו אורך הקידוד הארוך ביותר המוקצה לתו בודד שמגדיר T? לאיזה תו או תווים יש קידוד באורך זה?

קוד לבניית עץ האפמן:

```
def calculate_distribution(text):
   dist = {}
   for x in text:
      dist[x] = dist.get(x,0) + 1
   return dist


# the argument dist is the output of calculate_distribution above
def build_huffman_tree(dist):
      queue = [(px, x) for x, px in dist.items()]
      while len(queue) > 1:
            print (queue)
            px, x = extract_min(queue)
            py, y = extract_min(queue)
            queue.append((px+py, [x,y])) # create intermediate node
      px, x = extract_min(queue) # only root node left
      print (queue)
      return x

def extract_min(queue):
      P = [px for px,x in queue]
      return queue.pop(P.index(min(P)))
```