מס' ת"ז_____ מס' מחברת_____

מבחן בקורס מבוא מורחב למדעי המחשב, CS1001.py
עם תיקונים קלים והבהרות שניתנו במהלך המבחן

ביה"ס למדעי המחשב, אונ' תל אביב

סמסטר ב' 2015-16, מועד א, 15/7/2016

מרצים: ד"ר דניאל דויטש, אמיר רובינשטיין

מתרגלות: יעל ברן, מיכל קליינבורט, אמיר גלעד

משך הבחינה: 3 שעות.

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

- במבחן 14 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.

בהצלחה !

טבלת ציונים: (לשימוש הבודקים)

שאלה	ערך	ניקוד
1	25	
2	20	
3	15	
4	20	
5	20	
סה"כ	100	

שאלה 1 (25 נק')

להלן הגדרה רקורסיבית: $S(n,k) = S(n-1,k-1) + k \cdot S(n-1,k)$. כמו כן, מתקיים

1. $S(n,n) = 1$
2. $S(n,k) = 0$ לכל $k > n$
3. $S(n,0) = 0$ לכל $n > 0$

(המספרים המוגדרים כך ידועים כמספרי Stirling אך אין לכך כל חשיבות בשאלה זו).

א. השלימו את הפונקציה הרקורסיבית `stirling_mem` כך שפונקציית המעטפת `stirling` שמקבלת מספרים טבעיים n, k כקלט תחזיר את $S(n,k)$. על הפונקציה להשתמש בממואיזציה.
יש למלא קוד בשורות החסרות בלבד.
דוגמאות הרצה:

```
>>> stirling (1,1)
1
>>> stirling (3,2)
3
>>> stirling (3,4)
0
```

```
def stirling(n, k):
    d = dict()
    return stirling_mem(n,k,d)
```

```
def stirling_mem(n, k, d):
    if _____ or _____:
        return _____
    elif _____:
        return _____
    elif _____:
        return _____

    _____

    return _____
```

ב. מהו עומק עץ הרקורסיה (המסלול הארוך ביותר מהשורש לעלה) עבור חישוב $S(n,k)$ באמצעות הפונקציה `stirling` כאשר $n \geq k$? תנו תשובה הדוקה ככל שתוכלו.

כתבו את תשובתכם כתלות ב- n,k ובמונחי $O(\ldots)$ והסבירו בקצרה.

O(_____)

הסבר:

ג. כעת נגדיר באמצעות $S(n,k)$ את הסדרה $B(n)$ באופן הבא: $B(n) = S(n,0) + S(n,1) + \cdots + S(n,n)$

(מספרים אלו ידועים כמספרי Bell אך אין לכך כל חשיבות בשאלה זו).

השלימו את הפונקציה `bell_stirling` שמקבלת מספר טבעי n ומחזירה את $B(n)$. על הפונקציה להשתמש בפונקציה `stirling`.

דוגמאות הרצה:

```
>>> bell_stirling(1)
1
>>> bell_stirling(2)
2
>>> bell_stirling(3)
5
>>> bell_stirling(4)
15
```

```
def bell_stirling(n):
    _____
    _____:
    _____
    return _____
```

ד. ידועה גם הנוסחה הרקורסיבית לחישוב $B(n)$: $B(n) = \sum_{k=1}^{n} \binom{n-1}{k-1} B(n-k)$. כמו כן, $B(1) = B(0) = 1$. היעזרו בפונקציה `binom_mem` הנתונה לכם (המחשבת את $\binom{n}{k}$, כפי שראינו בתרגול), והשלימו את מימוש הפונקציה `bell_mem`, כך שהפונקציה `bell_rec` תחשב את $B(n)$ בהינתן מספר טבעי n כקלט. על הפונקציה להשתמש בממואיזציה.

```
def binom_mem(n, k, d):
   if k < 0 or n < 0 or n < k :
       return 0
   if k == 0 or k == n:
       return 1
   elif (n,k) in d:
       return d[(n,k)]
   d[(n,k)] = binom_mem(n-1,k,d) + binom_mem(n-1,k-1,d)
   return d[(n,k)]


def bell_rec(n):
   d1 = dict()
   d2 = dict()
   return bell_mem(n,d1,d2)
```

```
def bell_mem(n, d1, d2):

   if _____ or _____:

      return _____

   elif _____:

      return _____

   else:

      _____

      for k in range (_____,_____):

         _____

      _____

      return _____
```

שאלה 2 (20 נק')

בשאלה זו נממש חיפוש של תת-מחרוזות משותפות לשתי מחרוזות טקסט באמצעות אלגוריתם Karp-Rabin וגנרטורים.

נתונות שתי הפונקציות הבאות (הראשונה זהה והשניה דומה מאד לקוד של KR מהשיעור):

```
def fingerprint(string, basis=2**16, r=2**32-3):
# identical to code from class
   """ used to computes karp-rabin fingerprint of the pattern
   employs Horner method (modulo r) """
   partial_sum=0
   for x in string:
       partial_sum=(partial_sum*basis+ord(x)) % r
   return partial_sum

def slide(prev_fp,prev_char,next_char,b_power,basis=2**16,r=2**32-3):
     new_fp=((prev_fp-ord(prev_char)*b_power)*basis+ord(next_char)) % r
     return new_fp
```

א. השלימו את פונקצית הגנרטור הבאה אשר מייצרת את רצף ה- fingerprints של תת מחרוזות באורך length ממחרוזת נתונה text, על פי סדר הופעתן במחרוזת text. קריאה ל- next כאשר תמו ה- fingerprints תגרום לזריקת exception, כרגיל כפי שראינו בכיתה.

<u>דרישות נוספות:</u> על הפונקציה לעשות שימוש בפונקציות fingerprint ו slide. במהלך ייצור רצף ה fingerprints המלא עבור מחרוזת נתונה תקרא הפונקציה fingerprint <u>פעם אחת בלבד</u>.

לדוגמה:

```
>>> list(kr_gen("bye bye",3))
[7930251, 6619531, 2097553, 6422745, 7930251]
```

```
def kr_gen(text, length, basis=2**16, r=2**32-3):
     if length<=len(text):
          _____________________________________________

          _____________________________________________

          b_power = pow(basis,length-1,r)

          for s in range(1,len(text)-length+1):

               ____________________________________

               ____________________________________
```

ב. השלימו את פונקצית הגנרטור הבאה, אשר מקבלת שתי מחרוזות טקסט ומייצרת את האינדקסים של כל תת המחרוזות באורך length המשותפות להן. ליתר דיוק, הפונקציה מייצרת רצף של tuples, כשכל tuple מייצג match ומכיל זוג מספרים: המספר הראשון הוא אינדקס המופע במחרוזת הראשונה, והשני במחרוזת השניה. אין חשיבות לסדר ייצור ה tuples (אבל, כאמור, יש חשיבות לסדר המספרים בתוך כל tuple). זיכרו כי כרגיל בעת שימוש ב-fingerprints לצורך השוואת מחרוזות, תיתכן שגיאה בהסתברות קטנה, בדומה לקוד שראינו בשיעור. אין צורך לטפל בבעיה זו.

<u>דרישות נוספות:</u>

1. על הפונקציה לעשות שימוש ב- kr_gen.
2. אסור לפונקציה להקצות זיכרון בגודל שהוא לינארי באורכי מחרוזות הטקסט או יותר מכך. בפרט, אסור לפונקציה בשום שלב לפרוש את רצף ה fingerprints המלא של אחת ממחרוזות הטקסט.

לדוגמה:

```
>>> list(generate_shared_substrings("abcdef","xcdefx",3))
[(2, 1), (3, 2)]
```

תזכורת מהשיעור - try ו except:

```
def division(a,b):
    try:
        return a/b
    except ZeroDivisionError:
        print("division by zero")
```

```
def generate_shared_substrings(text1, text2, length):
    _____________________________________________
    i1 = -1
    while True:
        _____________________________________________
        i1 += 1
        _____________________________________________
        i2 = -1
        while True:
            try:
                _____________________________________
                i2 += 1
            except StopIteration:
                _____________________________________
            if fp1 == fp2:
                yield (i1,i2)
```

שאלה 3 (15 נק')

נתון הקוד הבא לניקוי תמונות שכולל את הפונקציות שנלמדו בשיעור, וגרסאות מעט שונות שלהן. הפונקציות בעמוד זה זהות לאלו שראינו בכיתה, ואילו בעמוד הבא ישנן פונקציות חדשות.

```
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


def average(lst):
     l = len(lst)
     return round(sum(lst)/l)


def median(lst):
     sort_lst = sorted(lst)
     l = len(sort_lst)
     if l%2==1:     # odd number of elements. well defined median
          return sort_lst[l//2]
     else:          # even number of elements. average of middle two
          return (int(sort_lst[-1+l//2]) + int(sort_lst[l//2])) // 2


def local_operator(A, op, k=1):
     n,m = A.dim()
     res = copy(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 local_means(A, k=1):
     return local_operator(A, average, k)


def local_medians(A, k=1):
     return local_operator(A, median, k)
```

```
### New code starts here

def local_operator_new(A, op, k=1):
 n,m = A.dim()
 for i in range(k,n-k):
     for j in range(k,m-k):
         A[i,j] = op(items(A[i-k:i+k+1,j-k:j+k+1]))
 return A

def local_medians2(A, k=1):
 return local_operator_new(A, median, k)


def noisy_pic():
   im = Matrix(10,10)  # initialized to 0 (black)
   for i in range(0,10,2):
       for j in range(0,10):
           im[i,j] = 255 # white
   return im
```

תזכורת: הפיקסלים בתמונה מסודרים כך שהפיקסל במקום (0,0) הוא בפינה השמאלית העליונה.

נתונות 4 הפקודות הבאות:

1. noisy_pic()
2. local_means(noisy_pic())
3. local_medians(noisy_pic())
4. local_medians2(noisy_pic())

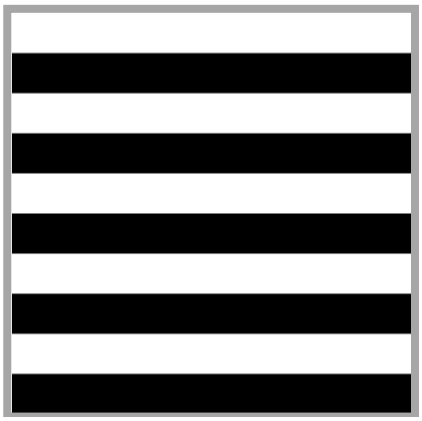
בעמוד הבא מופיעות 4 תמונות (שימו לב: הוספנו מסגרת אפורה).

התאימו את התוצאות לפקודות, כלומר כתבו ליד כל תמונת תוצאה את מספר הפקודה שתוביל אליה.

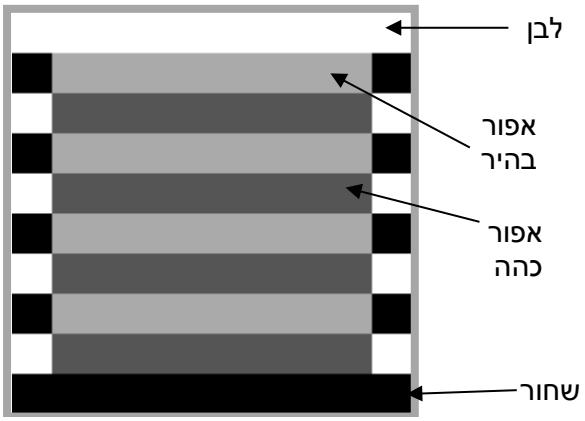
הסבירו בקצרה את תשובתכם.

הסבר:

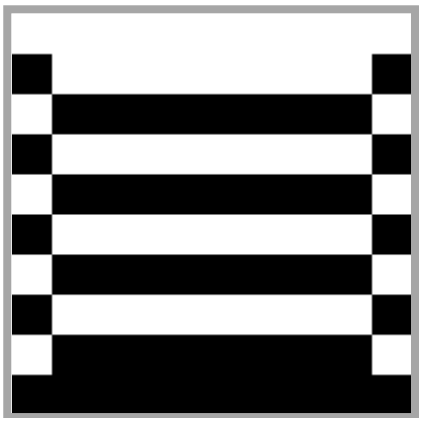
מס' ת"ז_____ מס' מחברת____



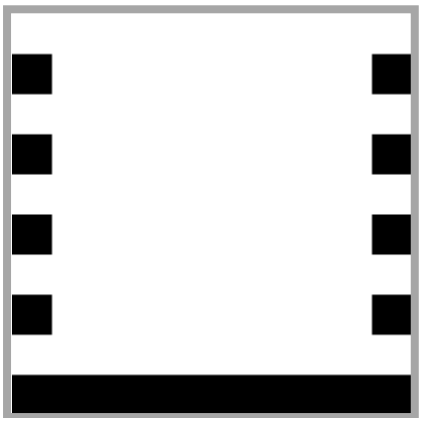
פקודה מס': ____



פקודה מס': ____



פקודה מס': ____



פקודה מס': ____

<u>שאלה 4 (20 נק')</u>

תהי $f: \mathbb{N}^+ \to \mathbb{Z}$ פונקציה מהשלמים החיוביים (גדולים ממש מ-0) לשלמים.

נאמר ש- f גדלה מונוטונית אם לכל ערך x מתקיים ש $f(x+1) > f(x)$.

בהינתן פונקציה f שגדלה מונוטונית, נרצה למצוא את הערך n המינימלי עבורו f מחזירה ערך גדול ממש מ-0 . (מאחר ש- f גדלה מונוטונית אז מתקיים לכל $x > n$ ש- $f(x) > 0$ וכמו כן לכל $x < n$ מתקיים ש $f(x) < 0$.)

בשאלה זו הניחו כי סיבוכיות הזמן עבור הפעלת f על קלט כלשהו הינה $O(1)$.

א. השלימו את מימוש הפונקציה `find_first_positive1` שמקבלת כקלט פונקציה $f: \mathbb{N}^+ \to \mathbb{N}$ הגדלה מונוטונית ומחזירה את הערך n המינימלי עבורו f מחזירה ערך גדול ממש מ- 0. שימו לב כי בסעיף זה המימוש הוא נאיבי, ללא נסיון לייעל את זמן הריצה.

```
def find_first_positive1(f):
    n = 1
    while  _____________________________________________:
        _____________________________________________
    return n
```

ב. מהי סיבוכיות זמן הריצה של `find_first_positive1` כפונקציה של הפלט n, כאשר n הינו הערך המינימלי שמקיים כי $f(n) > 0$? תנו תשובה במונחי סדר גודל $O(...)$, הדוקה ככל שתוכלו:

O(_____)

ג. נתונה הפונקציה הבאה `what`.

```
def what(f, a, b):
   if b < a:
     return None
   c = a + (b - a)//2
   if f(c) > 0 and (c == a or f(c-1) <= 0):
       return c
   if f(c) <= 0:
       return what(f, c + 1, b)
   return what(f, a, c - 1)
```

יוליוס הריץ את הפקודות הבאות:

```
>>> f = lambda x: x-10
>>> x = what (f, 200, 1000)
>>> y = what (f, 2, 10)
>>> z = what (f, 5, 15)
```

מה יהיו ערכי המשתנים x, y, z לאחר הרצת הפקודות?

x: _____	y: _____	z: _____

נמקו בקצרה את תשובתכם:

ד. השלימו את הפונקציה `find_first_positive2` שמקבלת כקלט פונקציה $f: \mathbb{N}^+ \to \mathbb{Z}$ שגדלה מונוטונית ומחזירה את הערך n המינימלי עבורו f מחזירה ערך גדול ממש מ-0. על הפונקציה להשתמש בפונקציה `what` מסעיף ג'. על הפונקציה `find_first_positive2` לרוץ בסיבוכיות זמן $O(\log n)$, כאשר n הינו הערך המינימלי שמקיים כי $f(n) > 0$.

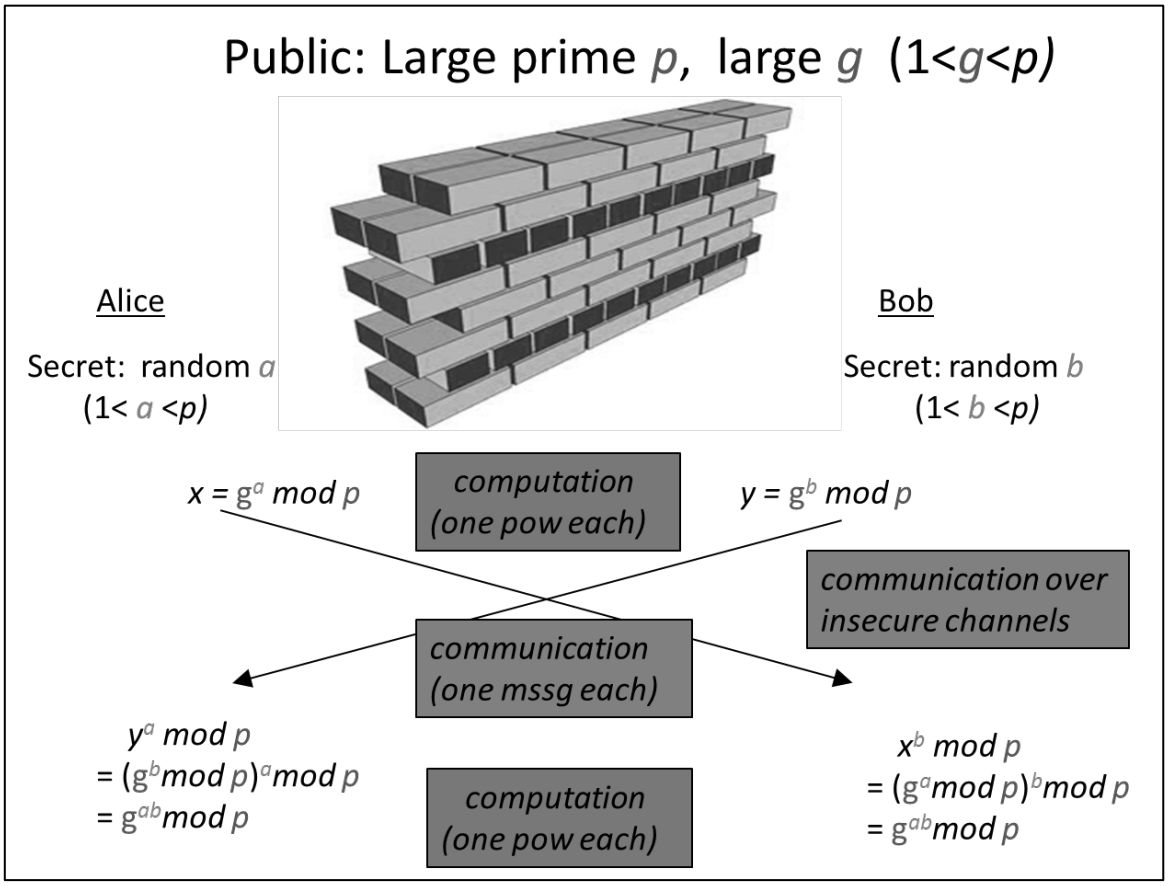
```
def find_first_positive2(f):
    _____________________________________________________
    while_____________________________________________________:
        _____________________________________________________
    return _____________________________________________________
```

הסבירו ב-2-3 שורות מדוע הפונקציה שכתבתם עונה לדרישות הסיבוכיות.

שאלה 5 (20 נק')

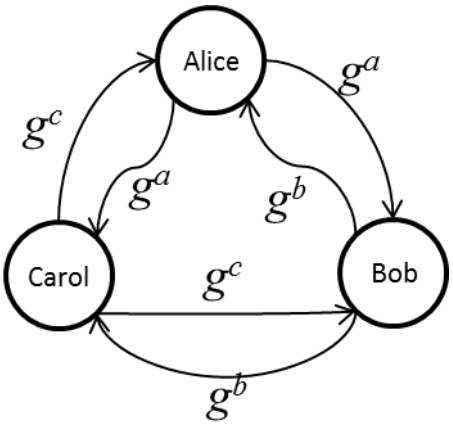
שאלה זו דנה בהרחבת פרוטוקול דיפי-הלמן להחלפת מפתח סודי, ליותר משני משתתפים. נסמן את מספר המשתתפים המעוניינים לייצר להם סוד משותף ב- N.

תזכורת לפרוטוקול דיפי-הלמן לשני משתתפים (שקף מתוך ההרצאה):

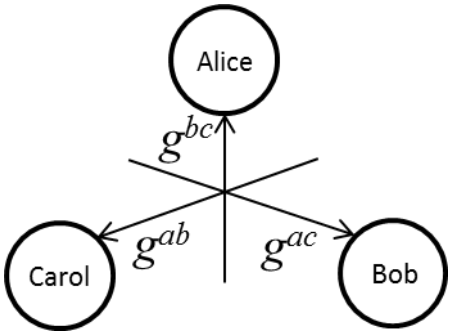


כל החישובים נעשים מודולו p ראשוני כלשהו שנבחר מראש. לשם פשטות, בשאלה זו לא נציין זאת בביטויים שבהמשך. כלומר, בכל מקום שכתוב למשל g^a הכוונה היא ל- $g^a \bmod p$.

א. נניח N=3. להלן תיאור פרוטוקול מורחב אפשרי. בשלב הראשון מתבצע הפרוטוקול כפי שלמדנו בכיתה, בין כל זוג משתמשים. לדוגמה, Alice מחשבת את g^a (פעם אחת), ושולחת זאת ל- Bob ול- Carol. תרשים הודעות שנשלחות:



כעת Alice ו- Bob חולקים את הסוד g^{ab}, Alice ו- Carol את הסוד g^{ac}, ואילו Bob ו- Carol את הסוד g^{bc}. כל זוג שולח את הסוד המשותף שלו למשתמש השלישי. למשל, Alice ו- Bob שולחים (אחד מהם או שניהם, לא משנה) ל- Carol את g^{ab}, ובדומה גם שאר הזוגות:



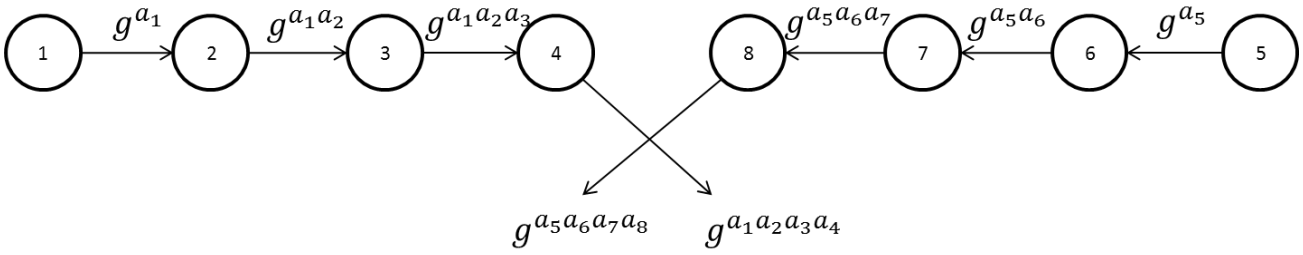
וכל משתמש יכול כעת לחשב את הסוד המשותף g^{abc}.

כמה פעולות modular exponentiation מבצע כל משתמש? _____

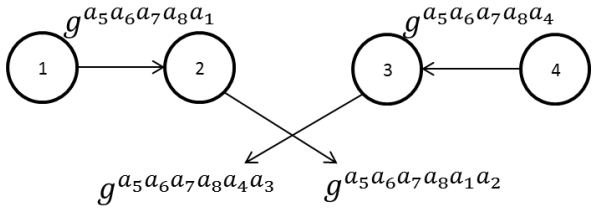
ב. תארו במילים פרוטוקול דומה, בו כל משתמש מבצע **3** פעולות modular exponentiation בלבד. ציינו אילו הודעות נשלחות בכל שלב. אפשר להיעזר באיור בדומה לאיורים לעיל.

ג. להלן תיאור של פרוטוקול מורחב עבור N=8 משתתפים. נסמן את הסוד הפרטי של משתתף i ב-a_i.

<u>שלב 1</u>: מחלקים את המשתמשים ל- 2 קבוצות שוות, ושולחים את ההודעות הבאות:



שלב 2: מחלקים כל קבוצה באופן דומה וחוזרים על התהליך, כאשר בכל קבוצה ההודעה ההתחלתית היא ההודעה שנשלחה בסוף השלב הקודם מהקבוצה השנייה. למשל, משתתפים 1, 2, 3 ו- 4 מחולקים שוב וחוזרים על התהליך, עם ההודעה ההתחלתית $g^{a_5a_6a_7a_8}$:



1) איזו הודעה ישלח משתתף 1 ל- 2 בשלב השלישי (והאחרון)? _____

2) איזו פעולה יעשה משתתף 2 לאחר השלב השלישי, כדי לחשב את הסוד המשותף? _____

3) מהו הסוד המשותף לכל 8 המשתתפים? _____

4) עבור N משתתפים, כמה פעולות modular exponentiation מבצע כל משתתף? יש לתת תשובה במונחים של O, הדוקה ככל שניתן.

O(_____)

דף נוסף למקרה הצורך