מבחן בקורס "מבוא כללי לתכנות", מקבץ הסייבר

סמסטר א' 2014

מועד א

מרצה: אמיר רובינשטיין

משך הבחינה: שעה וחצי

חומר עזר מותר: כל חומר עזר, למעט אלקטרוני (מחשב, מחשבון) וביולוגי (חברים)

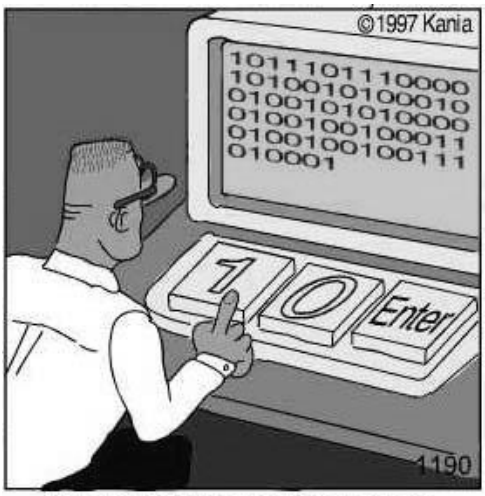
- יש לכתוב את כל התשובות בטופס בחינה זה. מחברות הטיוטה לא ייאספו.
- יש לכתוב את כל התשובות במקום המוקצב ובכתב קריא. חריגות משמעותיות מהמקום המוקצב, תשובות הכתובות בכתב קטן / לא ברור או תשובות שדורשות מאמצים רבים להבנתן עלולות לגרור הורדת ציון.
- במבחן 7 עמודים ו- 7 שאלות – בידקו שכולם בידיכם.
- מומלץ לא "להיתקע" על אף שאלה בודדת, אלא להמשיך לשאלות אחרות ולחזור לשאלה אח"כ.
- קיראו היטב את השאלות. הקפידו לענות בדיוק על מה שנשאלתם, ולנמק אם נדרשתם.

נא לרשום מספר תעודת זהות (ללא שם):

בהצלחה !

שאלה 1 (10 נק')

להלן בדיחה קצרה:



Real programmers code in binary.

מתכנתים אמיתיים כותבים קוד בבינארית.

הסבירו בקצרה את הבדיחה, תוך שימוש נכון במושגים "שפת תכנות", "שפת מכונה", "שפה בינארית", ו-interpreter.

תשובה:

שפת מכונה היא שפה בינארית (אוסף של פקודות הבנויות מרצפים של אפסים ואחדות) המובנת ע"י מחשב.
כאשר מתכנתת כותבת קוד, היא כותבת אותו בשפת תכנות, המובנת לבני אדם. קוד הנכתב בשפת תכנות מצריך תרגום לשפת מכונה, על מנת שהמחשב יוכל לבצע את הפקודות.
תרגום זה נעשה על ידי ה-interpreter שמפרש את שפת התכנות לשפת מכונה.

לכן מתכנתות אמיתיות כותבות ישירות בשפת מכונה, בדיוק כמו שהמחשב מבין!

שאלה 2 (10 נק')

נתון מספר בינארי כלשהו בן 2 סיביות (bits). נסמן מספר זה ב- a. מוסיפים 3 אפסים מימין ל- a ומוסיפים 1 משמאל ל- a. למה שווה המספר החדש? הקיפו את התשובה הנכונה. אין צורך להסביר.

1. 8a+31
2. a+32
3. 8a+32
4. 31a+8
5. אף אחת מהתשובות הקודמות אינה נכונה

הסבר: עבור המספר a, המיוצג בבינארית על ידי 2 סיביות XX, כל הוספה של אפס מימין מכפילה את המספר פי 2. לכן, הוספה של 3 אפסים (כלומר, XX000) מכפילה את המספר פי 2*2*2=8. עד כה קיבלנו שהמספר החדש הוא 8*a.

בנוסף, כאשר מוסיפים 1 משמאל, המספר שמתקבל הוא 1XX000, ולכן ה-1 שהוספנו הוא בביט ה-5. כלומר, הוספנו $1*2^5=32$ למספר 8*a, וקיבלנו 8*a+32.

שאלה 3 (15 נק')

להלן פונקציה בשפת Python:

```
def calc(a, b):
    res = 0
    while b>0:
        res+=a
        b-=1
    return res
```

א. מה תחזיר הפונקציה עבור הקלט a=1, b=0? 0

ב. מה תחזיר הפונקציה עבור הקלט a=1, b=5? 5

ג. סמנו את התשובה הנכונה (אין צורך להסביר). עבור a ו- b שלמים חיוביים או אפס:

1. הפונקציה מחשבת את a*b
2. הפונקציה מחשבת את b**a
3. הפונקציה מחשבת את a+b-1
4. הפונקציה מחשבת את b-a

הסבר: הלולאה סוכמת b פעמים את הערך a לתוך המשתנה res.

שאלה 4 (20 נק')

בהינתן רשימה של מספרים, נגדיר "**פסגה**" כמספר שגדול (ממש) גם משכנו השמאלי וגם משכנו הימני. אם לא קיים אחד מהשכנים הללו (בקצוות הרשימה), איבר ייקרא פסגה אם הוא גדול (ממש) מהשכן שקיים לו (שמאלי או ימני, בהתאם לקצה).

בשאלה זו עליכם להשלים הפונקציה peaks הבאה, המקבלת רשימה lst, ומחזירה את כמות הפסגות שיש בה. להלן דוגמאות הרצה:

```
>>> peaks([1,2,1,2,1])
2
>>> peaks([1,1,1,2,1])
1
>>> peaks([1,2,3,4,5])
1
>>> peaks([3,2,3,4,5])
2
>>> peaks([3,5,3,4,5])
2
```

השלימו את הפונקציה:

```
def peaks(lst):
        cnt_peaks = 0
        # check first item in the list
        if lst[0] > lst[1]:
                cnt_peaks += 1
        # check last item in the list
        if lst[len(lst)-1] > lst[len(lst)-2]:
                cnt_peaks += 1

        # check the rest of the items in the list
        i = 1
        while i < len(lst)-1:
                if lst[i] > lst[i-1] and lst[i] > lst[i+1]:
                        cnt_peaks += 1
                i += 1

      return cnt_peaks
```

שאלה 5 (20 נק')

כזכור, חיפוש בינארי מניח שהרשימה בה מתבצע החיפוש ממוינת. לנוחיותכם מופיע בהמשך הקוד לחיפוש בינארי.

נניח שמבצעים חיפוש בינארי על רשימה שאיננה ממוינת. נקרא למספר שמחפשים ברשימה x.

לכל אחד מהמצבים הבאים: סמנו בעיגול האם הוא אפשרי או לא. אם לדעתכם הוא אפשרי, תנו דוגמה לרשימה לא ממוינת עם 5 מספרים, ומספר נוסף x שכאשר יינתנו לפונקציה לחיפוש בינארי יתרחש המצב המתואר (אין צורך להסביר במקרה זה). אם לדעתכם המצב איננו אפשרי, הסבירו מדוע.

מצב 1: הפונקציה מחזירה False עבור x שנמצא ברשימה.

אפשרי / לא אפשרי

עבור הרשימה [5,4,3,2,1] והקלט x=5, לאחר ההשוואה הראשונה לאיבר האמצעי ברשימה, הפונקציה תמשיך עם חצי הרשימה הימנית שלא מכילה את המספר x=5 ולכן תחזיר False.

מצב 2: הפונקציה מחזירה True עבור x שלא נמצא ברשימה.

אפשרי / לא אפשרי

מצב זה בלתי אפשרי שכן הפונקציה מחזירה True רק כאשר מתקיים **my_list[mid]==x**.
כיוון שלפי ההנחה, x לא נמצא ברשימה, התנאי הנ"ל לעולם לא יתקיים.

מצב 3: הפונקציה מחזירה False עבור x שלא נמצא ברשימה.

אפשרי / לא אפשרי

עבור הרשימה [5,4,3,2,1] והקלט x=6, לאחר ההשוואה הראשונה לאיבר האמצעי ברשימה, הפונקציה תמשיך עם חצי הרשימה הימנית שלא מכילה את המספר x=6 ולכן תחזיר False.

```
def binary_search(my_list, x):
    ''' search for x in my_list, which MUST BE SORTED !! '''
    left=0
    right=len(my_list)-1

    while left<=right:
        mid = (left+right)//2
        if my_list[mid]==x:
            return True
        elif my_list[mid] < x: #go to right half
            left = mid+1
        else:                  #go to left half
            right = mid-1
    return False  #if we got here the search failed
```

שאלה 6 (10 נק')

להלן בעיית הכרעה: **בהינתן מחרוזת, האם היא פלינדרום?**

תזכורת: פלינדרום היא מחרוזת שנקראת באותו אופן משמאל ומימין. למשל: "ABBA".

להלן האלגוריתם שראינו בכיתה לבדיקה האם מחרוזת נתונה היא פלינדרום:

```
def is_palindrome(st):
    n = len(st)
    for i in range(n//2): #check if n//2 is enough
        if st[i] != st[n-1-i]:
            return False #not a palindrome
    return True #if we for here, st IS a palindrome
```

ליד כל טענה סמנו בעיגול האם הטענה נכונה או לא נכונה, והסבירו בקצרה. אנו מניחים כמובן כי P≠NP.

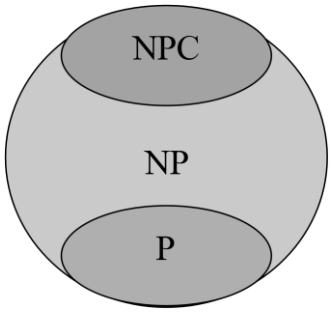
א. הבעיה שייכת ל- P (מחלקת הבעיות שיש להן פתרון בזמן פולינומי). **נכון / לא נכון**

נימוק:
הפונקציה עוברת על כל תו במחרוזת בדיוק פעם אחת. כלומר, זמן הריצה של הפונקציה הוא לינארי (כלומר פולינומי).

ב. הבעיה שייכת ל- NP (מחלקת הבעיות שיש להן אלגוריתם אימות לפתרון חוקי נתון בזמן פולינומי). **נכון / לא נכון**

נימוק:
הראנו בסעיף א' שהבעיה שייכת ל-P, וכיוון ש-P היא מחלקה המוכלת בתוך NP, נקבל שהבעיה שייכת ל-NP.

לנוחיותכם, האיור הבא שנלמד והוסבר בכיתה:



שאלה 7 (15 נק')

להלן הפונקציה Caesar_decrypt_try_all_auto שראינו בכיתה, לפענוח צופן קיסר:

```
1    def Caesar_decrypt_try_all_auto(encryption, common, th):
2        ''' find offsets in which encryption has >=th words from dictionary '''
3        for offset in range(1,128):
4            match = 0
5            possible_text = Caesar_decrypt(encryption, offset)
6            for w in common:
7                match += possible_text.count(w)
8            if match >= th:
9                print(offset, possible_text)
```

מחליפים את שורה 7 בשתי השורות הבאות (כמו קודם, בתוך לולאת ה- for):

```
7.1            if w in possible_text
7.2                match += 1
```

הסבירו מה שונה בפונקציה לעומת הגרסה המקורית שלה לפני השינוי, וציינו יתרון אחד וחיסרון אחד בשינוי זה.

תשובה:
ההבדל הוא שבגרסה החדשה לא סופרים את כמות ההופעות של מילה מתוך common בתוך התרגום הנבדק, אלא רק בודקים האם כל מילה מתוך common מופיעה או לא בתרגום.

יתרון לשינוי – אם למשל הטקסט שהוצפן הוא באנגלית, ובתוך common מופיעה המילה "is". זו מילה שכיחה בשפה ולכן עלולה להגדיל את match שלא לצורך. כלומר, יתכן שהפענוח היה נכון ביחס לשתי האותיות i,s אבל טעה לגבי אותיות אחרות, אבל זה הספיק כדי ש-match תעבור את הרף th הנתון.

חסרון לשינוי – עבור מילה שמופיעה בתדירות נמוכה בטקסט, היינו רוצים לתת לה משקל גבוה יותר, אם היא פוענחה נכון.
אם נשתמש בגרסה החדשה, מילה שמופיעה בתדירות נמוכה תקבל משקל זהה למילה כמו "is", למרות שזו אינדיקציה טובה יותר לכך שהפענוח נעשה נכון.

סוף!