מכללת אורט כפר-סבא

<u>מבני נתונים ויעילות אלגוריתמים</u>

<u>תרגיל מס' 6</u>

פתרו את השאלות הבאות. יש לסיים את התרגיל עד יום ד' (5.11).

<u>שאלה 1</u>

בשיעור הכרנו את הזהות: $\binom{n}{k}=\binom{n-1}{k}+\binom{n-1}{k-1}$

השתמשנו בזהות זו כדי לכתוב פונקציה רקורסיבית בשם choose, המקבלת את שני השלמים n ו-k ומחזירה את C(n,k). כעת, נשים לב כי מתקיים:

$$\binom{n}{k}=\frac{n!}{k!\cdot(n-k)!}=\frac{n\cdot(n-1)!}{k\cdot(k-1)!\cdot(n-k)!}=\frac{n}{k}\cdot\frac{(n-1)!}{(k-1)!\cdot(n-k)!}=\frac{n}{k}\cdot\binom{n-1}{k-1}$$

א. נמקו מדוע שרשרת שוויונות זו היא נכונה (כלומר: מדוע הביטוי השמאלי ביותר אכן שווה לזה שלימינו, ומדוע ביטוי זה אכן שווה לזה שימינו, וכך הלאה עד לשוויון הימני ביותר).

ב. מסעיף א' נובע כי מתקיימת הזהות: $\binom{n}{k}=\frac{n}{k}\cdot\binom{n-1}{k-1}$

כתבו גרסה רקורסיבית נוספת של הפונקציה choose, המתבססת על זהות זו.

ג. (<u>**רשות**</u>) הוכיחו כי הפונקציה הרקורסיבית choose שכתבתם בסעיף ב' היא יעילה יותר מהפונקציה הרקורסיבית choose שכתבנו בשיעור. האם היא גם יעילה יותר מהגירסה הלא-רקורסיבית של choose שכתבנו בשיעור?

<u>שאלה 2</u>

בשיעור הכרנו את משולש פסקל (Pascal's Triangle).

א. כתבו תכנית המקבלת כקלט מס' טבעי n, ומציגה כפלט את n השורות הראשונות של משולש פסקל. מותר להשתמש בפונקציה choose (באיזו גרסה שתרצו).

לדוגמא, עבור הקלט n = 5, יוצג הפלט:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

ב. **(<u>רשות</u>)** כתבו תכנית דומה לזו שכתבתם בסעיף א', פרט לכך שהפלט יעוצב באופן הבא (עבור הקלט $n = 5$):

```
        1
      1   1
    1   2   1
  1   3   3   1
1   4   6   4   1
```

<u>שאלה 3</u>

א. ממשו פונקציה שכותרתה:

```
void merge (int a[], int b[], int c[], int size)
```

הפונקציה מקבלת שני מערכים `a[]` ו-`b[]` בגודל `size` המכילים מספרים שלמים והממוינים בסדר עולה. היא תמזג אותם לתוך המערך `c[]` שגודלו `size*2`.

ב. ממשו פונקציה שכותרתה:

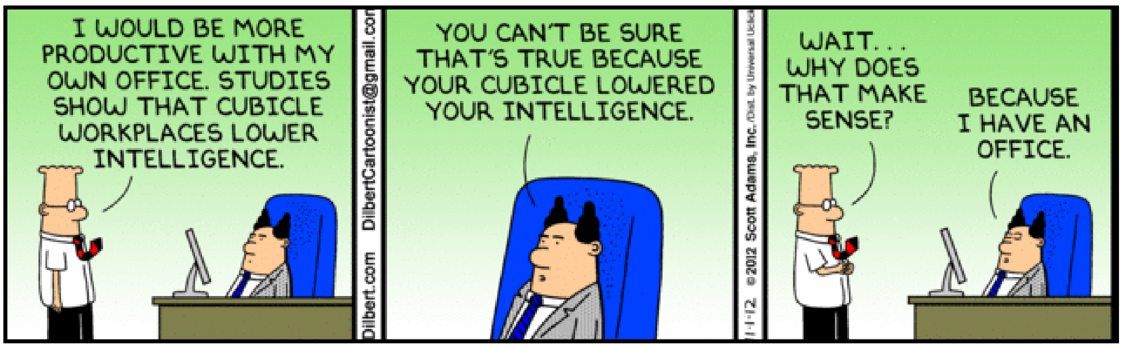
```
void merge (int a[], int low, int middle, int high)
```

הפונקציה מקבלת מערך `a[]` של מספרים שלמים ושלושה אינדקסים שלמים המקיימים $0 \le low \le middle \le high$. בנוסף, ידוע כי תת-המערכים `a[low...middle]` ו-`a[middle+1...high]` ממוינים בסדר עולה. הפונקציה תמיין בסדר עולה את המערך `a[]` בתחום `a[low...high]`. כתבו את הפונקציה כך שתעשה שימוש במערך עזר שיוקצה דינאמית (על-ידי malloc).

ג. ממשו פונקציה שכותרתה:

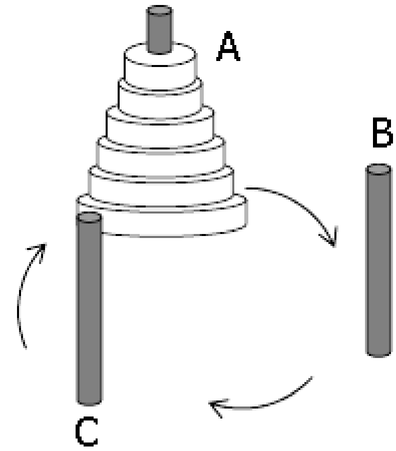
```
void merge_sort (int a[], int low, int high)
```

הפונקציה מקבלת מערך `a[]` של שלמים, ואינדקסים המקיימים: $0 \le low \le high$. היא תמיין **באופן רקורסיבי** את המערך `a[low...high]`, באמצעות האלגוריתם למיון מיזוג (Merge Sort) שלמדנו בכיתה. היעזרו בפונקציה `merge` מסעיף ב'.



שאלה 4

בשאלה זו עליכם לכתוב פונקציה המדפיסה הוראות לפתרון בעיית מגדלי האנוי (בדומה לפונקציה hanoi שכתבנו בשיעור ולפונקציה constrained_hanoi שכתבתם בתרגיל הקודם), אך בהינתן אילוץ נוסף: המגדלים מסודרים במעגל, ובכל צעד, מותר להזיז דיסקה רק בין שני מגדלים סמוכים **עם כיוון השעון בלבד**. כלומר: ממגדל A ניתן להזיז דיסקה רק למגדל B, וממגדל B ניתן להזיז דיסקה רק למגדל C, וממגדל C ניתן להזיז דיסקה רק למגדל A.



לפניכם הגדרת טיפוס:

```
typedef enum {A,B,C} tower;
```

נתונה הפונקציה cyclic_hanoi, הפותרת את הבעיה. בפונקציה חסרים שבעה ביטויים, הממוספרים (1)-(7), ויש להשלים אותם.

```
void cyclic_hanoi (int n, tower from, tower to)
{
 tower next = (from == C) ? A : from + 1;

 if (n == 0)
    return;

 if (to == next) {                    /* מגדל היעד עוקב למגדל המקור */
      ____(1)____;
      ____(2)____;
      ____(3)____;

 } else {               /*  המגדל next מפריד בין מגדל היעד למגדל המקור */
      ____(4)____;
      ____(5)____;
      ____(6)____;
      printf ("move disc from %c to tower %c\n", 'A'+next, 'A'+to);
      ____(7)____;
 }
}
```

<u>שאלה 5</u>

נניח כי המערך A מכיל n מספרים שלמים **השונים כולם זה מזה**, וכי n הוא מספר **אי-זוגי**. נגדיר את החציון (median) של קבוצת איברי המערך A בתור אותו איבר מתוך A הגדול בדיוק מחצי מהאיברים, והקטן בדיוק מחצי מהאיברים.

כלומר, במערך באורך n, החציון הוא איבר הגדול מ- $\frac{n-1}{2}$ איברים והקטן מ- $\frac{n-1}{2}$ איברים.

לדוגמא: אם n = 5 והמערך A הוא (4, -1, 6, 13, 1) אז החציון הוא המספר 4 (שגדול בדיוק משני איברים, 1- ו-1, וקטן בדיוק משני איברים, 6 ו-13).

מעוניינים לפתח אלגוריתם המקבל כקלט מערך כנ״ל ואת גודלו, ומחזיר את ערכו של החציון. שני סטודנטים, מתושלח וירחמיאל, הציעו שני אלגוריתמים שונים לפתרון הבעיה.

א. הסטודנט מתושלח הציע את האלגוריתם הבא: נעבור בלולאה על כל איברי המערך, ועבור כל אחד מהם – נעבור בלולאה פנימית על המערך ונמנה כמה איברים במערך קטנים ממנו. אם מספרם שווה ל- $\frac{n-1}{2}$ אז נחזיר אותו ונסיים. אחרת – נמשיך הלאה.
כתבו פונקציה בשפת C המממשת אלגוריתם זה.

ב. הסטודנט ירחמיאל הציע את האלגוריתם הבא: נמיין את איברי המערך בשיטת מיון מיזוג, ואז נחזיר את האיבר במערך שהאינדקס שלו הוא $\frac{n-1}{2}$.
כתבו פונקציה בשפת C המממשת אלגוריתם זה.

ג. מהי סיבוכיות זמן הריצה של כל אחד מהאלגוריתמים? האם האלגוריתם של ירחמיאל מהווה שיפור בסדר גודל (order of magnitude improvement) לעומת האלגוריתם של מתושלח, או אולי רק שיפור בקבוע (improvement by factor)? הסבירו.