



## מבוא למדעי המחשב מ"ח' (234114 \ 234117)

### סמסטר אביב תשע"ח

### מבחן מסכם מועד א', 8 ביולי 2018

2	3	4	1	1	
---	---	---	---	---	--

רשום/ה לקורס:

--	--	--	--	--	--	--	--	--	--

מספר סטודנט:

#### משך המבחן: 3 שעות.

חומר עזר: אין להשתמש בכל חומר עזר.

#### הנחיות כלליות:

- מלאו את הפרטים בראש דף זה ובדף השער המצורף, בעט בלבד.
- בדקו שיש 24 עמודים (4 שאלות) במבחן, כולל עמוד זה.
- כתבו את התשובות על טופס המבחן בלבד, במקומות המיועדים לכך. שימו לב שהמקום המיועד לתשובה אינו מעיד בהכרח על אורך התשובה הנכונה.
- העמודים הזוגיים בבחינה ריקים. ניתן להשתמש בהם כדפי טיוטה וכן לכתיבת תשובותיכם. סמנו טיוטות באופן ברור על מנת שהן לא תבדקנה.
- יש לכתוב באופן ברור, נקי ומסודר. ניתן בהחלט להשתמש בעיפרון ומחק, פרט לדף השער אותו יש למלא בעט.
- בכל השאלות, הינכם רשאים להגדיר ולממש פונקציות עזר כרצונכם. לנוחיותכם, אין חשיבות לסדר מימוש הפונקציות בשאלה, ובפרט ניתן לממש פונקציה לאחר השימוש בה.
- אלא אם כן נאמר אחרת בשאלות, **אין להשתמש בפונקציות ספריה או בפונקציות שמומשו בכיתה**, למעט פונקציות קלט/פלט והקצאת זיכרון (`malloc`, `free`). ניתן להשתמש בטיפוס `bool` המוגדר ב-`stdbool.h`.
- אין להשתמש במשתנים סטטיים וגלובאליים אלא אם נדרשתם לכך מפורשות.
- כשאתם נדרשים לכתוב קוד באילוצי סיבוכיות זמן/מקום נתונים, אם לא תעמדו באילוצים אלה תוכלו לקבל בחזרה מקצת הנקודות אם תחשבו נכון ותציינו את הסיבוכיות שהצלחתם להשיג.
- נוהל "לא יודע": אם תכתבו בצורה ברורה "לא יודע/ת" על שאלה (או סעיף) שבה אתם נדרשים לקודד, תקבלו 20% מהניקוד. דבר זה מומלץ אם אתם יודעים שאתם לא יודעים את התשובה.
- נוסחאות שימושיות:

$$1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} = \Theta(\log n) \quad 1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \frac{1}{25} + \dots = \Theta(1)$$

$$1 + 2 + \dots + n = \Theta(n^2) \quad 1 + 4 + 9 + \dots + n^2 = \Theta(n^3) \quad 1 + 8 + 27 + \dots + n^3 = \Theta(n^4)$$

צוות הקורס 234114/7

**מרצים:** פרופ' מירלה בן-חן (מרצה אחראית), גב' יעל ארז **מתרגלים:** איתי הנדלר, נג'ב נבואני, עמר צברי, דמיטרי רבינוביץ' (מתרגל אחראי), יאיר ריעאני.

**בהצלחה!**





## שאלה 1 (25 נקודות):

א. (8 נקודות) חשבו את סיבוכיות הזמן והמקום של הפונקציה  $f1$  המוגדרת בקטע הקוד הבא, כפונקציה של  $n$ . אין צורך לפרט שיקולים. חובה לפשט את הביטוי ככל שניתן. הניחו שסיבוכיות הזמן של  $\text{malloc}(n)$  היא  $\Theta(1)$ , וסיבוכיות המקום של  $\text{malloc}(n)$  היא  $\Theta(n)$ .

```
void f1(int n)
{
    int k = 0, result = 1;
    while (n > 0){
        k += n;
        n--;
    }
    void* ptr = malloc(k);
    while (k > 0){
        result *= k;
        k /= 2;
    }
    free(ptr);
    return result;
}
```

סיבוכיות זמן:  $\theta(\quad)$       סיבוכיות מקום:  $\theta(\quad)$

ב. (9 נקודות) חשבו את סיבוכיות הזמן והמקום של הפונקציה  $f2$  המוגדרת בקטע הקוד הבא, כפונקציה של  $n$ . אין צורך לפרט שיקולים. חובה לפשט את הביטוי ככל שניתן.

```
int aux(int n)
{
    if(n < 0) return 1;
    return aux(n-1)+aux(n-1)+aux(n-1);
}

int f2(int n)
{
    return aux(n/3);
}
```

סיבוכיות זמן:  $\theta(\quad)$       סיבוכיות מקום:  $\theta(\quad)$

[illegible]



ג. (8 נקודות): חשבו את סיבוכיות הזמן והמקום של הפונקציה  $f3$  המוגדרת בקטע הקוד הבא, כפונקציה של  $n$ . אין צורך לפרט שיקולים. חובה לפשט את הביטוי ככל שניתן.

```
int aux(int n, int x)
{
    if(x * x * x > n) return 1;
    return aux(n, x + 1);
}

void f3(int n)
{
    aux(n, 0);
    aux(n, n / 3);
    aux(n, 2 * n / 3);
    aux(n, n);
}
```

סיבוכיות זמן:  $\theta(\text{_____})$  סיבוכיות מקום:  $\theta(\text{_____})$

[illegible]



## שאלה 2 (25 נק')

בשאלה זאת נניח כי משתנה מסוג `int` יכול לייצג מספרים גדולים מאוד כרצוננו (למשל 256 סיביות או יותר).

נתון לנו מימוש יעיל במיוחד של הפונקציה הבאה (אין צורך לממש אותה!):

```
bool is_high_factor_in_range(int n, int low, int high)
```

פונקציה זאת מקבלת מספר טבעי  $n$  ו-2 מספרים  $low$  ו- $high$ , אשר מייצגים שני קצוות של טווח החיפוש. בשאלה זו אנו נניח, כי  $n$  הינו מכפלה של שני ראשוניים:  $p$  ו- $q$ . הפונקציה תבדוק האם הגורם הראשוני הגדול מבין  $p, q$  נמצא בטווח החיפוש, ואם כן תחזיר 1. אחרת היא תחזיר 0. (טווח החיפוש כולל את הקצוות).

לדוגמה, עבור  $n = 1,000,730,021$  וטווח בין 20,000 עד 40,000 הפונקציה תחזיר 0. בעוד עבור הטווח בין 10 ל-101,000 היא תחזיר 1. (הגורם הראשוני הוא 100,003).

סיבוכיות הזמן של הפונקציה `is_high_factor_in_range` היא  $\Theta(\log n)$ .

עליכם לממש את הפונקציה `find_factors` בצורה יעילה ככל הניתן. הפונקציה תקבל מספר גדול  $n$  שהוא מכפלה של 2 מספרים ראשוניים גדולים ( $n = p \cdot q$ ) ותמצא ותחזיר את  $p$  ו- $q$ .

לדוגמה, עבור  $n = 8509$  הפונקציה תחזיר  $p = 67$  ו- $q = 127$ . מאחר  $67 \cdot 127 = 8509$ .

אנא ציינו כאן את הסיבוכיות שהגעתם אליה:

זמן \_\_\_\_\_ מקום נוסף \_\_\_\_\_

```
void find_factors(int n, int *p, int *q)
```

```
{
```

[illegible]





[illegible]



### שאלה 3 (25 נקודות) :

א. (10 נק')

הגדרה: בהינתן מחרוזת str הבנויה מתווים שערכי ה-ASCII שלהם נעים בין 1 ל-127 נאמר שתו a "מציק" אם מספר הפעמים שהוא מופיע ב-str הוא מקסימאלי (דהיינו, לא קיים תו אחר במחרוזת str שמופיע בה מספר רב יותר של פעמים).

שימו לב שייתכנו כמה תווים מציקים עבור מחרוזת מסוימת.

דוגמאות:

במחרוזת "hakuna matata" התו 'a' מציק.

במחרוזת "nanana\$\$" התווים 'a' ו-'n' מציקים.

השלימו את הפונקציה annoying שמקבלת מחרוזת ומחזירה תו מציק כלשהו. עבור המחרוזת "" (מחרוזת ריקה) יש להחזיר '\0', אך בשאר המקרים אין להתחשב בתו '\0'.

דרישות:

- סיבוכיות זמן:  $\Theta(n)$  כאשר n הוא אורך המחרוזת, וסיבוכיות מקום נוסף:  $\Theta(1)$ .

אם לפי חישוביכם לא עמדתם בדרישות הסיבוכיות אנא ציינו כאן את הסיבוכיות שהגעתם אליה:  
זמן \_\_\_\_\_ מקום נוסף \_\_\_\_\_

```
char annoying (char* str) {
```

[illegible]

[illegible]

[illegible]



ב. (15 נק')

השלימו את הפונקציה unAnnoy אשר מקבלת מערך של מחרוזות, ואת גודלו, ומוחקת מכל מחרוזת את כל המופעים של התו המציק בה. לדוגמה עבור מערך המחרוזות הבא:

```
{"hakuna matata", "nnnanana$$", ""}
```

נקבל לאחר הפעלת הפונקציה את המערך הבא:

```
{"hkun mtt", "aaa$$", ""}
```

הערות:

- ניתן להניח שבכל מחרוזת במערך יש רק תו מציק אחד.
- ניתן להשתמש בפונקציה מסעיף א' גם אם לא מימשתם אותה.
- מחרוזת ריקה אינה עוברת שום שינוי.

דרישות:

- סיבוכיות זמן:  $\Theta(mn)$  כאשר  $m$  הינו האורך של המחרוזת הארוכה ביותר ו-  $n$  הינו גודלו של מערך המחרוזות. סיבוכיות מקום נוסף:  $\Theta(1)$ .

אם לפי חישוביכם לא עמדתם בדרישות הסיבוכיות אנא ציינו כאן את הסיבוכיות שהגעתם אליה:  
זמן \_\_\_\_\_ מקום נוסף \_\_\_\_\_

```
void unAnnoy (char *arr[], int n){
```

[illegible]





[illegible]



#### שאלה 4 (25 נקודות) :

נתונות קוביות של אותיות. על כל פאה של קובייה מצוירת אות **אחת**, אך בכל קובייה יתכן מספר **שונה** של פאות.

לדוגמה נשתמש ב-4 קוביות:

לקובייה הראשונה 3 פאות, ובהן האותיות **A, B, C**

לקובייה השנייה 3 פאות, ובהן האותיות **A, E, I**

לקובייה השלישית 3 פאות, ובהן האותיות **E, O, U**

לקובייה הרביעית 5 פאות, ובהן האותיות **L, N, R, S, T**

כתבו פונקציה שמקבלת מערך של מחרוזות שמייצגות קוביות, את מספר הקוביות  $n$ , ומילה. הפונקציה מחזירה **true** אם ניתן להרכיב את המילה מהקוביות הנתונות ו **false** אחרת.

למשל, מהקוביות שבדוגמא ניתן להרכיב את המילים: **CAT, BAT, BONE**.

לעומת זאת, מהקוביות שבדוגמא לא ניתן להרכיב את המילה **BAIT** כיוון שאם נשתמש בקובייה השנייה לאות **A** לא נוכל להשתמש בה שוב לאות **I**.

#### הערות:

- יש להשתמש בשיטת **backtracking** כפי שנלמדה בכיתה.
- בשאלה זו אין דרישות סיבוכיות, אולם כמקובל ב-**backtracking** יש לוודא שלא מתבצעות קריאות רקורסיביות מיותרות עם פתרונות שאינם חוקיים.
- ניתן להניח שבכל קובייה יש לפחות אות אחת ובמילה **word** יש לפחות אות אחת.
- ניתן ומומלץ להשתמש בפונקציות עזר (ויש לממש את כולן).

<code>bool Scrambleable(char *cubes[], int n, char* word)</code>
<code>{</code>



[illegible]

[illegible]

[illegible]

[illegible]