



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

### סמסטר אביב תש"פ

### מבחן מסכם מועד א', 28 ביולי 2020

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 22 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 תיבדקנה.
- יש לכתוב באופן ברור, נקי ומסודר, ובעט בלבד.
- בכל השאלות, הנכם 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:

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

$$1^k + 2^k + 3^k + \dots + n^k = \Theta(n^{k+1}) \quad 1 + k + k^2 + k^3 + \dots + k^n = \Theta(k^n), \quad k > 1$$

צוות הקורס 234114/7

**מרצים:** פרופ' ניר אילון (מרצה אחראי), גב' יעל ארז **מתרגלים:** עמית ברכה, דן ברקוביץ', גיא ברשצקי, עמר דהרי, קטרין חדאד, דמיטרי רבינוביץ' (מתרגל אחראי)

**בהצלחה!**

[illegible]



## 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.

```
int g(int n)
{
    if(n <= 1) return 1;
    return g(n / 2) + 1;
}

void f1(int n)
{
    for(int i = 1; g(i) < n; i *= 2)
    {
        printf(" * **\n");
    }
}
```

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 g2(int n, int m)
{
    if (m >= n)
    {
        for (int i = 0; i < n; ++i)
            printf("#");
        return 1;
    }

    return 1 + g2(n / 2, 4 * m);
}

int f2(int n)
{
    return g2(n, 1);
}
```

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.

```
void f1(int n)
{
    int x = 0;
    for (int i = 0; i <= n / 2; i += 3)
        for (int j = i; j <= n / 4; j += 2)
            x++;
}
```

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





## שאלה 2 (25 נק')

נתון מערך ממוין  $a$  שמכיל מספרים שלמים. המספרים שבמערך שונים זה מזה. אינדקס  $i$  ייקרא נקודת שבת של המערך אם  $a[i]=i$ .

ממשו את הפונקציה `FindFixedPoint` המקבלת מערך  $a$  של מספרים שלמים ואת אורכו  $n$  ומחזירה את אחת מנקודות השבת של המערך (אם קיימת):

```
int FindFixedPoint (int a[], int n)
```

לדוגמה: אם נתון המערך

-3	0	1	3	4	6	13	42
----	---	---	---	---	---	----	----

אז הפונקציה תחזיר 3. תשובה אפשרית אחרת היא 4.

## שימו לב:

- המערך מכיל מספרים שלמים בלבד.
- המערך ממוין (בסדר עולה).
- המספרים במערך שונים זה מזה.

## הערות:

- אם אין נקודות שבת במערך הפונקציה תחזיר -1.

## דרישות:

- ממשו את הפונקציה בסיבוכיות זמן  $O(\log n)$ . אין הגבלה על סיבוכיות המקום.

אם לא עמדתם בדרישות הסיבוכיות, אנא ציינו כאן את הסיבוכיות שהגעתם אליה:

זמן \_\_\_\_\_ מקום נוסף \_\_\_\_\_

```
int FindFixedPoint (int a[], int n)
```

```
{
```

[illegible]



[illegible]

[illegible]

**שאלה 3 (25 נקודות) :**

ממשו את הפונקציה:

```
void reverse_words(char *str)
```

הפונקציה תקבל מחרוזת `str` המכילה מילים המופרדות ברווחים ותשנה את המחרוזת `str` כך שסדר המילים יתהפך.

**דוגמא:**

- עבור המחרוזת `str[] = "geeks quiz practice code"` הפונקציה תשנה את `str` ל- `"code practice quiz geeks"`

- עבור המחרוזת

```
str[] = " getting good at coding needs a lot of practice"
```

הפונקציה תשנה את המחרוזת ל-

```
"practice of lot a needs coding at good getting "
```

**הערות:**

- ניתן להניח כי המילים מורכבות מאותיות אנגליות קטנות בלבד.
- אם מספר רווחים מפריד בין שתי מילים עוקבות במחרוזת, אז, כתוצאה מהפעלת הפונקציה, אותן מילים תופרדנה ע"י אותו מספר של רווחים.
- רווחים שמופיעים לפני המילה הראשונה בקלט יופיעו אחרי המילה האחרונה בפלט. בדומה, רווחים שמופיעים אחרי המילה האחרונה בקלט יופיעו לפני המילה הראשונה בפלט (ראו דוגמה).

**דרישות:**

- סיבוכיות זמן  $O(n)$ , כאשר  $n$  הינו אורך של המחרוזת `str`, סיבוכיות מקום  $O(1)$ .

אם לפי חישוביכם לא עמדתם בדרישות הסיבוכיות אנא ציינו כאן את הסיבוכיות שהגעתם אליה:

זמן \_\_\_\_\_ מקום נוסף \_\_\_\_\_

```
void reverse_words(char *str)
```

```
{
```

[illegible]

[illegible]

[illegible]



## שאלה 4 (25 נקודות) :

נתון מערך `arr` באורך `n` לא ממוין המכיל מספרים שלמים. ממשו פונקציה

```
bool IsKSplittable(int arr[], int n, int k)
```

שמקבלת את המערך `arr`, אורכו `n` ומספר שלם `k` ומחזירה האם קיימת חלוקה של קבוצת המספרים במערך ל-`k` תת-קבוצות. כך שסכום כל המספרים בכל תת-קבוצה זהה. וכך שכל איבר במערך שייך לתת-קבוצה אחת ויחידה.

לדוגמה: אם נתון מערך הבא:

9	-2	3	17	5	3	8	4	1	1	2
---	----	---	----	---	---	---	---	---	---	---

אז `IsKSplittable(arr, 11, 3)` תחזיר `true`, כי חלוקה אפשרית הבאה `{17}, {9, 4, 1, 1, 2}` ו-`{-2, 3, 5, 3, 8}` עונה על כל הדרישות.

בעוד `IsKSplittable(arr, 11, 4)` תחזיר `false`, מאחר ולא ניתן לחלק את המספרים לארבע קבוצות העונות על כל הדרישות.

## 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 ומומלץ להשתמש בפונקציות עזר (ויש לממש את כולן).

```
bool IsKSplittable(int arr[], int n, int k)
```

```
{
```

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]