

מבוא למדעי המחשב מי/חי

# 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:
- 1							1					

## משך המבחן: 3 שעות.

חומר עזר: אין להשתמש בכל חומר עזר.

### הנחיות כלליות:

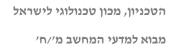
- מלאו את הפרטים בראש דף זה ובדף השער המצורף, בעט בלבד.
  - בדקו שיש 22 עמודים (4 שאלות) במבחן, כולל עמוד זה.
- כתבו את התשובות על טופס המבחן בלבד, במקומות המיועדים לכך. שימו לב שהמקום המיועד לתשובה אינו מעיד בהכרח על אורך התשובה הנכונה.
- העמודים הזוגיים בבחינה ריקים. ניתן להשתמש בהם כדפי טיוטה וכן 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
   stdbool.h.-a המוגדר ב-bool
  - 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:

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

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!







# :(שאלה 1 (25 נקודות)

א. (8) נקודות) חשבו את סיבוכיות הזמן והמקום של הפונקציה (1) המוגדרת בקטע הקוד הבא, מפונקציה של (n) אין צורך לפרט שיקוליכם. 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");
    }
}</pre>
```

 $\Theta(n)$  סיבוכיות מקום:  $\Theta(n^2)$  סיבוכיות מקום:

ב. (<u>9 נקודות)</u>: חשבו את סיבוכיות הזמן והמקום של הפונקציה f2 המוגדרת בקטע הקוד הבא, כפונקציה של n. אין צורך לפרט שיקוליכם. <u>חובה לפשט את הביטוי ככל שניתן.</u>

 $\Theta(\log n)$  סיבוכיות מקום:  $\Theta(n^{2/3})$  סיבוכיות מקום:





סמסטר אביב 2020



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

 $\Theta(1)$  סיבוכיות מקום:  $\Theta(n^2)$  סיבוכיות מקום:





İ	
İ	
İ	
İ	
İ	
İ	
İ	

הפקולטה למדעי המחשב



מבוא למדעי המחשב מי/חי

## (נקי) אלה 2 (25 נקי)

ייקרא i ספרים שמכיל מספרים שלמים. המספרים שבמערך שונים זה מזה. אינדקס i ייקרא מערך ממוין a [i] = i a

ממשו את הפונקציה FindFixedPoint המקבלת מערך a של מספרים שלמים ואת אורכו מ ומחזירה את אחת מנקודת השבת של המערך (אם קיימת):

int FindFixedPoint (int a[], int n)

לדוגמה: אם נתון המערך

$\begin{vmatrix} -3 & 0 & 1 & 3 & 4 & 6 & 13 \end{vmatrix}$	12
---	----

4 אז הפונקציה תחזיר 3 תשובה אפשרית אחרת היא

### שימו לב:

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

## :הערות

-1 אם אין נקודות שבת במערך הפונקציה תחזיר -1

### :דרישות

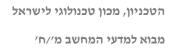
. אין הגבלה על סיבוכיות המקום ממשו את הפונקציה בסיבוכיות זמן $O(\log n)$ . אין	•	

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

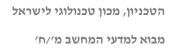
i	int	Fin	dFix	redPo	oint	(in	t a[	],	int	n)			
{													



```
int FindFixedPoint(int a[], int n)
{
      int under = 0, above = n - 1;
      while (under <= above)</pre>
      {
            int hit = (under + above) / 2;
            if (a[hit] == hit)
                  return hit;
            if (a[hit] < hit)</pre>
                  under = hit + 1;
            else
                  above = hit - 1;
      }
      return -1;
```













## שאלה 3 (25 נקודות):

ממשו את הפונקציה:

void reverse words(char \*str)

קב  $\operatorname{str}$  המכילה מילים המופרדות ברווחים ותשנה את המחרוזת  $\operatorname{str}$  כך  $\operatorname{str}$  שסדר המילים יתהפך.

#### :דוגמא

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

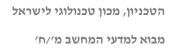
	.0(1) סיבוכיות זמן ( $n$ ), כאשר $n$ הינו אורך של המחרוזת str, סיבוכיות מקום ( $n$ ).	
ליה:	לפי חישוביכם לא עמדתם בדרישות הסיבוכיות אנא ציינו כאן את הסיבוכיות שהגעתם א	אם י
	מקום נוסף	זמן
id 1	reverse_words(char *str)	



```
void reverse(char* start, char* end)
{
      while (start < --end)</pre>
            swap(start++, end);
}
void reverse_words(char* str)
      reverse(str, str + strlen(str));
      char* start, * end = str;
      while (*end)
      {
            start = end;
            while (*start == SPACE)
                  ++start;
            end = start;
            while (*end != '\0' && *end != SPACE)
                  ++end;
            reverse(start, end);
      }
}
```













## : (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:

$\begin{array}{ c c c c c c c c c c c c c c c c c c c$	7 5 3 8	4 1 1 2
--	---------	---------

 $\{17\}$ , מחזיר באה (י חלוקה אפשרית הבאה (17 $\}$ , אז (17 $\}$  IsKSplittable (arr, 11, 3) אז (17 $\}$  ו- $\{9, 4, 1, 1, 2\}$ 

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

#### הערות:

- יש להשתמש בשיטת backtracking כפי שנלמדה בכיתה.
- ש לוודא שלא backtracking- 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).

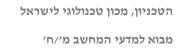
```
bool IsKSplittable(int arr[], int n, int k)
{
    int* sums = malloc(k * sizeof(int));
    for (int i = 0; i < k; ++i)
        sums[i] = 0;

    bool isPossible = IsKSplittable_aux(arr, n, sums, k, 0);
    free(sums);

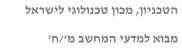
    return isPossible;
}</pre>
```



```
bool AreAllSumsEqual(int sums[], int k)
{
      for (int i = 1; i < k; ++i)</pre>
            if (sums[i] != sums[i - 1])
                  return false;
      return true;
}
bool IsKSplittable_aux(int arr[], int n, int sums[], int k, int current)
{
      if (current == n)
            return AreAllSumsEqual(sums, k);
      for (int where = 0; where < k; ++where)</pre>
      {
            sums[where] += arr[current];
            if (IsKSplittable_aux(arr, n, sums, k, current + 1))
                  return true;
            sums[where] -= arr[current];
      return false;
}
```









ĺ	
ĺ	
i	
ĺ	
ĺ	
ĺ	
ĺ	
ĺ	
ĺ	
ĺ	
l	
i	
i	
ĺ	
i	
ĺ	
ĺ	
ĺ	
ĺ	
i	
ĺ	

