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

סמסטר אביב תשע"ח

מבחן מסכם מועד ב', 7 באוקטובר 2018

מספר סטודנט: | | | | | | | | | | *רשום/ה לקורס:* | | | | | | 2 | 3 | 4 | 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 20 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, **ובעט בלבד**.
- 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:

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

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

צוות הקורס 234114/7

מרצים: פרופ' מירלה בן-חן (מרצה אחראית), גב' יעל ארז **מתרגלים:** איתי הנדלר, נג'יב 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 שיקוליכם. חובה לפשט את הביטוי ככל שניתן.
הניחו שסיבוכיות הזמן של הוצאת שורש ריבועי sqrt(n) היא $\Theta(1)$, וסיבוכיות המקום של sqrt(n) היא $\Theta(1)$.

```
void aux(int n)
{
      for(int i = n; i > 2; i = sqrt(i))
            printf("*");
}

void f1(int n)
{
      aux(n);
      aux(n * n);
      aux(n * n * n);
}
```

סיבוכיות זמן: $\theta(\log\log n)$ סיבוכיות מקום: $\theta(1)$

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

```
void aux(int from, int to, int n)
{
      if(n == 0) return;
      aux(from, 6 - from - to, n - 1);
      printf("Moved %d from %d to %d\n", n, from, to);
      aux(6 - from - to, to, n - 1);
}

void f2(int n)
{
      aux(1, 2, n);
}
```

סיבוכיות זמן: $\theta(2^n)$ סיבוכיות מקום: $\theta(n)$

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

```
int f3(int n)
{
      if(n <= 2) return 1;
      f3(1 + f3(n-2));
      return n - 1;
}
```

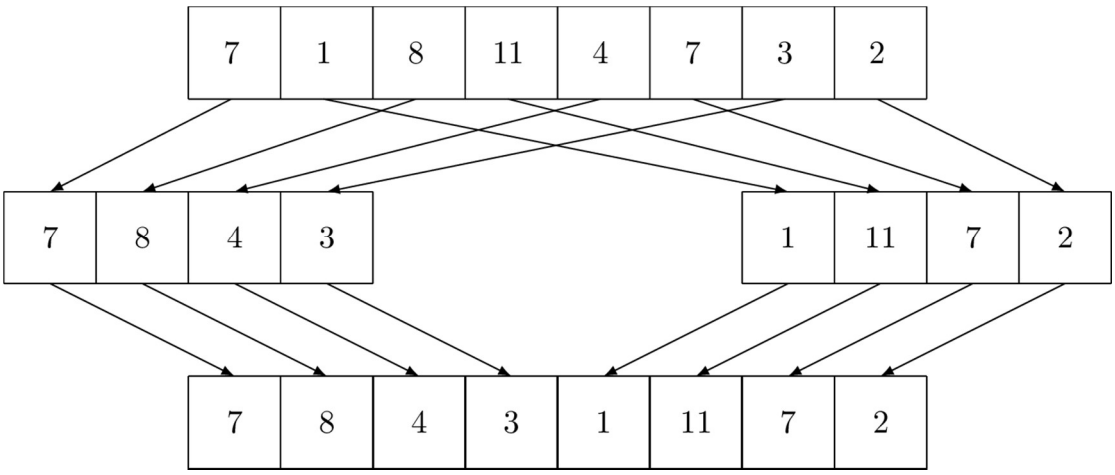
סיבוכיות זמן: $\theta(\ 2^{\frac{n}{2}}\ )$ סיבוכיות מקום: $\theta(\ n\ )$

שאלה 2 (25 נק')

עליכם לכתוב פונקציה המקבלת מערך ומשנה את סדר האברים בו כך שהאברים במקומות הזוגיים יעברו לתחילת המערך, והאברים במקומות האי-זוגיים יעברו לסוף המערך. הסדר היחסי של האברים הזוגיים והאי-זוגיים צריך להישמר.

בדוגמא שלמטה, האברים `{7,8,4,3}` נמצאים במקומות זוגיים ולכן הם מועברים לתחילת המערך, והאברים `{1,11,7,2}` נמצאים במקומות אי זוגיים ולכן הם מועברים לסוף המערך.

הניחו שאורך המערך גדול מ `1`, והוא חזקה של `2`.



דרישות:

- סיבוכיות זמן $O(n \log n)$ וסיבוכיות מקום $O(\log n)$.
- פתרון בסיבוכיות מקום $O(n)$ **אינו** מזכה בניקוד מעבר לנוהל **לא יודע/ת.**

זמן _____ מקום נוסף _____.

```
void stable_shuffle(int arr[], int n) {
        if (n <= 2) return;

        stable_shuffle(arr, n / 2);
        stable_shuffle(arr + n / 2, n / 2);
        for (int i = 0; i < n / 4; ++i)
                swap(arr + n / 4 + i, arr + n / 2 + i);
}
```

שאלה 3 (25 נקודות) :

השאלה עוסקת במציאת התו הראשון שמופיע בדיוק פעם אחת במחרוזת.

ממשו פונקציה שמקבלת מחרוזת `str` הבנויה מתווים שערכי ה- ASCII שלהם נעים בין 1 ל-127, ומחזירה את התו הראשון שאינו חוזר על עצמו במחרוזת, או את התו `'\0'` אם אין כזה.

שימו לב, כדי שהפונקציה תתאים לעבודה מול זיכרונות איטיים (כגון דיסק קשיח וכו'), עליכם להגביל את כמות הגישות לזיכרון לצורך הפתרון. לכן, **<u>עליכם לעבור על המחרוזת פעם אחת בלבד</u>**.

דוגמאות:

במחרוזת `"blazqnqbla"` התו `'z'` הוא הראשון שאינו חוזר על עצמו.

במחרוזת `"nanana$$"` כל התווים חוזרים על עצמם ולכן הפונקציה תחזיר `'\0'`.

דרישות:

- סיבוכיות זמן: $\theta(n)$, וסיבוכיות מקום נוסף: $\theta(1)$.

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

```
char find_first_unique(char *str)
{
#define ABC 128
#define NOT_SEEN -1
```

```
char find_first_unique(char *str)
{
	int occurences[ABC] = { 0 };
	int first[ABC];

	for (int letter = 0; letter < ABC; ++letter)
		first[letter] = NOT_SEEN;

	int at = 0;
	do
	{ // count letters and store first occurrence index
		if (occurences[*str]++ == 0)
			first[*str] = at;
		++at;
	} while (*str++);

	char first_letter = '\0';
	int first_letter_occured = first[first_letter];

	for (int letter = 0; letter < ABC; ++letter)
	{ // find first unique letter
		if (occurences[letter] == 1 && first[letter] < first_letter_occured)
		{
			first_letter_occured = first[letter];
			first_letter = letter;
		}
	}

	return first_letter;
}
```

שאלה 4 (25 נקודות) :

בהינתן מספר שלם חיובי n נתעניין בשאלה הבאה: האם ניתן לסדר 2 עותקים של המספרים מ-1 עד ל-n במערך שאורכו 2n, כך שמספר האיברים בין שני העותקים של המספר i, הוא בדיוק i.

לדוגמה, עבור n = 3, {3,1,2,1,3,2} הינו סידור אפשרי , מכיוון שמספר האיברים בין שני העותקים של 1 הוא 1, מספר האיברים בין שני העותקים של 2 הוא 2 ומספר האיברים בין שני העותקים של 3 הוא 3.

עבור n = 4, {4,1,3,1,2,4,3,2} הינו סידור אפשרי.

עבור n = 2, אין אף סידור אפשרי .

כתבו פונקציה שמקבלת מספר טבעי n ומערך באורך 2n ומחזירה ערך בוליאני המסמן האם ניתן למלא את המערך בהתאם לדרישות לעיל. במקרה זה, אחד מהסידורים האפשריים יימצא במערך המסופק לאחר החזרה מהפונקציה. במקרה ולא קיים אף סידור הפונקציה תחזיר false ותוכן המערך אינו משנה.

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).

```
#define UNSET 0
```

```
bool Proportionalable(int arr[], int n)
{
        for (int i = 0; i < 2 * n; ++i)
                arr[i] = UNSET;
        return Proportionalable_aux(arr, n, 0);
}
```

```
bool Proportionalable_aux(int arr[], int n, int current)
{
        if (current > n) return true;

        for (int i = 0; i < 2 * n - current - 1; ++i)
        {
                if (arr[i] == UNSET && arr[i + current + 1] == UNSET)
                {
                        arr[i] = arr[i + current + 1] = current;
                        if (Proportionalable_aux(arr, n, current + 1))
                                return true;
                        arr[i] = arr[i + current + 1] = UNSET;
                }
        }

        return false;
}
```