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

## סמסטר חורף תשע"ח

## מבחן מסכם מועד א', 20 לפברואר 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 28 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:

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

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

צוות הקורס 234114/7

**מרצים**: פרופ' תומר שלומי (מרצה אחראי), פרופ' יוסף גיל, גב' יעל ארז **מתרגלים:** עמית אליהו, איתי הנדלר, ליאור כהן, בן לידרמן, תומר לנגה, גסוב מזאבי, נג'יב נבואני, צופית פידלמן, יורי פלדמן, 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.

```
#define LARGE_PRIME 65537
void f1(int n)
{
      int i = 1;
      while(i < n)
      {
            if((n - i) %2)
                  i *= 3;
            else
                  i *= 2;
      }
      for(int j = 0; j < n % LARGE_PRIME; ++j)
            putchar('#');
}
```

סיבוכיות זמן: $\Theta(\log n)$ 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.
הניחו שסיבוכיות הזמן של malloc(n) היא $\Theta(1)$, וסיבוכיות המקום של malloc(n) היא $\Theta(n)$.

```
int f2(int n)
{
      if(n <= 1)
            return 1;
      int* t = (int*) malloc(sizeof(int) * n);

      *t = f2((3 * n - 1) / 4);
      int result = *t;

      free(t);
      return result;
}
```

סיבוכיות זמן $\Theta(\log n)$: 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.

```
void f3(int n)
{
      int i = 2;
      while(i < n)
            i *= i;

      printf("%d\n", i);
}
```

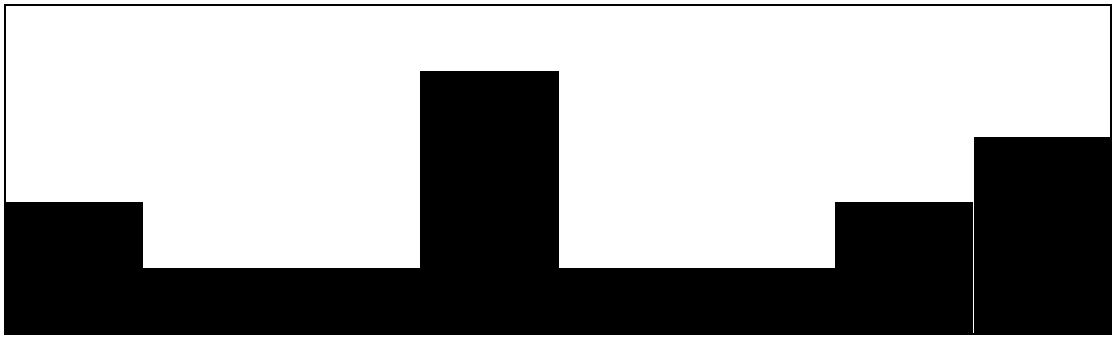
סיבוכיות זמן: $\Theta(\log\log n)$ סיבוכיות מקום: $\Theta(1)$

## שאלה 2 (25 נק׳)

נתון כלי לצבירת מי גשם. חתך הכלי מתואר ע"י מערך הגבהים `heights` באורך `n`.

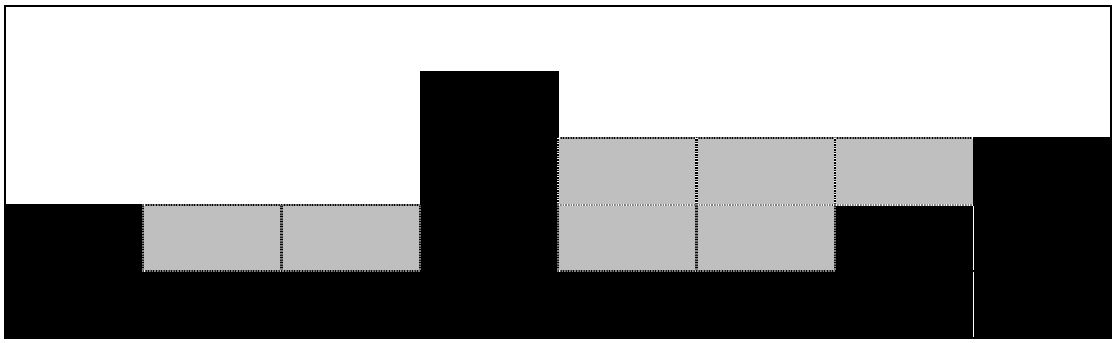
למשל עבור מערך הגבהים $\{2,1,1,4,1,1,2,3\}$

צורת הכלי היא:



יש לחשב את כמות הגשם שניתן לצבור בכלי עם מערך גבהים נתון (ניתן להניח שכל הגבהים חיוביים).

למשל עבור הכלי לעיל ניתן לצבור 7 יחידות גשם (האזורים האפורים).



הנחיה: יש לחשב בכל מקום בכלי מה גובה העמודה המקסימלית משמאלו ומימינו.

**דרישות**: סיבוכיות זמן $\mathbf{O(n)}$. סיבוכיות מקום $\mathbf{O(n)}$.

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

```
unsigned CalculateWaterVolume(int heights[],int n)
{ // O(1) space solution
      unsigned totalSum = 0, sum = 0;
      int left = 0, right = 1;
      // run from left to right
      while (right < n) {
            if (heights[left] <= heights[right]) {
                  totalSum += sum;
                  sum = 0;
                  left = right;
            }
            sum += (heights[left] - heights[right]);
            ++right;
      }
      // run back to the highest point
      --right;
      int back_left = right - 1;
      sum = 0;
      while (back_left > left) {
            if (heights[back_left] >= heights[right]) {
                  totalSum += sum;
                  sum = 0;
                  right = back_left;
            }
            sum += (heights[right] - heights[back_left]);
            --back_left;
      }

      totalSum += sum;
      return totalSum;
}
```

```
unsigned CalculateWaterVolume(int heights[],int n)
{
        unsigned* lefts = (unsigned *) malloc(sizeof(unsigned) * n);
        unsigned* rights = (unsigned *) malloc(sizeof(unsigned) * n);

        lefts[0] = heights[0];
        rights[n - 1] = heights[n - 1];

        for (int i = 1; i < n; ++i) {
                lefts[i] = max(lefts[i - 1], heights[i]);
                rights[n - i - 1] = max(rights[n - i], heights[n - i - 1]);
        }

        unsigned totalSum = 0;
        for (int i = 0; i < n; ++i) {
                totalSum += min(lefts[i], rights[i]) - heights[i];
        }

        free(rights);
        free(lefts);

        return totalSum;
}
unsigned min(unsigned lhs, unsigned rhs)
{
        return lhs < rhs ? lhs : rhs;
}
unsigned max(unsigned lhs, unsigned rhs)
{
        return lhs > rhs ? lhs : rhs;
}
```

## שאלה 3 (25 נקודות) :

במשחק Words in Word המטרה היא להרכיב כמה שיותר מילים שונות מאותיות של מילה נתונה. **שימו לב**, אם אות כלשהי מופיעה במילה הנתונה x פעמים, לא ניתן להרכיב מילים שמכילות אות זו יותר מ-x פעמים.

לדוגמה: בהינתן המילה `Technion` ניתן להרכיב את המילים הבאות (רשימה חלקית)

| | | |
|---|---|---|
| `toe` | `cent` | `techno` |
| `he` | `icon` | `ethnic` |
| `it` | `ice` | `ninth` |

אבל לא ניתן להרכיב את המילה `tent`, מאחר והיא מכילה שתי אותיות `t`, בעוד ישנה רק אות `t` אחת במילה הנתונה.

א. **(7 נק')** ממשו פונקציה

```
bool IsWord(char mainWord[], char testWord[])
```

המקבלת שתי מחרוזות: הראשונה `mainWord` מייצגת את המילה הנתונה, והשניה `testWord` המייצגת את המילה המועמדת להרכבה מהאותיות של המילה `mainWord`. הפונקציה תחזיר **`true`**, אם ניתן להרכיב את המילה המועמדת.

לדוגמה, נניח שהמילה הנתונה היא `Technion`,

עבור המילה המועמדת להרכבה `icon` תחזיר הפונקציה **`true`**

עבור המילה המועמדת להרכבה `iconic` היא תחזיר **`false`**

**דרישות**:

- סיבוכיות זמן: $\mathbf{O(m+n)}$ כאשר $m$, $n$ מייצגים את אורכי המילה הנתונה ומילת בדיקה. סיבוכיות מקום נוסף: $\mathbf{O(1)}$.
- ניתן להניח, כי שתי המילים מכילות אותיות אנגליות קטנות בלבד.

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

```
bool IsWord(char mainWord[], char testWord[])
{
#define ABC_SIZE 'z' - 'a' + 1

      int letters[ABC_SIZE] = { 0 };

      while (*mainWord)
            ++letters[*mainWord++ - 'a'];

      while (*testWord)
            --letters[*testWord++ - 'a'];

      for (int letter = 0; letter < ABC_SIZE; ++letter)
            if (letters[letter] < 0)
                  return false;

      return true;
}
```

**(18 נק')** ממשו פונקציה שתעריך את מספר המילים באנגלית שניתן להרכיב ממילה נתונה( על פי ההגדרה שבסעיף הקודם):

```
int WordScore(char *dictionary[], int n, char *word)
```

הפונקציה מקבלת את המילון dictionary שמכיל את כל המילים בשפה האנגלית, את מספר המילים שבו n, ומילה נתונה word.

לדוגמה, בהינתן מילון אנגלי ומילה נתונה "test", הפונקציה תחזיר 7, מאחר שניתן להרכיב מאותיות המילה "test" את כל המילים הבאות (שהן מילים חוקיות באנגלית, למרות שאת חלקן גם אנחנו לא הכרנו. לא כולל "test" עצמה):

| | | |
|---|---|---|
| te | tes | sett |
| | tet | est |
| | es | set |

**דרישות והנחות:**

- סיבוכיות זמן: $\mathbf{O(n+m)}$ כאשר $m$ מייצג את אורך המילה הנתונה, $n$ – מספר המילים במילון. סיבוכיות מקום נוסף: $\mathbf{O(1)}$.
- **שימו לב**, ניתן להניח, כי המילים במילון **קצרות**, לא תעלנה על 20 אותיות (למשל), אך לא ניתן להניח זאת על המילה הנתונה.

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

```
int WordScore(char *dictionary[], int n, char *word)
{
      int count = 0, letters[ABC_SIZE] = { 0 };
      FillHistogram(word, letters);
      for (int i = 0; i < n; ++i) {
            if (strcmp(word, dictionary[i]))
                  count += IsWord(letters, dictionary[i]);
      }
      return count;
}
```

```
void FillHistogram(char word[], int letters[ABC_SIZE])
{
	while (*word)
		++letters[*word++ - 'a'];
}

bool IsWord(int letters[ABC_SIZE], char testWord[])
{
	int testLetters[ABC_SIZE] = { 0 };

	FillHistogram(testWord, testLetters);
	for (int letter = 0; letter < ABC_SIZE; ++letter)
		if (letters[letter] < testLetters[letter])
			return false;

	return true;
}
// for the implementation of strcmp - see tutorial slides
```

## שאלה 4 (25 נקודות) :

אתם מתכננים ארוחת ערב חגיגית בבניין האו״ם. עקב תקריות דיפלומטיות כאלו ואחרות, חלק מנציגי המדינות לא מוכנים לשבת באותו שולחן עם נציגי מדינות מסוימות. יו״ר האו״ם מוכן להקצות לאירוע לכל היותר 42 שולחנות. לכן עליכם לבדוק האם קיימת חלוקה חוקית של הנציגים המוזמנים לאירוע למקסימום של ארבעים ושניים שולחנות ואם כן, לשבץ כל נציג בשולחן מתאים. חלוקה חוקית לשולחנות היא חלוקה בה כל נציג ישובץ לאחד מהשולחנות שיוקצו לאירוע (שולחן 0, שולחן 1, וכו') וכל יושבי שולחן מסוים מוכנים לשבת אחד עם השני.

עליכם לכתוב את הפונקציה:

```
bool SetTables(int vetoes[][M], int sitting[], int n)
```

הפונקציה תקבל כקלט מטריצה בגודל $n\ x\ M$ המייצגת את $n$ המדינות. המדינה ה-$i$ מיוצגת ע״י השורה ה-$i$ במטריצה. נציג כל מדינה יכול להטיל וטו על ישיבה באותו שולחן עם נציגי $M$ מדינות אחרות לכל היותר. הטלת וטו תתבצע ע״י כתיבת אינדקסי המדינות שאיתן אינו מוכן לשבת בשולחן בשורה שלו במטריצה (או 1-, אם אין).

אם קיימת ישיבה חוקית כלשהי, הפונקציה **תחזיר** `true`, תשבץ כל אחת מהמדינות לאחד השולחנות $0, 1, \ldots$, ותרשום את מספר השולחן בו שובצה המדינה ה-$i$ בתא המתאים לה (התא ה-$i$) במטריצה `sitting`. אם לא קיימת אף ישיבה חוקית, הפונקציה **תחזיר** `false` ותוכן המערך `sitting` אינו משנה.

לדוגמה עבור M=2, n=5:

| | | |
|---|---|---|
| 0 | 3 | 1 |
| 1 | 0 | -1 |
| 2 | -1 | -1 |
| 3 | 2 | -1 |
| 4 | -1 | 1 |

מדינה 0 לא מוכנה לשבת בשולחן עם מדינות 3,1.

מדינה 1 לא מוכנה לשבת בשולחן עם מדינה 0.

מדינה 2 מוכנה לשבת בשולחן עם כולם.

מדינה 3 לא מוכנה לשבת בשולחן עם מדינה 2.

מדינה 4 לא מוכנה לשבת בשולחן עם מדינה 1.

(1- מסמן תא ריק)

כאמור, השולחנות החוקיים ממוספרים 0, 1, ... ועליכם להחזיר במערך sitting את השולחן בו תושיבו נציג כל מדינה, אם קיים פתרון חוקי. אם קיימים מספר פתרונות, לא משנה איזה מהם תחזירו (כל עוד הוא חוקי).

למשל עבור המדינות בטבלה לעיל הפתרון הבא יהיה חוקי:

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 0 | 1 | 0 | 2 | 0 |

מכיוון שאף מדינה לא שובצה לאותו שולחן עם מדינה עליה הטילה וטו.

אבל הפתרון הבא לא יהיה חוקי מכיוון שמדינה 4 לא תסכים לשבת באותו שולחן עם מדינה 1

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |

**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 שהקלט תקין, כלומר שמטריצת `vetoes` מכילה רק מספרים בטווח בין 1- לבין n-1.
- ניתן ומומלץ להשתמש בפונקציות עזר (ויש לממש את כולן).
- ניתן להניח, כי מספר מקומות הישיבה ליד כל שולחן **אינו מוגבל**.

```
bool SetTables(int vetoes[][M], int sitting[], int n)
{
        return SetTablesAux(vetoes, sitting, n, 0);
}

#define TABLES_COUNT 42
```

```
bool VetoedBy(int vetoes[][M], int vetoer, int stateId)
{
        for (int foe = 0; foe < M; ++foe)
        {
                if (vetoes[vetoer][foe] == stateId)
                        return true;
        }
        return false;
}

bool ValidSitting(int vetoes[][M], int sitting[], int n, int stateId)
{
        for (int id = 0; id < stateId; ++id)
        {
                if (sitting[id] == sitting[stateId] &&
                                ( VetoedBy(vetoes, id, stateId) ||
                                 VetoedBy(vetoes, stateId, id) ) )
                        return false;
        }
        return true;
}
```

```
bool SetTablesAux(int vetoes[][M], int sitting[], int n, int stateId)
{
        if (stateId == n)
                return true;

        int highestIdOpened = 0;
        for (int i = 0; i < stateId; ++i)
            highestIdOpened = max(highestIdOpened, sitting[i]);

        int tableId;
        for (tableId = 0; tableId <= highestIdOpened + 1 &&
                                        tableId < TABLES_COUNT; ++tableId)
        {
                sitting[stateId] = tableId;
                if (ValidSitting(vetoes, sitting,n, stateId) &&
                            SetTablesAux(vetoes, sitting, n, stateId + 1))
                        return true;
        }
        return false;
}
```