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

סמסטר חורף תשע"ז

מבחן מסכם מועד ב', 21 למרץ 2017

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 16 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), מר איהאב וואתד, גב' יעל ארז **מתרגלים:** גב' דניאל עזוז, גב' צופית פידלמן, מר תומר לנגה, מר יובל בנאי, מר יואב נחשון, מר יורי פלדמן, מר דמיטרי רבינוביץ', מר שי וקנין, מר איתי הנדלר, מר יותם אשל

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.

```
void f1(int n){
  int i=1;
  while (i<n)
  {
     if  (i < 729*(n % 2))
           i += 2;
     else
           i *= 2;
  }
}
```

סיבוכיות זמן: Θ (log(n)) (4 נק') סיבוכיות מקום: Θ (1) (4 נק')

ב. (8 נקודות): חשבו את סיבוכיות הזמן והמקום של הפונקציה f2

```
void f2(int n){
  int i=1;
  while (i<n)
  {
     if  (i % 2)
           i += 2;
     else
           i=i*2+1;
  }
}
```

סיבוכיות זמן: Θ (n) (4 נק') סיבוכיות מקום: Θ (1) (4 נק')

ג. (9 נקודות): חשבו את סיבוכיות הזמן והמקום של הפונקציה f3

```
double g3(double n) {
    if (n<=1)
        return 2;
    double temp = g3(n/2);
    return temp*temp;
}

double f3(double n){
    return g3(g3(n));
}
```

סיבוכיות זמן: Θ (n) (5 נק') סיבוכיות מקום: Θ (n) (4 נק')

שאלה 2 (25 נק')

נתון מערך של n ספרות. כל ספרה בין 0 ל-9, כולל 0 ו-9.

הספרה השולטת על הספרה x היא הספרה הגדולה ביותר מבין כל הספרות שמופיעות מיד אחרי x במערך, בתנאי שהיא גדולה מ-x.

למשל עבור המערך:

3	5	3	7	9	4

הספרה השולטת על 7 היא 9.

הספרה השולטת על 3 היא 7 (שגדולה מ-5).

ל-5 אין ספרה שולטת כיוון ש-3 קטן מ-5.

סדרת השליטה על x מוגדרת כך:

- האיבר הראשון בסדרה הוא הספרה x.
- האיבר במקום i+1 הוא הספרה השולטת על האיבר במקום i.

למשל :

עבור המערך למעלה סדרת השליטה על 3 היא 3,7,9 כיוון ש:

7 היא הספרה השולטת על 3
9 היא הספרה השולטת על 7
ול-9 אין ספרה שולטת.

עבור המערך למעלה סדרת השליטה על 9 היא 9.

ממשו את הפונקציה:

```
int find_dominant(int a[], int n, int x)
```

אשר מקבלת מערך של ספרות a באורך n וספרה x ומוצאת את הספרה הגדולה ביותר בסדרת השליטה של x.
הערות:

- ניתן להניח כי x מופיע במערך.
- ניתן להניח כי כל איברי המערך הם ספרות בין 0 ל-9 (כולל).

דרישות: סיבוכיות זמן O(n) וסיבוכיות מקום נוסף O(1) כאשר n הוא אורך המערך.
אם לפי חישוביכם לא עמדתם בדרישות הסיבוכיות אנא ציינו כאן את הסיבוכיות שהגעתם אליה:
זמן _____ מקום נוסף _____

```
#define N 10


int find_dominant(int a[], int n, int x)
{
    int h[N] = {0};
    for (int i=0; i<n-1; i++)
    {
        if (a[i+1] > h[a[i]])
            h[a[i]] = a[i+1];
    }
    while (h[x] > x)
        x = h[x];
    return x;
}
```

שאלה 3 (25 נקודות) :

מערך מחרוזות **ממוין** הוא מערך בו המחרוזות מסודרות בסדר לקסיקוגרפי עולה (לפי ערכי ASCII).

מערך מחרוזות **ממוין ציקלית** מקיים את שני התנאים הבאים:

1. המערך הוא שרשור של שני תתי מערכים ממוינים (אחד מהם יכול להיות ריק).
2. כל המילים בתת המערך הימני קטנות (מבחינת סידור לקסיקוגרפי) מכל המילים בתת המערך השמאלי.

למשל המערך הבא הוא מערך ממוין ציקלית:

"ghost"	"tiger"	"zebra"	"bear"	"cow"

תת מערך שמאלי — תת מערך ימני

שימו לב שגם מערך ממוין הוא מערך ממוין ציקלית:

"bear"	"cow"	"ghost"	"tiger"	"zebra"

במקרה זה אחד מתתי המערכים הוא בגודל אפס.

עליכם למשש את הפונקציה:

```
int find_min(char *arr[], int n);
```

אשר מקבלת מערך מחרוזות arr **ממוין ציקלית** ואת אורכו n, ומחזירה את האינדקס של **המחרוזת הקטנה ביותר** (כלומר המחרוזת שמופיעה ראשונה במילון).

למשל עבור המערך הבא:

"ghost"	"tiger"	"zebra"	"bear"	"cow"

הפונקציה תחזיר 3, כיוון שהמילה "bear" מופיעה ראשונה במילון.

עבור המערך הבא:

"bear"	"cow"	"ghost"	"tiger"	"zebra"

הפונקציה תחזיר 0, כיוון ששוב המילה "bear" מופיעה ראשונה במילון.

הערות:

- ניתן להניח שאין במערך 2 מילים זהות.
- אסור לשנות את המחרוזות אפילו לא באופן זמני.
- ניתן להניח שהמחרוזות מורכבות מאותיות אנגליות קטנות בלבד.
- ניתן להניח שהמערך arr אינו ריק.
- מותר להשתמש בפונקציה strcmp(char* s1, char* s2) מהספריה string.h.
  הפונקציה מחזירה 0 אם המחרוזות זהות, מספר שלילי אם s1 קטנה מ s2 ומספר חיובי אם s2 קטנה מ s1.

דרישות: סיבוכיות זמן O(m log n) כאשר m מסמל את אורך המחרוזת הארוכה ביותר במערך, ו – n את אורך המערך. סיבוכיות מקום O(1).

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

```
int mycmp(char* s1, char* s2) {
    for (;*s1 && *s2 && *s1 == *s2; s1++, s2++);
    return *s1-*s2;
}
int find_min(char *arr[], int n) {
    if (n <= 1 || mycmp(arr[0], arr[n-1]) < 0)  // sorted array
        return 0;

    // if we're here, we have at least 2 strings and the minimal string is not in idx 0
    int l=0, r=n-1, m=0;
    while (l < r)
    {
        m = (l+r)/2;
        if (mycmp(arr[m], arr[m+1]) > 0)    // arr[m] > arr[m+1]
            return (m+1);
        else if (mycmp(arr[0], arr[m]) > 0) // arr[0] > arr[m]
            r = m;
        else                                // arr[0] < arr[m]
            l = m+1;
    }
    // we shouldn't get here
    return -1;
}
```

שאלה 4 (25 נקודות) :

נתון מערך דו מימדי בגודל NxN שמכיל ערכים בוליאניים - true/false.

איזור true הוא אוסף מקסימלי של תאים סמוכים שכולם בעלי ערך true.
תאים הממוקמים **באלכסון** זה לזה **לא** נחשבים לסמוכים.

למשל עבור המערך הבא כאשר N=5:

0	0	0	0	1
0	1	1	1	0
0	0	1	1	0
1	0	0	0	0
1	1	0	0	0

קיימים 3 איזורי true:

0	0	0	0	**1**
0	**1**	**1**	**1**	0
0	0	**1**	**1**	0
1	0	0	0	0
1	**1**	0	0	0

ממשו את הפונקציה:

```
int cntTrueRegions(bool matrix[N][N]);
```

הפונקציה מקבלת מטריצה ריבועית NXN, ומחזירה כמה איזורי true שונים קיימים במטריצה.

<u>הערות</u>:

1. איזור true מורכב לכל הפחות מתא אחד.
2. במידה ולא קיימים איזורי true במערך יוחזר 0.
3. על הפונקציה להשתמש **ברקורסיה** כדי לפתור את הבעיה.

```
bool islegal(int r, int c) {
   return (r >= 0 && r < N && c >= 0 && c < N);
}
void disable(bool matrix[N][N], int r, int c) {
   if (!islegal(r,c) || matrix[r][c] == false)
      return;
   matrix[r][c] = false;
   disable(matrix, r, c-1);
   disable(matrix, r, c+1);
   disable(matrix, r-1, c);
   disable(matrix, r+1, c);
}
int cntTrueRegions(bool matrix[N][N]) {
   int counter = 0;
   for (int r=0; r<N; r++) {
      for (int c=0; c<N; c++) {
         if (matrix[r][c] == true) {
            counter++;
            disable(matrix, r, c);
         }
      }
   }
   return counter;
}
```

בהצלחה!