

סמסטר חורף תשע"ח 2018

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

סמסטר חורף תשע"ח

מבחן מסכם מועד ב', 18 למרץ 2018

2	3	4	1	1	רשום/ה לקורס:					מספר סטודנט:
					1					i e e e e e e e e e e e e e e e e e e e

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 תיבדקנה.
- יש לכתוב באופן ברור, נקי ומסודר. <u>ניתן בהחלט להשתמש בעיפרון ומחק,</u> פרט לדף השער אותו יש לכתוב באופן ברור, נקי ומסודר. יש למלא בעט.
- 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:

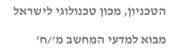
$$1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} = \Theta(\log n) \qquad 1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \frac{1}{25} + \dots = \Theta(1)$$

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

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 (1) המוגדרת בקטע הקוד הבא, מפונקציה של (n) אין צורך לפרט שיקוליכם. חובה לפשט את הביטוי ככל שניתן.

```
void f1(int n)
{
    int i = 1, j = 0;
    while (j < n)
    {
        j += i;
        i += i;
    }
    for (; i * i < n; ++i)
        printf("*");
}</pre>
```

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

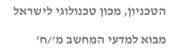
ב. $(9 \ tilde{tglish} \ til$

```
int f2(int n) {
    if(n < 3)
        return 1;

    int* arr = (int*) malloc(sizeof(int) * n);
    f2(f2(n - 3));
    free(arr);

return n;
}</pre>
```

 $\underline{\Theta(n^2)}$:סיבוכיות מקום $\underline{\Theta(2^{\frac{n}{3}})}$ סיבוכיות זמן



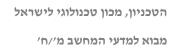




ג. (8) נקודות): חשבו את סיבוכיות הזמן והמקום של הפונקציה f המוגדרת בקטע הקוד הבא, כפונקציה של g. אין צורך לפרט שיקוליכם. <u>חובה לפשט את הביטוי ככל שניתו</u>. הניחו, כי g סיבוכיות זמן ומקום של הפונקציות g g של g מחזירה לוגריתם בבסיס g של g מעוגל כלפי מטה.

```
int aux(int n, int x) {
     if (x == 0)
           return n;
     int val;
     if(x % 2)
            val = aux(n * 2, x / 2);
     else
            val = aux(n / 2, x / 2);
     return val;
}
void f3(int n) {
     int m, i = log2(n);
     m = aux(4, i);
     for(i = 0; i < m; ++i)
           putchar('*');
}
```

 $\underline{\Theta(\ \log\log n\)}$ סיבוכיות מקום: $\underline{\Theta(\ \log n\)}$





1	



(נקי) אאלה 2 (25 נקי)

ממשו את הפונקציה FindDuplicate המקבלת מערך n, ואת אורכו החזירה המספר כלשהו FindDuplicate המופיע במערך פעמיים או יותר. המערך לא בהכרח ממוין, ומכיל מספרים שלמים בטווח בין 1 לבין n-1.

:דוגמאות

4 בהינתן מערך $\{1,2,3,4,5,4\}$, הפונקציה תחזיר

בהינתן מערך $\{1,1,1,2,2,2\}$, הפלטים האפשריים של הפונקציה הם 1 או 2.

דרישות:

- . $\underline{\mathbf{0}(1)}$ על הפונקציה לעמוד בסיבוכיות זמן און, $\underline{\mathbf{0}(n)}$, בסיבוכיות מקום
- פתרונות בסיבוכיות זמן/מקום גרועה מהנדרש יזכו לניקוד חלקי במידה והם נכונים.

שימו לב: ניתן לשנות את איברי המערך.

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



```
int FindDuplicate(int a[], int n)
\{//\ O(n)\ +\ O(1)\ solution, changing the array, by Itamar Ginsberg & Michael Amir
       for (int i = 0; ; ++i) {
              int x = abs(a[i]);
              if (a[x] < 0)
                     return x;
              a[x] *= -1; // mark index as seen
       }
int FindDuplicate(const int a[], int n)
\{// \ O(n \log n) + O(1) \ solution, w/o \ change, by --= dEmigOd =--
       int mRange, hRange = n - 1, lRange = 1;
       while (hRange > lRange) {
              mRange = lRange + (hRange - lRange) / 2; // split range in two
              int count = 0;
              for (int i = 0; i < n; ++i) // count elements in range</pre>
                     count += (a[i] >= lRange) && (a[i] <= mRange);
              if (count > mRange - lRange + 1) // drop range with not enough
elements
                     hRange = mRange;
              else
                     lRange = mRange + 1;
       return lRange;
}
```



```
int FindDuplicate(const int arr[])
\{ // O(n) + O(1), w/o \text{ changes, by --= dEmigOd =--} \}
       const int *prevFast, *fastPtr = arr;
       const int *slowPtr;
       slowPtr = arr;
       do { // find the loop in linked list
              slowPtr = GetNext(arr, slowPtr);
              fastPtr = GetNext(arr, fastPtr);
              fastPtr = GetNext(arr, fastPtr);
       } while (slowPtr != fastPtr);
       slowPtr = arr;
       do { // find the entry point to that loop
              prevFast = fastPtr; // preserve the index of possible entry
              slowPtr = GetNext(arr, slowPtr);
              fastPtr = GetNext(arr, fastPtr);
       } while (slowPtr != fastPtr);
       return *prevFast;
}
const int* GetNext(const int arr[], const int* curr)
{ // kinda a[a[0]] in Elinor's solution
       return arr + *curr;
}
```







: (שאלה 3 (25 נקודות)

. מחרוזת תקרא ${f k}$ שעמים או יותר על עצמו אחד בה חוזר על עצמו או יותר

 ${\sf e}$ למשל, ${\sf deadbeef}$ היא מחרוזת 3-שופעת, אך לא מחרוזת 4-שופעת, מאחר והתו הכי נפוץ בה מופיע שלוש פעמים.

.שופעת, אך לא -k-שופעת, אך לא א-שופעת. אם היא (-1)-שופעת.

בדוגמה לעיל, המחרוזת היא 4-חסרה.

ממשו פונקציה GetKShortSubstring שמקבלת מחרוזת str שמקבלת שמקבלת התת-מחרוזת הערכה אורך התת-מחרוזת k נתון. אם תת-מחרוזת כזו לא קיימת הפונקציה תחזיר k נתון.

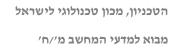
למשל, בדוגמה שמלווה אותנו, בעבור k=3, הפונקציה תחזיר 6, כי משל, בדוגמה שמלווה אותנו, בעבור k=3, האונה על הדרישות היא adbe וגם eadb ולכן הפונקציה תחזיר k=2 תת-מחרוזת העונה על הדרישות היא k=2 חסרה. k=2

דרישות:

- על הפונקציה לעמוד בסיבוכיות זמן 0(n) , וסיבוכיות מקום 0(1), כאשר 0 הוא אורך מחרוזת הקלט.
 - אפשר להניח שכל האותיות במחרוזת הקלט הן קטנות.
 - . פתרונות בסיבוכיות זמן/מקום גרועה מהנדרש יזכו לניקוד חלקי במידה והם נכונים.
 - שימו לב, כי תת-מחרוזת זהו **רצף** של תווים.



```
#define ABC_SIZE 'z' - 'a' + 1
int GetKShortSubstring(const char* str, int k)
{ // No one send a solution
       const char* fwdPtr = str - 1, *bckPtr = fwdPtr;
       int hist[ABC_SIZE] = { 0 };
       int longestSubstringSeen = 0;
       while (*++fwdPtr) {
              if (++hist[*fwdPtr - 'a'] == k) { // enlarge to the right as far}
                     if (fwdPtr - bckPtr - 1 > longestSubstringSeen)as possible
                            longestSubstringSeen = fwdPtr - bckPtr - 1;
                     do { //shorten from the left, until last letter hit
                            ++bckPtr;
                            --hist[*bckPtr - 'a'];
                     } while (*bckPtr != *fwdPtr);
              }
       for (char letter = 'a'; letter <= 'z'; ++letter) { // handle suffixes</pre>
              if (hist[letter - 'a'] == k - 1) {
                     if (fwdPtr - bckPtr - 1 > longestSubstringSeen)
                            longestSubstringSeen = fwdPtr - bckPtr - 1;
                     break;
              }
       }
       return longestSubstringSeen;
}
```











: (שאלה 4 (25 נקודות)

טקס האוסקר מתקרב, וכל מוזמנת צריכה לבחור צבע לשמלה. ממשו פונקציה המקבלת נתונים על העדפות צבע השמלה של N מוזמנות לטקס ומחזירה את מספר הצבעים המינימלי הדרוש לפי הכללים בכאים:

כל מוזמנת מציינת קבוצה של מוזמנות אחרות שצריך להיות להן את אותו צבע שמלה כמו שלה, $color_map[N][N][N]$, הדו ממדי [M] $color_map[N][N][N]$, ערך בשורה ה i במקום ה j, ערך של 1 מציין דרישה של מוזמנת i לאותו צבע שמלה כמו למוזמנת j, ערך של 1- מציין דרישה לצבע שונה, וערך 0 מציין שאין דרישה כלשהי.

ערך החזרה: מספר הצבעים המינימלי הדרוש, העונה על הדרישות (או 1- במקרה שאין כזה).

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 בפונק' עזר (ויש לממש את כולן).

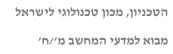
#define NO_SOLUTION -1
#define SHOULD_WEAR_SAME_COLOR 1
#define SHOULD_WEAR_DIFFERENT_COLOR -1
#define NO_COLOR 0
<pre>int FindMinimumNumDressColors(int color_map[N][N])</pre>
{ // based on the solution by Elinor Ginzburg and Yair Reani
<pre>int chosen_colors[N] = { NO_COLOR };</pre>
return colorAux(color_map, chosen_colors, 0, NO_COLOR);
}



```
int colorAux(int color_map[N][N], int chosen_colors[], int curr_actress, int
maxUsedColor)
       if (curr_actress == N)
              return maxUsedColor;
       int minColorsNeeded = N + 1;
       for (int color = 1; color <= maxUsedColor + 1; ++color) {</pre>
              chosen_colors[curr_actress] = color;
              if (!IsLegalColorAssignment(color_map, chosen_colors))
                     continue;
              int colorsInSolution = colorAux(color_map, chosen_colors,
curr_actress + 1, Max(maxUsedColor, color));
              if(colorsInSolution != NO_SOLUTION)
                     minColorsNeeded = Min(minColorsNeeded, colorsInSolution);
       }
       chosen_colors[curr_actress] = NO_COLOR;
       return minColorsNeeded > N ? NO_SOLUTION : minColorsNeeded;
}
bool ShouldWearSameColor(int color_map[N][N], int ba, int sa)
{
       return color_map[ba][sa] == SHOULD_WEAR_SAME_COLOR;
} //similar functionality is omitted further
```



```
bool IsLegalColorAssignment(int color_map[N][N], int chosen_colors[N])
{ //check all actresses who got their color
       for (int ba = 0; chosen_colors[ba] != NO_COLOR && ba < N; ++ba)</pre>
       { //best actress
              for (int sa = 0; chosen colors[sa] != NO COLOR && sa < N; ++sa)</pre>
              { //supporting actress
                     if (ba == sa) continue;
                     if ((ShouldWearSameColor(color_map, ba, sa) &&
                                    WearDifferentColor(chosen_colors, ba, sa)) ||
                             (ShouldWearDifferentColor(color_map, ba, sa) &&
                                    WearSameColor(chosen_colors, ba, sa)))
                             return false;
              }
       //if we checked all actresses and there wasn't contradiction or any
problem with this color. she can wear it
    return true;
}
```





1	