



## מבוא למדעי מחשב מ' / ח' (234117 / 234114)

### סמסטר חורף תשע"ה

### מבחן מסכם מועד א', 17 פברואר 2015

2	3	4	1	1	
---	---	---	---	---	--

רשום/ה לקורס:

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 14 עמודים (4 שאלות) במבחן, כולל עמוד זה.
- וודאו שאתם נבחרים בקורס המתאים.
- 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 ומחקק, פרט לדף השער אותו יש למלא בעט.
- חובה לקרוא הוראות לכתיבת קוד המופיעות בעמוד הבא לפני פתרון המבחן.
- 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:

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

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

צוות הקורס 234114/7

**מרצים:** פרו"פ/ח תומר שלומי (מרצה אחראי), יחיאל קימחי, איתן יעקובי, אנסטסיה דוברובינה.



## הנחיות לכתיבת קוד במבחן

- בכל השאלות, הנכם רשאים להגדיר ולממש פונקציות עזר כרצונכם. לנוחיותכם, אין חשיבות לסדר מימוש הפונקציות בשאלה, ובפרט ניתן לממש פונקציה לאחר השימוש בה. מותר להשתמש בפונקציה שנכתבה בסעיף אחר, בתנאי שתציינו באופן ברור איפה הפונקציה ממומשת.
- **חובה** להקפיד על תכנות מבני (כלומר, חלוקה נכונה לפונקציות). אם כתבתם פונקציה שאורכה יותר מ 22 שורות, זוהי אינדיקציה ברורה לכך שיש לפרק את הפתרון לפונקציות עזר. אורך הפונקציה נמדד בהתאם להנחיות שניתנו בשיעורי בית.
- 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.
- ניתן להשתמש בהקצאות זיכרון בסגנון C99 (מערכים בגודל משתנה), בכפוף לדרישות סיבוכיות זיכרון.

**בהצלחה!**



## שאלה 1 (25 נקודות):

א. (8 נקודות) חשבו את סיבוכיות הזמן והמקום של הפונקציה  $f$  המוגדרת בקטע הקוד הבא, כפונקציה של  $n$ . אין צורך לפרט שיקולים. חובה לפשט את הביטוי ככל שניתן.

```
void f(int n)
{
    int j, s;
    for(j=0, s=1; s<n; j++, s*=2)
        printf("!");
    double values[j];
    for(int k=0; k<j; k++)
        values[k]=0;
    while(j--)
        for(int k=1; k<j; k++)
            values[k]+=1.0/k;
}
```

סיבוכיות זמן:  $\Theta(\quad)$       סיבוכיות מקום:  $\Theta(\quad)$

ב. (9 נקודות): חשבו את סיבוכיות הזמן והמקום של הפונקציה  $f()$

```
void f(int n)
{
    for (int i=1; i<=n; i++)
        for (int j=1; j<=n*n/i; j+=i)
            printf("*");
}
```

סיבוכיות זמן:  $\Theta(\quad)$       סיבוכיות מקום:  $\Theta(\quad)$

ג. (8 נקודות): חשבו את סיבוכיות הזמן והמקום של הפונקציה  $f()$  (ולא של הפונקציה  $g$ )

```
void g(int n, int i){
    if(i*i>n) return;
    g(n, i+1);
    printf("#");
}
void f(int n)
{
    g(n, 0);
    g(n, n/2);
}
```

סיבוכיות זמן:  $\Theta(\quad)$       סיבוכיות מקום:  $\Theta(\quad)$





## שאלה 2 (25 נקודות):

מחרוזת תקרא "חדשה-עולה" אם כל אות "חדשה", שזו הפעם הראשונה שהיא מופיעה במחרוזת, גדולה יותר מכל האותיות הקודמות במחרוזת (החל מהאות השנייה). לדוגמא: "aba" היא חדשה-עולה כי  $a < b$  ו- $b > a$  שבמקום האחרון במחרוזת כבר הופיעה לפני כן. "acabf" אינה חדשה-עולה כי b אות חדשה אבל יותר קטנה מהאות c שקדמה לה.

עליכם לממש פונקציה המקבלת מחרוזת המכילה אותיות קטנות ומחזירה את אורך תת המחרוזת הרצופה הארוכה ביותר שלה שהיא חדשה-עולה.

חתימת הפונקציה:

```
int max_new_increase_substr(char* string)
```

דוגמא: על המחרוזת "abfaaczca" הפונקציה תחזיר 6 כי תת המחרוזת בעלת המספר הגדול ביותר של אותיות שהיא חדשה-עולה היא "aaczca", המכילה 6 אותיות.

ניתן להניח כי מחרוזת הקלט לא ריקה, וכי היא מכילה רק אותיות אנגליות קטנות.

דרישות סיבוכיות: סיבוכיות זמן צריכה להיות  $O(n^2)$ . סיבוכיות המקום צריכה להיות  $O(1)$ .

```
int max_new_increase_substr(char* string)
```

```
{
```





## ממשו את הפונקציה שחתימתה

```
bool mul_elements(int a[], int b[], int len_a, int len_b, int goal);
```

הפונקציה מקבלת כקלט שני מערכים ממויינים  $a$  ו  $b$  של מספרים שלמים באורך  $len_a$  ו  $len_b$  (בהתאמה) ומספר שלם  $goal$ . הפונקציה מחפשת שני איברים  $a[i]$  - במערך  $a$ , ו-  $b[j]$  במערך  $b$ , כך שמכפלתם תהיה שווה בדיוק ל- $goal$ :  $a[i] * b[j] = goal$ .

אם נמצאו איברים מתאימים יש להחזיר `true`, אחרת יש להחזיר `false`.

המערכים ממויינים בסדר עולה, כלומר  $a[i] \leq a[i+1]$  ו  $b[i] \leq b[i+1]$ .

**דרישות:** על הפונקציה לעבוד בסיבוכיות זמן  $O(len_a + len_b)$  וסיבוכיות מקום נוסף  $O(1)$ .

```
bool mul_elements(int a[], int b[], int len_a, int len_b, int goal)
```

```
{
```







## שאלה 4 (25 נקודות) :

נתון מבוך בגודל  $N \times N$ , המיוצג כמערך דו-ממדי של שלמים. כל איזור במבוך יכול להיות פנוי, קיר (חסום), או להכיל דרקון. מכל איזור ניתן להתקדם ימינה, שמאלה, למעלה או למטה (אך לא באלכסון), ולא ניתן לעבור דרך קיר.

המטרה היא למצוא מסלול שמתחיל בנקודה (0,0) ומסתיים בנקודה (N-1, N-1), ועובר דרך כמה שפחות דרקונים. הפונקציה צריכה להחזיר את מספר הדרקונים במסלול שמצאה.

קיר מסומן ע"י הערך 0, איזור פנוי מסומן ע"י 1, ודרקון מסומן ע"י הערך 3.

ניתן לשנות את המערך (המבוך) במהלך ריצת הפונקציה. בסיום ריצת הפונקציה אין חובה שהמערך יכיל ערכים בעלי משמעות.

דוגמה: עבור המערך המוצג מימין, קיים פתרון העובר דרך דרקון אחד בלבד המסומן באפור. לכן הפונקציה צריכה להחזיר 1. עבור המערך המוצג משמאל הפתרון הכי טוב עובר דרך 4 דרקונים, לכן הפונקציה תחזיר 4.

1	1	3	3
3	0	1	1
3	0	0	0
1	3	3	1

1	1	3	3
3	0	1	1
3	0	0	1
1	3	3	1

אם אין אף פתרון חוקי יש להחזיר את הקבוע IMPOSSIBLE השווה ל  $N \cdot N + 1$ .

## הערות:

- ניתן להשתמש בקבועים הבאים:  $N$ , IMPOSSIBLE, WALL=0, EMPTY=1, DRAGON=3
- יש להשתמש בשיטת backtracking כפי שנלמדה בכיתה.
- בשאלה זו אין דרישות סיבוכיות, אולם כמקובל ב-backtracking יש לוודא שלא מתבצעות קריאות רקורסיביות מיותרות.

```
int min_dragons_in_maze(int maze[N][N])
```

```
{
```









