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

סמסטר אביב תשע"ה

מבחן מסכם מועד ב', 18 ספטמבר 2015

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 18 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.
- ניתן להשתמש בהקצאות זכרון בסגנון C99 (מערכים בגודל משתנה), בכפוף לדרישות סיבוכיות זכרון.
- 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.
- בשאלות 2, 3, 4 תשובה "לא יודע" (ללא תוספות) תזכה את הנבחן ב- 4 נקודות. כדאי לכתוב "לא יודע" אם אתם יודעים שאתם לא יודעים את התשובה. (לא משתלם "לנחש" קוד).
- נוסחאות שימושיות: $1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n}=\Theta(\log n)$ $\quad 1+\frac{1}{4}+\frac{1}{9}+\frac{1}{16}+\frac{1}{25}+...=\Theta(1)$

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

צוות הקורס 234114/7
מרצים: פרופ' ניר אילון (מרצה אחראי), פרופ' מירי בן-חן, ד"ר רוני קופרשטוק
מתרגלים: מר נחשון כהן (מתרגל ראשי), גב' דניאל עזוז, גב' גילי יבנה, מר שרגא לבציון, מר מאור ינקוביץ', מר יפתח זיסר ,מר איהאב וותד

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 שיקוליכם. חובה לפשט את הביטוי ככל שניתן.

```
void f(int n)
{
   for(int i=0; i<n; ++i)
        for(int j=0; j<i; ++j)
           for(int k=i*j; k>0; k/=2)
                 printf("~");
}
```

סיבוכיות זמן (6 נק): Θ (_____) סיבוכיות מקום (2 נק): Θ (_____)

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

```
double f(int n){
    double result = 0.0;
    int i;
    for (i = 1; (1.0/i)-(1.0/(i+1)) > 1.0/n; i++)
       result += 1.0/(i*i);
    return result;
}
```

סיבוכיות זמן: (6 נק) Θ (_____) סיבוכיות מקום (3 נק): Θ (_____)

ג. (8 נקודות): חשבו את סיבוכיות הזמן והמקום של הפונקציה f() (כפונקציה של n)

```
void g(int n){
    int i=0;
    char str[26] = {0};
    do {
        str[i] = 'a'+i;
        i++;
    } while (i < 26 && i < n);
    f(n);
}
void f(int n){
    if(n<1)
        return;
    g(n/5);
}
```

סיבוכיות זמן (4 נק): Θ (_____) סיבוכיות מקום (4 נק): Θ (_____)

שאלה 2 (25 נקודות):

ממשו את הפונקציה שחתימתה:

```
bool my_compare_str(char* s1, char* s2);
```

הפונקציה מקבלת שתי מחרוזות המורכבות מאותיות באנגלית ('a'..'z', 'A'..'Z'), ורווחים (' '). הפונקציה צריכה להשוות בין המחרוזות, ולקבוע האם הן זהות כאשר הפונקציה תתעלם מהבדלים בגודל האות (case insensitive), וכמו כן תחשיב רצפי רווחים בכל אורך (לפחות אחד), **כאילו היו רווח יחיד**.

לדוגמא, שתי המחרוזות הבאות נחשבות **זהות**:

```
"hello have a nice day"
"hello            have A NICE day"
```

לכן עבורן הפונקציה תחזיר TRUE.

לעומת זאת, כל המחרוזות הבאות **שונות** זו מזו:

```
"cat"
"   cat"
"cat   "
"car"
"ca  t"
```

דרישות: יש לממש פונקציה רקורסיבית, **אין להשתמש בלולאות**. (כלומר: אין להשתמש במילים השמורות for, while).

ניתן להשתמש בפונקציות עזר. אסור לשנות את תוכן מחרוזות הקלט.

סיבוכיות זמן: עבור שתי מחרוזות שאורכן m, n, סיבוכיות הזמן צריכה להיות O(m+n)

סיבוכיות מקום נוסף: O(m+n)

אין צורך לבדוק את תקינות הקלט.

`bool my_compare_str(char* s1, char* s2);`
`{`

שאלה 3 (25 נקודות):

מספר שלם חיובי n הוא 50-חלק אם כל הגורמים הראשוניים שלו הם מספרים בתחום 2..50.
במילים אחרות, מספר הוא 50-חלק אם ניתן לכתוב אותו כמכפלה $n=a_1*a_2*\ldots*a_k$ של מספרים, כאשר כל a_i הוא מספר ראשוני בין 2 ל- 50 לכל i=1..k. לדוגמא, המספר 100 הוא 50-חלק כי ניתן לכתוב אותו כ 5*5*2*2 והמספרים 2,5 ראשוניים בתחום [2..50]. לעומת זאת, המספר 106 לא 50-חלק, כי יש לו גורם ראשוני גדול מדי (53).

כיתבו פונקציה is_smooth50 שחתימתה להלן, אשר מחזירה TRUE אם מספר n הוא 50-חלק, אחרת FALSE. מותר (ולא חובה) להשתמש ברקורסיה. בשאלה זו אין דרישות של סיבוכיות זמן ומקום. עליכם:

a) לכתוב תכנית שעובדת (כלומר תחזיר תשובה נכונה לכל קלט n>=2) .
[סעיף זה שווה 70% מהניקוד של השאלה]

b) לחשב נכון את סיבוכיות המקום והזמן של הפיתרון **שלכם**. [סעיף זה שווה 30%, אבל רק אם סעיף a נפתר בהצלחה, או שלפחות ברור מהקוד שלכם מה האלגוריתם שניסיתם לממש, והאלגוריתם נכון. אחרת, הניקוד על סעיף זה 0%].

סיבוכיות זמן: Θ (_____) סיבוכיות מקום: Θ (_____)

```
bool is_smooth50(int n)
{
```

שאלה 4 (25 נקודות) :

ישנם N מועמדים להשתתף במשחקי הרעב. מועמדים אלו נדרשים להסתדר בתור להרשמה. כל מועמד מיוצג על-ידי מספר שלם בין 0 ל – N-1. התור מיוצג על-ידי מערך של מספרים שלמים, בו המועמד הראשון בתור נמצא באינדקס 0, המועמד השני בתור באינדקס 1 וכך הלאה.
עקב אווירת החשדנות הכללית, אף אחד לא מוכן שאחד מ 3 האנשים שמאחוריו בתור יהיה מישהו שמאיים עליו. כלומר, האחרון בתור אדיש לסדר של התור, המועמד שלפני האחרון לא מוכן שהמועמד האחרון יאיים עליו, ובאופן כללי מועמד במקום i לא מוכן שהמועמדים במקום i+1, i+2, i+3 יאיימו עליו.
נתונה מטריצה דו מימדית threats המתארת את יחסי האיום בין המועמדים. במיקום threats[i][j] מסומן 1 אם מועמד j מאיים על מועמד i ו 0 אחרת. לא ניתן להניח שהמטריצה סימטרית, כלומר i יכול לאיים על j בלי ש j יאיים על i.

עליכם לכתוב פונקציה

```
bool orderQueue(int threats[N][N], int queue[N]);
```

המקבלת את מטריצת האיומים וכותבת לתוך המערך queue את מספר המועמד בכל מקום בתור.
אם לא ניתן לסדר את התור יש להחזיר false. אחרת יש להחזיר true.

לדוגמה, עבור N=4 המטריצה:

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

קיים פתרון אחד, התור (משמאל לימין):

0	1	2	3

והפונקציה תחזיר true.

הסבר: מועמד 3 מאוים מכל שאר המתמודדים, ולכן הוא חייב להיות אחרון (אף אחד לא יכול לעמוד אחריו).
מועמד 2 מאוים ממתמודדים 0 ו – 1 ולכן חייב להיות לפני האחרון.
מועמד 1 מאוים ממועמד 0 ולכן מועמד 1 יהיה שני בתור, ו – 0, עליו אף אחד לא מאיים, יהיה ראשון.

לעומת זאת, עבור המטריצה:

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

אין פיתרון (כי שחקנים 0 ו- 1 מאיימים אחד על השני, ויש רק ארבעה מקומות בתור) והפונקציה תחזיר false.

הערות:

- N מוגדר ע"י `#define`.
- ניתן להניח שבאלכסון הראשי של המטריצה threats יש אפסים בלבד.
- 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.

```
bool orderQueue(int threats[N][N], int queue[N])
{
```