

מבוא למדעי מחשב מ' / ח' (234114 / 234117) סמסטר חורף תשס"ח

מבחן מסכם מועד א', 7 פברואר 2008

שם פרטי	,	שם משפחה				נט	טודו	מספ	

משך המבחן: 3 שעות.

חומר עזר: אין להשתמש בכל חומר עזר בכתב, מודפס או אלקטרוני.

הנחיות והוראות:

- מלאו את הפרטים בראש דף זה.
- בדקו שיש 24 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.
 - אין לכתוב הערות והסברים לתשובות אם לא נתבקשתם מפורשות לכך.
 - בכל השאלות, הינכם רשאים להגדיר ולממש פונקציות עזר כרצונכם.
 - אין להשתמש בפונקציות ספריה או בפונקציות שמומשו בכיתה אלא אם צוין אחרת בשאלה.
 - פתרון שלא עומד בדרישות הסיבוכיות יקבל ניקוד חלקי בלבד.

צוות הקורסים 234114/7	2341	14/7	הקורסים	צוות
-----------------------	------	------	---------	------

מרצים: פרופ'/ח' מיכאל אלעד (מרצה אחראי), סאהר אסמיר, ד"ר צחי קרני, רן רובינשטיין.

מתרגלים: אלדר אהרוני, גדי אלקסנדרוביץ', רון בגלייטר, שגיא בן-משה, אורי זבולון, מרק זילברשטיין, סשה סקולוזוב, ולנטין קרבצוב, אייל רגב, אייל רוזנברג, אנדרי קלינגר (מתרגל אחראי).

בודק	הישג	ערך	שאלה
		25	1
		25	2
		25	3
		25	4
		100	סה"כ

בהצלחה!



l	



<u>שאלה 1 (25 נקודות)</u>

סעיף א

בכל אחד מהסעיפים הבאים מופיעות מספר שורות קוד. לכל קטע קוד, הקיפו בעיגול את התיאור המתאים והסבירו את בחירתכם בקצרה:

- א. ללא שגיאות הקוד יתקמפל ללא כל שגיאה וירוץ ללא תקלות.
- ב. **שגיאת זמן ריצה** הקוד יתקמפל ללא שגיאות, אולם עלול לגרום לשגיאה בזמן ריצתו (כלומר הפסקה מוקדמת של התוכנית ללא הגעה לסוף הפונקציה main).
 - ג. שגיאת קומפילציה הקוד לא יעבור קומפילציה.

```
1.
    double a[8] = {0,1,2,3,4,5};
    double *p = a + 5;
    a = p;
```

```
א. ללא שגיאות
```

- ב. שגיאת זמן ריצה
- ג. שגיאת קומפילציה

:הסבר

```
void f(int i) {
    i = i - 1;
}
int r(int i) {
    if(i == 0) return 1;
    f(i);
    return r(i);
}
int main() {
    r(10);
    return 0;
}
```

א. ללא שגיאות

- ב. שגיאת זמן ריצה
- ג. שגיאת קומפילציה

:הסבר

```
3. float *p, x;
p = &x;
x = *p = 4.5;
```

```
א. ללא שגיאות
```

- ב. שגיאת זמן ריצה
- ג. שגיאת קומפילציה

:הסבר



הפקולטה למדעי המחשב סמסטר חורף תשס"ח (2007)

הטכניון, מכון טכנולוגי לישראל מבוא למדעי המחשב מ'/ח'

l	



<u>סעיף ב</u>

נתונה הפונקציה הבאה:

```
#define N 4
int foo(int arr[N][N], int q, int w)
  int s = q*w;
  int i, j, k, l;
  if((N % q != 0) || (N % w != 0)) {
    return 0;
  s = (s+1)*s/2;
  for (i=0; i<=N-q; i+=q) {
   for (j=0; j<=N-w; j+=w) {
      int count = 0;
      for (k=0; k<q; k++) {
        for (1=0; 1<w; 1++) {
          count += arr[i+k][j+l];
      }
      if (count != s) {
       return 0;
      }
 return 1;
}
```

:w, q, N מהי סיבוכיות הפונקציה? ניתן לבטא באמצעות הערכים

Θ()	סיבוכיות מקום נוסף:	Θ()	:סיבוכיות זמן
	. 10.12 = 11.12 = 11.01 = 0		.,

l -	
l 	
-	

:נתון המערך



2. מה יהיה ערכו של \mathbf{c} בתום הקריאה הבאה?

3. מה יהיה ערכו של c בתום הקריאה הבאה?

c = foo(arr,1,4) + foo(arr,4,1) + foo(arr,1,2);

:ערכו של **c** יהיה



l	
· · · · · · · · · · · · · · · · · · ·	



<u>שאלה 2 (25 נקודות)</u>

נתונים שני מערכים: מערך a שאורכו na ומערך b שאורכו nb. איברי שני המערכים הם מטיפוס int. **אנו נניח** שנים שני מערכים: מערך b שאורכו b ש- na≤nb (כלומר המערך b ארוך יותר) ושכל מערך מכיל איברים שונים זה מזה ללא חזרות.

בשאלה זו (בשני הסעיפים) מותר להשתמש בכל פונקציה או אלגוריתם שנלמדו בכיתה. במידה ובחרתם להשתמש בפונקציות כאלה, ציינו בטבלה הבאה את החתימות שלהן, ואת סיבוכיות הזמן והמקום שלהן.

סיבוכיות מקום נוסף	סיבוכיות זמן	חתימה



_
_



<u>סעיף א</u>

בסעיף זה נניח כי המערכים **ממוינים**. כתבו פונקציה שמקבלת את שני המערכים, ומחזירה את מספר האיברים שמופיעים רק באחד מן המערכים אך לא בשניהם.

לדוגמה, עבור המערכים הבאים על הפונקציה להחזיר 4, כיוון שיש ארבעה מספרים בדיוק המופיעים רק באחד משני המערכים אך לא בשניהם (המספרים מסומנים בקו):

int a[]=
$$\{1,2,3,5\}$$
;
int b[]= $\{0,2,3,5,\underline{6},\underline{7}\}$;

על הפונקציה לעמוד **בסיבוכיות זמן טובה ככל האפשר**, ובסיבוכיות מקום נוסף (O(1). פונקציה לא אופטימלית תזכה בניקוד חלקי בלבד.

שימו לב:

- פתרונות שמגדירים מערכים באורך MAX_INT (או אורך דומה) לא יתקבלו.
 - במידת הצורך, ניתן בפונקציה לשנות את תוכן המערכים.

ילימו את סיבוכיות הזמן של הפונקציה שלכם: סיבוכיות זמן: ()
<pre>int diff(int a[], int b[], int na, int nb) {</pre>



_
_



סעיף ב

בסעיף זה **אין להניח** כי המערכים ממוינים. עליכם לממש פונקציה שמקבלת את שני המערכים, ומחזירה את מספר האיברים שמופיעים בשני המערכים בו-זמנית (כלומר, את גודל קבוצת החיתוך). לדוגמה, עבור המערכים הבאים הפונקציה תחזיר את הערך 2 (המספרים שמסומנים בקו משותפים לשני המערכים):

int a[]=
$$\{\underline{1},9,\underline{5},4\}$$
;
int b[]= $\{\underline{5},10,18,\underline{1},3\}$;

על הפונקציה להיות בעלת **סיבוכיות זמן טובה ככל האפשר**. בנוסף, במידת האפשר יש לחסוך גם בזיכרון, אך בכל מקרה לא על חשבון סיבוכיות הזמן. פתרונות לא אופטימאליים יזכו בניקוד חלקי בלבד.

:שימו לב

- . כיוון ש- nb≥na, אנו נחשיב את (na) בתור **טוב יותר** מ- θ(nb) לצרכי השוואת יעילות.
 - פתרונות המגדירים מערכים באורך MAX_INT (או אורך דומה) לא יתקבלו.
 - במידת הצורך, ניתן בפונקציה לשנות את תוכן המערכים.

השלימו את סיבוכיות הפונקציה שכתבתם:

() פיבוכיות מקום נוסף: (Θ()	סיבוכיות זמן:
<pre>int intersect(int a[], int b[], int na, int nb) {</pre>		



_
_



<u>שאלה 3 (25 נקודות)</u>

נתון מערך a באורך n, המכיל מספרים שלמים אי שליליים. עליכם לממש פונקציה שממיינת את המערך באופן הבא:

- . בתחילת המערך יופיעו כל המספרים שמתחלקים ב- 3 ללא שארית.
 - .1 אחריהם יופיעו כל המספרים שמתחלקים ב- 3 עם שארית
- בסוף המערך יופיעו כל יתר המספרים (אלו שמתחלקים ב- 3 עם שארית 2).

אין חשיבות לסדר הפנימי של המספרים בתוך כל קבוצה.

על הפונקציה לעבוד בסיבוכיות זמן O(n) וסיבוכיות מקום נוסף

ניתן בשאלה זו להשתמש בפונקציה הבאה, המקבלת מצביעים לשני מספרים שלמים ומחליפה את תוכנם:

<pre>void sortby3(int a[], int n) {</pre>	
void Bores, s(inc a[1, inc n, [
- 	
- 	
	_



_
_



<u>שאלה 4 (25 נקודות)</u>

בשאלה זו נרצה לפתור את בעיית **מגדלי הוואי**. בדומה לבעיית מגדלי הנוי, בבעיה יש שלושה מגדלי הנוי, (נסמנם 1, 2 ו-3), ומספר טבעות המונחות על אחד המגדלים. ואולם כיאה להוואי, ובניגוד לבעיית מגדלי הנוי, הטבעות הן צבעוניות: מכל גודל טבעת יש שתי טבעות, האחת אדומה והשנייה לבנה, ובסה"כ יש m טבעות אדומות ו-m טבעות לבנות.

במצב ההתחלתי הטבעות ממוקמות זו מעל זו במגדל מס' 1 ומסודרות לפי גודלן (הגדולה ביותר למטה). מכל זוג טבעות באותו הגודל, זו הלבנה ממוקמת מתחת לזו האדומה. למשל, עבור m=3 המצב ההתחלתי הוא:



כללי הבעיה נותרים כמו בבעיה המקורית:

- אין להניח טבעת גדולה מעל טבעת קטנה ממנה. עם זאת ,מותר להניח שתי טבעות מאותו הגודל האחת מעל לשנייה, ללא קשר לצבע (כלומר מותר לבנה על אדומה וגם אדומה על לבנה מאותו הגודל).
 - אין להזיז יותר מטבעת אחת בו זמנית.

בשאלה זו אין הגבלה על סיבוכיות הפונקציות. לשם הזזת הטבעות, ניתן להשתמש בפונקציות הבאות:

```
void movered(int from, int to) {
  printf("move red ring from %d to %d\n", from, to);
}
```

```
void movewhite(int from, int to) {
  printf("move white ring from %d to %d\n", from, to);
}
```



l	
-	
	_
-	



<u>סעיף א</u>

כתבו פונקציה שמקבלת את m – מספר זוגות הטבעות במגדל המקור (כלומר סה"כ ישנן 2m טבעות, שתיים בכל גודל), וכן את מספר מגדל המקור ומספר מגדל היעד. הפונקציה מדפיסה את ההוראות להעברת כל בכל גודל), וכן את מספר מגדל המקור ומספר מגדל השלישי כמגדל עזר. במצב הסופי, על הטבעות הטבעות ממגדל המקור למגדל המקור – כלומר לבנה מתחת לאדומה. לדוגמה, עבור m=1, מגדל המקור 1 ומגדל היעד 3, הפונקציה תדפיס:

move red ring from 1 to 2
move white ring from 1 to 3
move red ring from 2 to 3

void	hawaii	1(int	m, :	int	from,	int	to)	{	

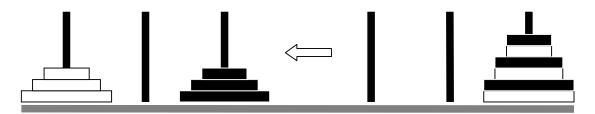


-	
	_
	_
	



סעיף ב

כתבו פונקציה שמקבלת את m – מספר זוגות הטבעות במגדל המקור, וכן את מספר מגדל המקור ומספר מגדל המקור ומספר מגדל היעד. הפונקציה מדפיסה הוראות להעברת הטבעות, כך שבסוף התהליך הטבעות מופרדות על פי צבען – כל הלבנות נמצאות במגדל היעד (הגדולה למטה) וכל האדומות נמצאות במגדל המקור (הגדולה למטה). המצב ההתחלתי והמצב הסופי נראים כך:



לדוגמה, עבור m=1, מגדל המקור 1 ומגדל היעד 3, הפונקציה תדפיס:

```
move red ring from 1 to 2
move white ring from 1 to 3
move red ring from 2 to 1
```

בסעיף זה ניתן להשתמש בפונקציה מהסעיף הקודם, גם אם לא מימשתם אותה.

<pre>void hawaii2(int m, int from, int to) {</pre>



-	
	_
	_
	





-	
	_
	_
	