

# מבוא למדעי מחשב מ' / ח' (234114 / 234117) סמסטר חורף תשע"ד

## מבחן מסכם מועד א', 20 פברואר 2014

| עח פרנוי | שח משפחה |  |  | 127 | וודו | יר ח | ากท |
|----------|----------|--|--|-----|------|------|-----|

משך המבחן: 3 שעות.

חומר עזר: אין להשתמש בכל חומר עזר (מודפס או ממוחשב).

#### 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 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.
  - בשאלה 4 בלבד, ניתן להשאיר פתרון ריק ולקבל עבורו 20% מהניקוד.

#### הנחיות תכנות כלליות, אלא אם מצוין אחרת בשאלה:

- אין להשתמש בפונקציות ספריה או בפונקציות שמומשו בכיתה, למעט פונקציות קלט/פלט והקצאת זיכרון (malloc)
  - אין להשתמש במשתנים סטטיים וגלובאליים אלא אם נדרשתם לכך מפורשות.
  - ניתן להשתמש בהקצאות זיכרון בסגנון C99 (מערכים בגודל משתנה), בכפוף לדרישות סיבוכיות זיכרון.
    - stdbool.h-ביתן להשתמש בטיפוס bool hool ניתן להשתמש בטיפוס

צוות הקורס 234114/7

מרצים: גב' אנסטסיה דוברובינה, ד"ר רן רובינשטיין, פרופ' רון קימל (מרצה אחראי).

**מתרגלים**: נמרוד סבן פרטוש, אריק יודין, נדיה ליבאי, אביב סגל, בת-חן גולדן, רן ברנשטיין, דור הריס, ברק פת, נחשון כהן (מתרגל אחראי).



| <br> |
|------|
| _    |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |



## <u>שאלה 1 (25 נקודות)</u>

בכל אחד מקטעי הקוד הבאים, חשבו את סיבוכיות הזמן והמקום של הפונקציה f כתלות ב-n.

לנוחיותכם, להלן תזכורת לחוקי לוגריתמים:

```
\log(a \cdot b) = \log(a) + \log(b)\log(a/b) = \log(a) - \log(b)\log(a^b) = b \cdot \log(a)\log_a(a^b) = b
```

א.

```
int f(int n)
{
   int k=0;
   while (n>0)
   {
      k += n;
      n--;
   }

   int res=0;
   while (k>0)
   {
      k /= 4;
      res++;
   }

   return res;
}
```

O (\_\_\_\_\_\_\_ ) סיבוכיות זמן: (\_\_\_\_\_\_ ) Θ סיבוכיות מקום:



| <br> |
|------|
| _    |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |



ב.

מבצעת חיפוש בינארי במערך ח. הפונקציה מחזירה את bin\_search() בסעיף זה, הפונקציה א האיבר במערך, או bin\_search(). האינדקס בו נמצא האיבר במערך, או 0- אם אינו נמצא. הפונקציה בעלת סיבוכיות זיכרון

```
int bin_search(int a[], int n, int x);

int f(int a[], int n)
{
   int i=1, x=1;
   while (i<n)
   {
      if (bin_search(a,i,x) >= 0)
      {
        return x;
      }
      i *= 2;
      x *= 2;
   }

   return 0;
}
```



| - |   |
|---|---|
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   | _ |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |
|   |   |



.λ

```
int aux(int n)
  int m=0;
  while (n>0)
    m += n;
    n /= 2;
  }
  return m;
}
int g(int n, int k)
  if (n < 2)
     return 1;
  int x = aux(n) + aux(k);
  return x + g(n/2, k*2);
}
int f(int n)
  return g(n,1);
}
```



| [ <del></del> |
|---------------|



## <u>שאלה 2 (25 נקודות)</u>

כתבו פונקציה המקבלת מערך של מספרים שלמים a ואת אורכו n, ומוצאת את רצף האיברים במערך שסכומו מקסימלי. הפונקציה מחזירה את סכום הרצף שנמצא, וכן כותבת את האינדקס בו מתחיל הרצף ואת שסכומו מקסימלי. הפונקציה מחזירה את סכום הרצף שמועברים לפונקציה באמצעות מצביעים. לדוגמה, עבור אורכו לזוג משתני פלט len-i id (בהתאמה) שמועברים לפונקציה באמצעות מצביעים. לדוגמה, עבור המערך הבא:

| -5 | 3 | 2 | -1 | 5 | -8 | -2 |
|----|---|---|----|---|----|----|

הרצף המקסימלי הוא הרצף (3, 2, -1, 5) שסכומו 9. הרצף מתחיל באינדקס 1 ואורכו 4. לפיכך, הפונקציה .len תחזיר 9 כערך ההחזרה, וכן תכתוב 1 למשתנה הפלט +1 ו למשתנה הפלט

 $\mathcal{O}(1)$  על הפונקציה לעמוד בסיבוכיות זמן  $\mathcal{O}(n^2)$  וסיבוכיות מקום נוסף

#### :הערות

- ייתכן יותר מרצף אחד שסכומו מקסימלי. במקרה זה יש להחזיר רצף כלשהו שסכומו מקסימלי.
- ייתכן כי הפתרון יהיה רצף באורך 0 (למשל, אם כל המספרים במערך שליליים). במקרה זה יש להחזיר את הערך 0 כסכום המקסימלי, וכן לכתוב 0 לשני משתני הפלט.

| <pre>int maxseq(int a[], int n, int* id, int* len) {</pre> |
|--|
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |





| - |  |
|---|--|
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
| [ |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |



### שאלה 3 (25 נקודות)

בממלכה רחוקה ישנן N ערים, ובכל עיר בדיוק K תושבים (כאשר N ו-K מוגדרים כ-#define). בממלכה קיימת מערכת גביית מיסים המאחסנת את הנתונים הבאים: המערך [x] מכיל את סכומי המס שמשלמת כל עיר, כאשר [i] הוא הסכום שמשלמת העיר ה-i. המערך [k] [x] מכיל את מספרי תעודת הזהות של תושבי כל עיר, כאשר השורה ה-i של המערך, [1-x] [i] מכילה את מספרי תעודת הזהות של התושבים בעיר ה-i.

בשאלה זו נרצה לכתוב פונקציה הממיינת את הערים על פי סכום המס שהן משלמות, בסדר עולה. עליכם a ו-id, וממיינת אותם כך שלאחר המיון המערך a לממש פונקציה (בעמוד הבא) המקבלת את המערכים a ו-id, וממיינת אותם כך שלאחר המיון השורה ה-ii יהיה בסדר עולה. בתהליך זה <u>יש לשמור על ההתאמה בין המערכים,</u> כלומר, גם לאחר המיון השורה ה-ii במערך id צריכה להכיל את רשימת התושבים בעיר שמשלמת מס בגובה [i].

.K. לפיכך, אדול מ-N, נרצה סיבוכיות קטנה ככל האפשר ביחס ל-K. לפיכך, דרישות סיבוכיות: כיוון שבסבירות גבוהה א גדול מ-N, גדול מ- $O(N^2 + NK)$ , זיכרון נוסף  $O(N^2 + NK)$ .

#### :הערות

- . בשאלה זו הערכים N ו-K אינם נחשבים כקבועים לצרכי סיבוכיות. ■
- יש לכתוב במקום המתאים למטה את סוג המיון בו השתמשתם, ולהסביר בקצרה מדוע הוא עומד בדרישות הסיבוכיות.
  - פתרון שאינו עומד בדרישות הסיבוכיות יקבל ניקוד חלקי בלבד.

| ַ המיון שהשתמשתי בו: |
|----------------------|
|                      |
| הסבר לסיבוכיות:      |
|                      |
|                      |
|                      |
|                      |
|                      |
|                      |





| [- <del></del> |  |
|----------------|--|
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |
|                |  |



| <pre>void sort(int a[N], int id[N][K]) {</pre> |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |





## <u>שאלה 4 (25 נקודות)</u>

בשאלה זו נממש פונקציה שמקבלת סכום כסף נתון sum, ומחזירה את מספר הצורות השונות שבהן ניתן coins[] להגיע לסכום כסף זה באמצעות צירופים של מטבעות. ערכי המטבעות הזמינים לנו נמצאים במערך שאורכו n, כאשר כל ערך של מטבע מופיע במערך פעם אחת בדיוק. ניתן להניח שהמערך [coins[] מכיל ערכים <u>שלמים, חיוביים, ושונים זה מזה</u>. שימו לב שניתן להשתמש בכל מטבע מספר לא מוגבל של פעמים על מנת להגיע לסכום sum.

:לדוגמה, עבור המערך [coins הבא

```
int coins[2] = \{1, 2\};
```

ועבור הסכום 4=sum, הפונקציה תחזיר 3 כיוון שניתן להגיע לסכום 4 באמצעות צירופי המטבעות הבאים:

```
1+1+1+1=4
1+1+2=4
2+2=4
```

<u>שימו לב</u>: בשאלה זו אין משמעות לסדר המחוברים. כלומר, הסכומים (2+1+1), (1+2+1) ו-(1+1+2) נחשבים לאותו הפירוק, ויש לספור אותם פעם אחת בלבד.

<u>יש לפתור את השאלה ברקורסיה</u>. בשאלה זו אין דרישות סיבוכיות.

| <pre>int change(int coins[], int n, int sum) {</pre> |
|--|
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |



| <br> |
|------|
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
| <br> |
| <br> |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |



|               | _           |
|---------------|-------------|
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
| - <del></del> | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
| <del></del>   | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               |             |
|               |             |
|               | _           |
|               | _           |
|               | _           |
|               | _           |
|               | _           |
|               | _           |
|               | _           |
|               | _<br>_<br>_ |
|               | _<br>_<br>_ |
|               | _<br>_<br>_ |
|               | _           |
|               |             |
|               |             |
|               |             |
|               |             |
|               |             |
|               |             |
|               |             |
|               |             |
|               |             |
|               |             |



| <br> |
|------|
| <br> |
|      |
|      |
|      |
| <br> |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |
|      |



| <del></del> |
|-------------|
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |
|             |



| - |  |
|---|--|
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |
|   |  |