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

סמסטר אביב תשע"ג

מבחן מסכם מועד ב', 2 באוקטובר 2013

מספר סטודנט: | | | | | | | | | *רשום/ה לקורס:* | 1 | 1 | 4 | 3 | 2 |

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.
- 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.

- אין להשתמש בפונקציות ספריה או בפונקציות שמומשו בכיתה, למעט פונקציות קלט/פלט והקצאת זיכרון (`malloc`). 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.
- נוסחאות שימושיות:

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

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

צוות הקורס 234114/7
מרצים: פרופ' ניר אילון (מרצה אחראי), ד"ר תמיר לוי, חביאר טורק.
מתרגלים: נחשון כהן, נדיה לבאי, עלי חמוד, אביב סגל, ירון קסנר, ורד כהן, אנסטסיה דוברובינה (מתרגלת אחראית).

בהצלחה!

שאלה 1 (25 נקודות):

א. (12 נקודות) חשבו את סיבוכיות הזמן והמקום של הפונקציה f המוגדרת בקטע הקוד הבא, כפונקציה של n. אין צורך לפרט שיקוליכם.

```
int f(int n)
{
    int x = 1, q = 2;

    while (x < n) {
        x *= q;
        q *= 4;
    }

    return x;
}
```

סיבוכיות זמן: $\Theta(\sqrt{log(n)})$ סיבוכיות מקום: $\Theta(1)$

ב. (13 נקודות) חשבו את סיבוכיות הזמן והמקום של הפונקציה f המוגדרת בקטע הקוד הבא, כפונקציה של n. אין צורך לפרט שיקוליכם.

```
int g(int n)
{
    if (n == 0)
        return n;
    return n + g(n/2);
}

int f(int n)
{
    int x = n;
    while (x < n * n)
        x = g(x);
    return x;
}
```

סיבוכיות זמן: $\Theta(log^2(n))$ סיבוכיות מקום: $\Theta(log(n))$

שאלה 2 (25 נקודות):

עליכם לממש את הפונקציה:

```
void intersect(char s1[], char s2[], char out[])
```

הפונקציה מקבלת כקלט שתי מחרוזות s1 ו-s2, שמכילות רק אותיות לטיניות. לדוגמא:

s1	z	Z	a	B	a	z	h	h	T	\0

s2	A	b	c	c	T	q	Q	m	z	z	\0

על הפונקציה להחזיר בתוך מערך תווים out את מחרוזת ה**חיתוך** של שתי המחרוזות, **כולל** חזרות ו**ללא** חשיבות לגודל האותיות. כלומר, את כל התווים המופיעים גם במחרוזת הראשונה וגם במחרוזת השנייה ובלי חשיבות לגודלם (לדוג': אותיות A ו a נחשבות זהות). בנוסף, אם אות מופיעה עם חזרות בשתי המחרוזות, היא תופיע יותר מפעם אחת ב-out. על מחרוזת המוחזרת להיות ממוינת בסדר עולה <u>ורק עם אותיות קטנות</u>. למשל, בדוגמא לעיל, הפונקציה תחזיר את המחרוזת הבאה:

out	a	b	t	z	z	\0

האות 'z' מופיעה פעמיים ב-out כי נמצאת פעמיים גם ב-s1 וב-s2.

דרישות: סיבוכיות זמן O(|s1| + |s2|) וסיבוכיות מקום נוסף O(1), כאשר |s1| הוא האורך של מחרוזת s1 ו-|s2| הוא האורך של מחרוזת s2.

דגשים:

1. אין לשנות את תוכן המחרוזות s1 ו-s2, אפילו לא באופן זמני.
2. יש להניח שהמחרוזות s1 ו-s2 מורכבות מאותיות לטיניות קטנות וגדולות בלבד.
3. יש להניח כי למערך out מספיק מקום לאותיות בחיתוך.

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

```
#define LEN ('z'-'a'+1)
char toLower(char c) {
    return (c>='A' && c<='Z')? c-'A'+'a' : c;
}
int min(int x, int y) {
    return (x>y)? y : x;
}
```

```
void intersect(char s1[], char s2[], char out[]){
    int hist1[LEN]={0}, hist2[LEN]={0};
    while(s1[0]!='\0'){
        hist1[toLower(s1[0])-'a']++;
        s1++;
    }
    while(s2[0]!='\0'){
        hist2[toLower(s2[0])-'a']++;
        s2++;
    }
    for(int i=0; i<LEN; ++i){
        int num = min(hist1[i], hist2[i]);
        for(int j=0; j<num; ++j){
            out[0]='a'+i;
            out++;
        }
    }
    out[0]='\0';
}
```

שאלה 3 (25 נקודות) :

מערך a של מספרים שלמים וחיוביים נקרא מערך ממוין "**משותף-m**" אם:

- לכל האיברים במערך מתקיים המחלק המשותף הגדול ביותר בין a[i] ו- **m**, קטן או שווה מהמחלק המשותף הגדול ביותר בין a[i+1] ו- **m**.

ממשו את הפונקציה הבאה, שחתימתה

```
int m_search(int a[], int n, int m, int x)
```

הפונקציה מקבלת כקלט מערך a ממוין "**משותף-m**" עם פרמטר m ובאורכו n, ועוד מספר חיובי x. על הפונקציה להחזיר את המיקום של איבר כלשהו במערך a אשר המחלק המשותף הגדול ביותר בינו ו-m שווה ל-x או 1- במידה ואין איבר כזה במערך.

לדוגמא, אם a הוא המערך הבא:

```
int a[7] = {1, 11, 9, 15, 6, 42, 24};
```

הוא ממוין "משותף-m" עם n=7, m=12, אז:

- עבור x=4, הפונקציה תחזיר 1- כי אין איבר ב-a אשר המחלק המשותף הגדול ביותר עם 12 אשר שווה ל-4.
- עבור x=6, הפונקציה תחזיר 4 (או 5) כי המחלק המשותף הגדול ביותר בין 6 ו-12 שווה 6.

דרישות: סיבוכיות זמן O(log(m)×log(n)) וסיבוכיות מקום נוסף O(1)

```
int m_search(int a[], int n, int m, int x){
    int low = 0, high = n - 1, mid;
    while (low <= high) {
        mid = (high + low) / 2;
        if (gcd(a[mid], m) < x)
            low = mid + 1;
        else if (gcd(a[mid], m) > x)
            high = mid - 1;
        else
            return mid;
    }
    return -1;
}
```

```
int gcd(int a, int b) {
    int tmp;
    while (b != 0) {
        tmp = b;
        b = a % b;
        a = tmp;
    }
    return a;
}
```

שאלה 4 (25 נקודות) :

עליכם לפתור את בעיית התכנון הפיננסי הבאה של איש עסקים בעל תיק השקעות הכולל חובות ונכסים. תשלומי החובות פרוסים על פני n חודשים, כך שבעוד j חודשים יהיה עליו לשלם סכום של debt[j]. מספר הנכסים הוא k. ערכי הנכסים משתנים לאורך זמן לפי הנוסחה הבאה: הערך של הנכס ה- i בעוד j חודשים הוא asset[i] + j*interest[i]. לדוגמא, אם asset[i]==5 ו- interest[i]==2, אז הערך של הנכס ה- i היום הוא 5, בעוד חודש הוא 7, בעוד חודשיים 9 וכן הלאה.

קופת המזומנים של איש העסקים ריקה היום. בכל חודש מותר לו למכור **לכל היותר** נכס אחד על מנת להזרים מזומנים לקופה לצורך תשלום החוב באותו החודש. עליכם לכתוב פונקציה

```
int saves(int debt[], int n, int asset[],
          int interest[], int k)
```

הפונקציה תחזיר את מספר הנכסים **המקסימלי** שהאיש יכול להציל לאחר n חודשים, בהנחה שהוא מספק את תשלומי החובות מידי חודש בחודשו. אם לא קיימת אף אפשרות לשלם את כל החובות, על הפונקציה להחזיר 0.

לדוגמא, עבור המערכים הבאים (n=5, k=4):

חודש i	0	1	2	3	4
debt[i]	1	1	1	1	99

נכס j	0	1	2	3
asset[j]	20	1	5	2
interest[j]	25	0	0	1

אם האיש מוכר את הנכס 0 בחודש 0, הוא לא יצליח לשלם את חובותיו בחודש 4. אם הוא מוכר את הנכס 1 בחודש 0, ומוכר את הנכס 3 בחודש 1, ומוכר את הנכס 0 בחודש 4, הוא יצליח להציל רק נכס אחד (נכס 2). אם הוא מוכר את הנכס 2 בחודש 0, ומוכר את הנכס 0 בחודש 4, הוא יצליח להציל 2 נכסים.

הערות:

- החודש הראשון הוא חודש 0, וקופת המזומנים מתחילה ב-0.
- 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.

```
int max(x,y);

int saves_aux(int debt[], int n, int asset[], int interest[], int k,
              int sold[], int month, int assets_saved, int cash)
{
    // pruning
    if (cash < 0)
        return 0;
    // stop condition
    if (month == n)
        return assets_saved;
    }
    // pay debt
    cash -= debt[month];
    // try not selling any asset
    int best_score = saves_aux(debt,n,asset,interest,k,
                               sold, month+1,assets_saved,cash);
    for (int i=0; i<k; ++i)
    {
        // try selling asset i
        //  mark asset as sold
        sold[i] = 1;
        cash_after_sale = cash+asset[i]+month*interest[i];
        score = saves_aux(debt,n,asset,interest,k,
                          sold, month+1,assets_saved-1,
                          cash_after_sale));
        best_score = max(best_score,score);
        //  roll back sold assets
        sold[i] = 0;
    }
    return best_score;
}
```

```
int saves(int debt[], int n, int asset[],
          int interest[], int k)
{
    int sold[k] = {0};
    return saves_aux(debt,n,asset,interest,k,
                     sold,0,k,0);
}

int max(x,y)
{
    return (x>y)?x:y;
}
```