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

סמסטר קיץ תשע"ה

מבחן מסכם מועד ב', 20 אוקטובר 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 19 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: $1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n}=\Theta(\log n)$ $\quad 1+\frac{1}{4}+\frac{1}{9}+\frac{1}{16}+\frac{1}{25}+...=\Theta(1)$

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

צוות הקורס 234114/7
מרצים: ד"ר רוני קופרשטוק
מתרגלים: גב' גילי יבנה, מר סער זהבי

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.

```
int f(int n)
{
   int x=0;
   for(int i=0; i<=n; ++i)
        for(int j=i+1; j<=n; j++){
           for(int k=1; k<j+1; k++)
                 x++;
           for(int m=k; m<n; m++)
                 x++;
        }
   return x;
}
```

סיבוכיות זמן (6 נקג): $\Theta(n^3)$ סיבוכיות מקום (2 נקג): $\Theta(1)$

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

```
void f(int n){
    for(int i=1; i<=n; ++i){
      double x=i;
      double delta = 1/(double)i;
      while( x>-x ){
        char* m = malloc( x/delta );
        x -= delta;
        free( m );
      }
    }
    return;
}
```

סיבוכיות זמן: (6 נקג) $\Theta(n^3)$ סיבוכיות מקום (3 נקג): $\Theta(n^2)$

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

```
void f(int n){
    if(n<1)
        return;
    int a=1;
    for (int i=2; i<=n; ++i) a*=i;
    char* m = malloc( n/5 );
    f(n/5); f(n/5); f(n/5); f(n/5); f(n/5);
    free( m );
}
```

סיבוכיות זמן (4 נקג): $\Theta(n\log n)$ סיבוכיות מקום (4 נקג): $\Theta(n)$

שאלה 2 (25 נקודות):

מחרוזת פלינדרום היא מחרוזת שאם קוראים אותה מההתחלה לסוף או מהסוף להתחלה מקבלים את אותו רצף האותיות, למשל "סוס" או "ילד כותב בתוך דלי".

עליכם לממש את הפונקציה שחתימתה:

```
bool is_palindrome(char* s);
```

הפונקציה מקבלת מחרוזת המורכבת מאותיות באנגלית ורווחים ומחזירה true אם היא פלינדרום ו- false אם לא. על הפונקציה להתעלם מגודל האותיות (להיות case insensitive).
דוגמא:
על המחרוזת "A b Cc B A" הפונקציה תחזיר true.
על המחרוזות "ABCA" ו- "E FFE" הפונקציה תחזיר false.
דרישות:
אין להשתמש בלולאות (כלומר אין להשתמש במילים for,while).
ניתן להשתמש בפונקציות עזר. אסור לשנות את תוכן מחרוזת הקלט.
סיבוכיות זמן O(n) וסיבוכיות מקום נוסף O(n), כאשר n הוא אורך המחרוזת.
אין צורך לבדוק את תקינות הקלט.
רמז: רקורסיה.

```
bool is_palindrome(char* s)
{
    int len = strlen_rec(s);
    return is_palindrome_rec(s, len);
}

bool is_palindrome_rec(char* s, int len)
{
    if (len <=1)
        return true;
    if (to_lower(*s) == to_lower(*(s+len-1)))
        return is_palindrome_rec(s+1, len-2);
    return false;
}
```

```
int strlen_rec(char* s)
{
    if (!*s) return 0;
    return 1 + strlen_rec(s+1);
}

char to_lower(char c)
{
    if (c > 'A' && c < 'Z')
        return c-'A'+'a';
    return c;
}
```

שאלה 3 (25 נקודות):

נתונה מטריצה ריבועית כך שכל ערך בעמודה הראשונה ובשורה הראשונה גדול מכל ערך בתת-מטריצה המתחילה ב- 1,1.
באופן דומה כל ערך בעמודה הראשונה ובשורה הראשונה של תת המטריצה המתחילה ב-1,1 גדול מכל ערך בתת-מטריצה המתחילה ב- 2,2, וכך הלאה עד תת המטריצה n,n.
בשאלה זו יש להניח ש n מוגדר ב #define.

300	65	200	60
80	9	25	20
26	10	6	5
27	8	2	1

עליכם לממש את ה הפונקציה שחתימתה:

```
bool inMatrix(int a[n][n], int x);
```

הפונקציה תחזיר true אם הערך x נמצא במטריצה a ו-false אחרת.
סיבוכיות זמן: O(n)
סיבוכיות מקום נוסף: O(1).

```
bool inMatrix(int a[n][n], int x)
{
    for (int i=0; i<n-1; i++)
    {
        if (a[i][i] > x && a[i+1][i+1] > x)
        {
            continue;
        }
        else
        {
            return (inRowOrCol(a,x,i) || inRowOrCol(a,x,i+1));
        }
    }
    return false;
}
```

```
bool inRowOrCol(int a[n][n], int x, int ind)
{
    for (int i=ind; i<n; i++)
    {
        if ((x==a[ind][i]) || (x==a[i][ind]))
            return true;
    }
    return false;
}
```

שאלה 4 (25 נקודות) :

מזל טוב! התקבלתם לעבודה כמתכנתים ומפתחי אלגוריתמים בחברת "זיל וזול", רשת סופרמרקטים בפרישה ארצית.
עליכם לכתוב פונקציה שמציעה ללקוח סל קניות מקוון באפליקציית אייפון, לפי הכללים הבאים:

- הלקוח מכניס שלושה מספרים protein, carb, budget לתוך טופס מקוון. המספר protein מייצג את כמות החלבון המינימלית שהוא מבקש לרכוש. המספר carb מייצג את כמות הפחמימות המינימלית. מהמספר budget מייצג את המקסימום שהוא מוכן לשלם. (ניתן להניח שהמספרים שלמים לצורך התרגיל.)
- יש n מוצרים ברשת "זיל וזול", כאשר המוצר ה- i מכיל p[i] חלבונים ו- c[i] פחמימות. מחיר המוצר ה- i הוא price[i]. מותר ללקוח להכניס לכל היותר 3 פעמים את המוצר ה- i לסל, לכל i.
- על האפליקצייה להציע סל **כלשהו** חוקי שמקיים את הדרישות, ולהחזיר את מחיר סל זה . אם לא קיים סל כזה, עם האפליקציה להחזיר 1-.
-

עליכם לכתוב פונקציה שפותרת בעייה זו, עם החתימה הבאה:

```
int zil_zol_shop(int n, int p[], int c[], int price[],
                 int protein, int carb, int budget,
                 int cart[])
```

המערך cart משמש כפלט, כאשר cart[i] הוא מספר הפעמים שיש להכניס את המוצר ה- i לסל. ערך החזרה של הפונקציה הוא המחיר הכולל של הסל, או 1- אם לא קיים סל חוקי.
באופן מתמטי, עליכם למצוא סדרת מספרים לא שליליים cart[0]...cart[n-1] כך ש

- הסכום של p[i] * cart[i] (כאשר i רץ מ 0 עד n-1) הוא לפחות protein.
- הסכום של c[i] * cart[i] (כאשר i רץ מ 0 עד n-1) הוא לפחות carb.
- 0<=cart[i]<=3 (לכל i מ 0 עד n-1)
- הסכום price[i] * cart[i] (כאשר i רץ מ 0 עד n-1) הוא לכל היותר budget. סכום זה מוחזר ע"י הפונקציה אם קיים פיתרון, אחרת מוחזר 1-.

דוגמא: protein=20, carbs=19, budget=53
n=2, p={6,2}, c={3,11}, price={10,20}
אז פיתרון אפשרי הוא cart={3,1}. מחיר פיתרון זה הוא 50.
אם נשנה את carbs ל 22 (ונשאיר את כל שאר הנתונים ללא שינוי) אז לא יהיה פיתרון, מכיוון שאז נהייה חייבים להכניס את המוצר השני לסל פעמיים כדי לספק את carbs, בעלות של 2*20=40. אבל אז ניוותר עם תקציב של 13 שיספיק למוצר ראשון בודד, ולא נוכל לספק את דרישות ה protein.
יש להשתמש ב backtracking ולגזום ענפים כמה שניתן.

```
int zil_zol_shop(int n, int p[], int c[], int price[],
            int protein, int carb, int budget, int cart[]) {
    if (protein <=0 && carb <=0 && budget >=0)
        return 0;
    if (n==0)
        return -1;
    for (int i=0; i<=3; i++)
    {
        cart[0] = i;
        int res= zil_zol_shop(n-1, p+1, c+1, price+1,
            protein-i*p[0], carb-i*c[0], budget-i*price[0], cart+1);
        if (res > -1)
            return res + i*price[0];
    }
    return -1;
}
```