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

סמסטר אביב תשע"ו

מבחן מסכם מועד ב', 14 ספטמבר 2016

2	3	4	1	1	

רשום/ה לקורס:

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.
- וודאו שאתם נבחנים בקורס המתאים.
- 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. <u>ניתן בהחלט להשתמש בעיפרון ומחק</u>, פרט לדף השער אותו יש למלא בעט.
- חובה לקרוא הוראות לכתיבת קוד המופיעות בעמוד הבא לפני פתרון המבחן.
- 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.
- נוהל "**לא יודע**": אם תכתבו **בצורה ברורה** "לא יודע/ת" על שאלה (או סעיף) שבה אתם נדרשים לקודד, תקבלו 20% מהניקוד. דבר זה מומלץ אם אתם יודעים שאתם לא יודעים את התשובה.
- 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"מ מירלה בן חן (מרצה אחראית), יעל ארז, איהאב ואתד

הנחיות לכתיבת קוד במבחן

- בכל השאלות, הנכם רשאים להגדיר ולממש פונקציות עזר כרצונכם. לנוחיותכם, אין חשיבות לסדר מימוש הפונקציות בשאלה, ובפרט ניתן לממש פונקציה לאחר השימוש בה (בלי צורך להצהיר על הפונקציה לפני). מותר להשתמש בפונקציה שנכתבה בסעיף אחר, בתנאי שתציינו באופן ברור איפה הפונקציה ממומשת.
- **חובה** להקפיד על תכנות מבני (כלומר, חלוקה נכונה לפונקציות).
- אלא אם כן נאמר אחרת בשאלות, **אין להשתמש בפונקציות ספריה או בפונקציות שמומשו בכיתה**, למעט פונקציות קלט/פלט והקצאת זיכרון (`malloc, free`) והפונקציה memcpy. ניתן להשתמש בטיפוס `bool` המוגדר ב-`stdbool.h`.
- חתימת הפונקציה memcpy: void memcpy(void *dest, void *src, unsigned size). שימו לב ש size הוא מספר **הבתים** שצריך להעתיק.
- אין להשתמש במשתנים סטטיים וגלובאליים אלא אם נדרשתם לכך מפורשות.
- כאשר נדרש לבצע מיון בסיבוכיות O(nlogn), ניתן לממש מיון מהיר (quicksort).

בהצלחה!

שאלה 1 (25 נקודות):

א. (5 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){
      int a[4] = {1, 1, 1, 1};
      for (int k = 0; k < n; k++) {
            a[k%4] *= 3;
      }

      for (int j = 0; j < a[0]; j++) {
            for(int k = 0; k < j; k++) {
                  printf("*");
            }
      }
}
```

סיבוכיות זמן: $\Theta(3^{\frac{n}{2}})$ סיבוכיות מקום: $\Theta(1)$

ב. (10 נקודות): חשבו את סיבוכיות הזמן והמקום של הפונקציה f (ולא של g) כפונקציה של n.

```
void g(int n){
      if (n < 1)
            return;

      g(n/2);
}

int f(int n){
      int x = 1;
      for (int k = 0; k < n; k++) {
            x *= n * n;
      }

      g(n);
      return x;
}
```

סיבוכיות זמן f : $\Theta(n)$ סיבוכיות מקום f : $\Theta(\log n)$

ג. (10 נקודות):חשבו את סיבוכיות הזמן והמקום של הפונקציה f, כפונקציה של n.

```
int g(int n){
      if (n < 1)
            return 0;
      return n + g(n-1);
}

int f(int n){
      int x = 1;
      while (x*x < g(n)) {
            x++;
      }
      return x;
}
```

סיבוכיות זמן: קיבלנו את שתי התשובות: $\Theta(n^2)$ $\Theta(n)$ סיבוכיות מקום: $\Theta(n)$

שאלה 2 (25 נקודות) :

ממשו את הפונקציה הבאה, שחתימתה

```
int findDistance(int points[][2], int n, double d);
```

הפונקציה מקבלת כקלט מערך `points` של n שורות (כל שורה היא נקודה המכילה קואורדינטת x וקואורדינטת y), ומספר נוסף d. המערך points ממויין ע"י המרחק מנקודה קבועה (X, Y), כלומר, לכל i בין 0 ל n-2 מתקיים שהמרחק בין arr[i] לנקודה (X,Y) קטן או שווה מהמרחק בין arr[i+1] לנקודה (X,Y).

$$\sqrt{(arr[i][0]-X)^2+(arr[i][1]-Y)^2} \le \sqrt{(arr[i+1][0]-X)^2+(arr[i+1][1]-Y)^2}$$

על הפונקציה להחזיר את האיבר האחרון שהמרחק שלו מ (X,Y) קטן ממש מ d. אם לא קיים אף איבר כזה הפונקציה צריכה להחזיר 1-. בניסוח מתמטי, מחפשים את ה i הכי גדול שמקיים:

$$\sqrt{(arr[i][0]-X)^2+(arr[i][1]-Y)^2} < d$$

לדוגמה: עבור הנקודה (0,0), והמערך הבא

3	0
3	-1
3	3
5	0
7	2

עבור $d=\sqrt{11}$, הפונקציה תחזיר 1, כי המרחק של (3,-1) מראשית הצירים הוא $\sqrt{10}$, אבל המרחק של (3,3) מראשית הצירים הוא $\sqrt{18}$.

עבור $d=\sqrt{26}$, הפונקציה תחזיר 3, כי המרחק של (5,0) מראשית הצירים הוא $\sqrt{25}$, אבל המרחק של (7,2) מראשית הצירים הוא $\sqrt{53}$.

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

הערות:

- ניתן להניח שכל האיברים במערך שונים זה מזה.
- אם קיימת נקודה במערך שמרחקה מ (X,Y) שווה בדיוק ל d אין להחזיר אותה. אפשר להתעלם מחוסר דיוק של double לצורך חישוב זה.
- ניתן להניח ש X, Y מוגדרים ב define.
- אסור להשתמש בפונקציה המחשבת שורש. עם זאת, במקרה שהשתמשתם בפונקציה כזו תאבדו רק 5 נקודות מהשאלה.

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

```
#define X 0
#define Y 0

int square(int x) {
  return x*x;
}

int squareDist(int points[][2], int i) {
      return square(points[i][0] - X)+square(points[i][1] - Y);
}

int findDistance(int points[][2], int n, double d) {
  if (n <= 0)
    return -1;
  if (n == 1)
    return squareDist(points, 0) < d*d ? 0 : -1;
  if (squareDist(points, n - 1) < d*d)
    return n - 1;
  if (squareDist(points, 0) >= d*d)
    return -1;

  int l = 0;
  int r = n-1;

  while (l < r) {
    int m = (l + r) / 2;
    int dist1 = squareDist(points, m);
    int dist2 = squareDist(points, m + 1);

    if (dist1 < d*d && dist2 >= d*d) {
      return m;
    }
    else if (dist1 < d*d) {
      l = m + 1;
    }
    else {
      r = m;
    }
  }

  return -1;
}
```

שאלה 3 (25 נקודות) : מחרוזות

פלינדרום הוא רצף תווים הנקרא באופן זהה משמאל לימין ומימין לשמאל.
למשל "amoroma", "racecar" הם פלינדרומים.

בשאלה זו נעבוד אך ורק עם פלינדרומים שבהם מספר **אי זוגי** של תווים ונניח שהמחרוזת מכילה אותיות אנגליות קטנות בלבד.

ממשו את הפונקציה הבאה, שחתימתה

```
int longest_pal(char *str);
```

הפונקציה מקבלת מחרוזת str ומחזירה את אורכו של הפלינדרום הארוך ביותר במחרוזת.
למשל במחרוזת "racecar" קיים פלינדרום באורך 7 : "racecar".
במחרוזת "here" קיים פלינדרום באורך 3 : "ere".
במחרוזת "stam" הפלינדרום המקסימלי אורכו 1.
אם המחרוזת ריקה הפונקציה תחזיר 0.

ניתן להניח שלא קיים במחרוזת פלינדרום עם מספר תווים זוגי.

דרישות סיבוכיות: עבור מחרוזת באורך n, סיבוכיות זמן $O(n^2)$, סיבוכיות מקום נוסף $O(1)$

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

```
int longest_pal(char *str){
      int longest = 0;
      for (int i = 0; *(str + i); i++){
            char *middle = (str + i);
            for (int j = 0; (j <= i) && *(middle + j) &&
                  (*(middle + j) == *(middle - j)); j++){
                  longest =
                  longest > 2 * j + 1 ? longest : 2 * j + 1;
            }
      }
      return longest;
}
```

שאלה 4 (25 נקודות) :

נתונים 2 ארגזים, ו- N סוגים של פריטים לאחסון בארגזים.

נתון כי:

- כל ארגז יכול להכיל חפצים שסך משקלם לכל היותר MAX_WEIGHT.
- חלק מהפריטים לא יכולים להיות מאוחסנים באותו ארגז.

ניתן להניח ש N ו- MAX_WEIGHT מוגדרים ב #define.

כתבו פונקציה:

```
int insert(int items[N][2], int rules[N][N]);
```

אשר מקבלת שני מערכים:

- מערך דו-מימדי items[N][2] שמתאר את המשקל של כל פריט וכמה פריטים כאלו קיימים. המשקל של פריט מספר i הוא items[i][0] ומספר הפריטים מסוג זה הוא items[i][1].
למשל עבור המערך:

1	5
2	4
5	20
2	1
3	2

משקלו של פריט מסוג 0 הוא 5 וקיימים 1 פריטים כאלו,
משקלו של פריט מסוג 2 הוא 20 וקיימים 5 פריטים כאלו.

- מערך דו-מימדי rules[N][N] שמתאר לכל שני פריטים אם ניתן לאחסן אותם באותו ארגז. אם rules[i][j] שווה אחד אז אפשר לאחסן את הפריטים i ו- j באותו ארגז אחרת אסור לאחסנם ביחד באותו ארגז. ניתן להניח שאיברי האלכסון במערך rules הם תמיד 1.
למשל עבור המערך:

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

אסור לאחסן פריט מסוג 0 עם פריט מסוג 3, כי M[0][3]==0.

הפונקציה תחזיר את המשקל המקסימלי של פריטים אשר אפשר לאחסן בשני הארגזים יחד.

לדוגמה עבור 5 פריטים, משקל מקסימלי 10 והמערכים שלעיל, הפונקציה תחזיר 19 :

ארגז ראשון יכיל: פריט אחד מסוג 1 במשקל 4 ועוד שלושה פריטים מסוג 4 במשקל 2. סך משקל 10.

ארגז שני יכיל: פריט אחד מסוג 0 במשקל 5 ופריט אחד מסוג 1 במשקל 4. סך משקל 9.

משקל כללי מקסימלי 19.

הערות:

- יש להשתמש בשיטת backtracking כפי שנלמדה בכיתה.
- אסור לשנות את תוכן המערך rules.
- **מותר** להשתמש בלולאות.
- בשאלה זו אין דרישות סיבוכיות, אולם כמקובל ב – backtracking יש לוודא שלא מתבצעות קריאות רקורסיביות מיותרות עם פתרונות שאינם חוקיים.
- מותר להשתמש בפונקציות עזר.

```
int insert(int items[N][2], int rules[N][N]) {
	int used[N][2] = { { 0, 0 } };
	int w[2] = { 0 };
	return insert_aux(items, rules, used, w, 0);
}

int insert_aux(int items[N][2], int rules[N][N], int used[N][2], int w[2], int index) {
	if (index == N) {
		return w[0] + w[1];
	}

	//dont take current element
	int maxWeight = insert_aux(items, rules, used, w, index + 1);
	for (int box = 0; box < 2; box++) {
		//try to add to 'box'
		if (can_add(items, rules, used[box], w[box], index)) {
			add(items, used[box], index, w, box);
			int current = insert_aux(items, rules, used, w, index);
			maxWeight = max(current, maxWeight);
			remove(items, used[box], index, w, box);
		}
	}

	return maxWeight;
}
```

```
int can_add(int items[N][2], int rules[N][N], int used[N], int w, int index) {
	if (w + items[index][0] > MAX_WEIGHT || items[index][1] == 0) {
		return false;
	}

	for (int i = 0; i < N; i++) {
		if (rules[index][i] == 0 && used[i] > 0) {
			return false;
		}
	}

	return true;
}

void add(int items[N][2], int used[N], int i, int w[2], int box) {
	w[box] += items[i][0];
	items[i][1]--;
	used[i]++;
}

void remove(int items[N][2], int used[N], int i, int w[2], int box) {
	items[i][1]++;
	used[i]--;
	w[box] -= items[i][0];
}

int max(int a, int b) {
	return a > b ? a : b;
}
```