מבוא למדעי המחשב מ'/ח' (234114 \ 234117)

סמסטר חורף תשע"ז

מבחן מסכם מועד א', 21 לפברואר 2017

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 22 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.
- 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' תומר שלומי (מרצה אחראי), מר איהאב וואתד, גב' יעל ארז **מתרגלים:** גב' דניאל עזוז, גב' צופית פידלמן, מר תומר לנגה, מר יובל בנאי, מר יואב נחשון, מר יורי פלדמן, מר דמיטרי רבינוביץ', מר שי וקנין, מר איתי הנדלר, מר יותם אשל

בהצלחה!

שאלה 1 (25 נקודות):

א. (8 נקודות) חשבו את סיבוכיות הזמן והמקום של הפונקציה f1 המוגדרת בקטע הקוד הבא, כפונקציה של n. אין צורך לפרט שיקוליכם. חובה לפשט את הביטוי ככל שניתן.
הניחו שסיבוכיות הזמן של malloc(n) היא $\Theta(1)$, וסיבוכיות המקום של malloc(n) היא $\Theta(n)$.

```
int f1(int n){
    int q;
    for(q=1; q<n*n; q*=2);
    int *vals = malloc(sizeof(int)*q);
    for(int i=1; i<q; i*=3){
        vals[i]=i/3;
        printf("!");
    }
    q = vals[0];
    return q;
}
```

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

ב. (9 נקודות): חשבו את סיבוכיות הזמן והמקום של הפונקציה f2 המוגדרת בקטע הקוד הבא, כפונקציה של n. אין צורך לפרט שיקוליכם. חובה לפשט את הביטוי ככל שניתן.

```
void g2(int n) {
    while (n>0) {
        printf("*");
        n--;
    }
}

void f2(int n){
    int k = n/2;
    for (int i=2; i<n; i*=i) {
        g2(k);
    }
}
```

סיבוכיות זמן: $\Theta(n*(\log(\log n)))$ סיבוכיות מקום: $\Theta(1)$

ג. (8 נקודות): חשבו את סיבוכיות הזמן והמקום של הפונקציה f3 המוגדרת בקטע הקוד הבא, כפונקציה של n. אין צורך לפרט שיקוליכם. חובה לפשט את הביטוי ככל שניתן.

```
int g3(int n) {
    if (n<=1)
        return n;
    return g3(n-1)+2*n-1;
}

void f3(int n){
    int temp = g3(n);
    for (int i=0; i<temp; i++)
        printf("%d\n", g3(i));
}
```

סיבוכיות זמן: $\Theta(n^4)$ סיבוכיות מקום: $\Theta(n^2)$

שאלה 2 (25 נק')

בהינתן מערך של n ציונים בין 0 ל-100, המרחק היחסי בין שני ציונים g_1, g_2 מוגדר באופן הבא:

$$\text{relative\_distance}(g_1, g_2) = \frac{n(g_2 - g_1 + 1)}{101}$$

את המרחק היחסי הנ"ל ניתן לחשב רק ל-2 ציונים g_1, g_2 אשר מקיימים את כל ארבעת התנאים הבאים יחדיו:

א. g_1 הוא או 0 או אחד הציונים במערך הקלט.

ב. g_2 הוא או 100 או אחד הציונים במערך הקלט.

ג. $g_2 > g_1$

ד. לכל ציון g שנמצא במערך הקלט מתקיים אחד מהשניים: או ש- $g \geq g_2$ או ש $g \leq g_1$

ממשו פונקציה שחתימתה:

```
double max_rel_dist(int grades[], int n);
```

הפונקציה מקבלת מערך של ציונים בין 0 ל-100 ואת אורכו ומחזירה את המרחק היחסי המקסימלי שיכול להתקבל במערך.

למשל עבור מערך הציונים:

80	85	100	80

הפונקציה תחזיר 3.2 שהוא יחס הפיזור המקסימלי. הוא מתקבל עבור הציונים 0 ו-80.

$$\text{relative\_distance}(0,80) = \frac{4 \cdot (80 - 0 + 1)}{101}$$

```
double max_rel_dist(int grades[], int n){
   int hist[101] = {0};
   hist[0] = 1;
   hist[100] = 1;
   for (int i=0; i<n; i++)
      hist[grades[i]] = 1;
   int g1 = 0;
   double max_dist = 0;
   for (int g2=1; g2<=100;g2++) {
      if (hist[g2]) {
         max_dist = (n*(g2-g1+1)/101 > max_dist) ? n*(g2-g1+1)/101 : max_dist;
         g1 = g2;
      }
   }
   return max_dist;
}
```

שאלה 3 (25 נקודות) :
סעיף א' (15 נק')

עליכם לממש את הפונקציה:

```
int strcmp_new(char *s1, char *s2, char *abc);
```

הפונקציה צריכה להשוות בין המחרוזות s1 ו – s2 על-פי סדר האותיות במערך abc. המערך abc הוא מערך בגודל 26 אשר מגדיר סדר חדש של אותיות קטנות באנגלית (כל אות אנגלית קטנה מופיעה בו פעם אחת בדיוק). המחרוזות s1 ו – s2 מכילות אותיות אנגליות קטנות בלבד.

אם s1 תופיע לפני s2 במילון שמוגדר ע"י סדר האותיות במערך abc, יש להחזיר מינוס 1. אם s1 ו – s2 בעלות תוכן זהה יש להחזיר 0, ואם s1 תופיע אחרי s2 במילון שמוגדר ע"י סדר האותיות במערך abc, יש להחזיר 1.

לדוגמא, אם מערך abc בעל התוכן הבא (משמאל לימין):

'b'	'c'	'd'	'a'	'e'	'f'	'm'	'g'	'h'	'i'	'j'	'k'	'l'	'n'	'o'	'z'	'p'	'q'	'r'	's'	't'	'u'	'v'	'w'	'x'	'y'

המחרוזת "s1="air תופיע במילון אחרי "s2="balloon כי במערך abc האות b קודמת לאות a, וערך ההחזרה של הפונקציה יהיה 1.

המחרוזת "s1="zero תופיע במילון לפני "s2="root כי במערך abc האות z קודמת לאות r, וערך ההחזרה של הפונקציה יהיה מינוס 1.

המחרוזת "s1="bool תופיע במילון אחרי "s2="boom כי במערך abc האות m קודמת לאות l, וערך ההחזרה של הפונקציה יהיה 1.

עבור "s1="language ו- "s2="language ערך ההחזרה יהיה 0 כי המחרוזות זהות.

דרישות: סיבוכיות זמן O(n) וסיבוכיות מקום נוסף O(1), כאשר n הוא מספר התוים במחרוזת הקצרה מבין s1, s2.

אסור לשנות את תוכן המערכים שמתקבלים כפרמטרים.

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

```
int strcmp_new(char *s1, char *s2, char *abc) {
	while (*s1 && *s2 && *s1==*s2) {
		s1++;
		s2++;
	}
	if (*s1==*s2) return 0;
	if (!*s1) return -1;
	if (!*s2) return 1;
	int i = 0;
	while (i < 26) {
		if (abc[i]==*s1) return -1;
		if (abc[i]==*s2) return 1;
		i++;
	}
	return -2; // we won't get here
}
```

סעיף ב' (10 נק')

עליכם לממש את הפונקציה:

```
int find_sorted_words(char *words[], int m, char *abc);
```

הפונקציה מקבלת מערך words של m מחרוזות ומחזירה אינדקס $0 \le i \le m-2$ עבורו הקריאה לפונקציה strcmp_new(words[i], words[i+1], abc) תחזיר -1.

לדוגמא, אם נתון המערך abc מהדוגמא שניתנה בסעיף א':

'b'	'c'	'd'	'a'	'e'	'f'	'm'	'g'	'h'	'i'	'j'	'k'	'l'	'n'	'o'	'z'	'p'	'q'	'r'	's'	't'	'u'	'v'	'w'	'x'	'y'

ונתון המערך:

char *words[6] = {"bool", "air", "balloon", "root", "zero", "language" }

הפונקציה יכולה להחזיר 0 כיוון שהמילה "bool" מופיעה לפני המילה "air".

הפונקציה יכולה גם להחזיר 2 כיוון שהמילה "balloon" מופיעה לפני המילה "root".

ניתן להשתמש בפונקציה מסעיף א' גם אם לא מימשתם אותה.

ידוע כי:

- m>=2
- המילה הראשונה במערך מופיעה לפני המילה האחרונה במערך, לפי סדר האותיות במערך abc. כלומר מובטח שהקריאה לפונקציה strcmp_new(words[0], words[m-1], abc) תחזיר -1.

דרישות: סיבוכיות זמן $O(n \cdot \log m)$ וסיבוכיות מקום נוסף O(1) כאשר n הוא מספר התווים במחרוזת הארוכה ביותר ב – words ו – m הוא מספר המחרוזות ב – words.

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

```
int find_sorted_words(char *words[], int m, char *abc)
{
	int l = 0, h = m-1;
	while (h-l>1) {
		int mid = (l+h)/2;
		if (strcmp_new(words[mid], words[mid+1], abc)==-1) return mid;
		if (strcmp_new(words[mid], words[l], abc)==-1) l = mid;
		else h = mid;
	}
	return l;
}
```

שאלה 4 (25 נקודות) :

הנסיך הפרסי (מסדרת משחקי המחשב הנודעת) מקפץ על גגות העיר. מטרתו למצוא את הוזיר הרשע ג'אפאר ולהילחם בו, על מנת להציל את הנסיכה.

מפת הגבהים של גגות העיר מיוצגת ע"י מטריצה בגודל NxN (N מוגדר ב-#define) מטיפוס שלם.
ניתן להניח שערך אי-שלילי במטריצה מייצג גובה של גג, ואילו מיקומו של ג'פאר מסומן ע"י קבוע שלילי JAFFAR (מוגדר ב-#define להיות 1-).

בכל צעד יכול הנסיך להתקדם למשבצת סמוכה: צפונה, דרומה, מזרחה או מערבה (לא באלכסון).
אם המשבצת הסמוכה נמצאת באותו גובה - יכול הנסיך ללכת אליה.
בנוסף יכול הנסיך לטפס על גג בגובה יחידה אחת, או לרדת מגג בגובה יחידה אחת או שתיים.
אם הוא יורד מגג בגובה שלוש יחידות, הוא נפצע (מאבד "לב"). אם נפצע שלוש פעמים, הוא נפסל.
אם הוא מנסה לרדת מגג בגובה ארבע יחידות או יותר, או לעלות לגג בגובה 2 יחידות או יותר, הוא נפסל מיד.

ממשו פונקציה שתתכנן את המסלול שיביא את הנסיך לג'אפאר במספר המשבצות הנמוך ביותר, בלי להיפסל. חתימת הפונק' תהיה

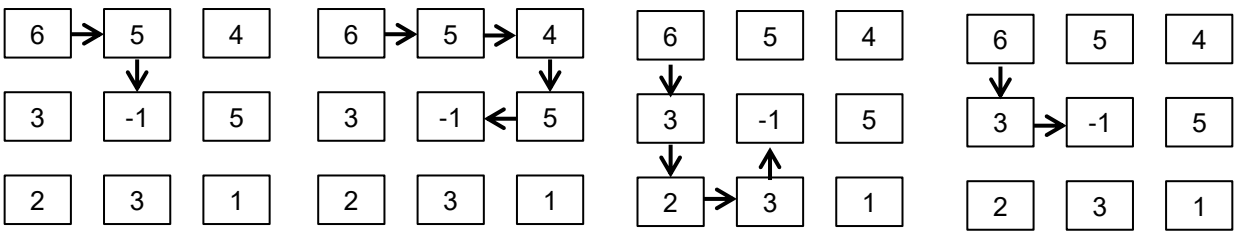
```
int prince_of_persia(int drm[N][N], int p_row, int p_col);
```

כאשר מפת גגות העיר נתונה ע"י הפרמטר drm (digital roofs map), ואילו p_row ו - p_col מציינים את אינדקס השורה והעמודה בהתאמה של המשבצת בה מתחיל הנסיך.
על הפונק' להחזיר את מספר המשבצות במסלול או 1- אם לא קיים מסלול חוקי. לאחר ריצת הפונק' על המפה להישאר ללא שינוי.

לדוגמא עבור N=3 והמפה הבאה:

6	5	4
3	-1	5
2	3	1

אם הנסיך יוצא משורה 0 עמודה 0, מכיוון שג'אפאר נמצא בשורה 1 עמודה 1 קיימים 2 מסלולים באורך 2 ו 2 מסלולים באורך 4, לכן הפונקציה תחזיר 2:



עבור אותה מפה, אם הנסיך יוצא משורה 2 עמודה 2, לא קיים מסלול חוקי ולכן הפונקציה תחזיר 1-.

הערות:

- N מוגדר ע"י `#define`.
- 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.
- ניתן להניח שלג'אפאר אפשר להגיע מכל משבצת סמוכה; הפרשי הגבהים לא משנים.
- ניתן ומומלץ להשתמש בפונק' עזר (ויש לממש את כולן).

```
#define N 4
#define JAFFAR -1
#define MAX_LIVES 3
#define STEP -2
#define INVALID_PATH -1

int min_check_invalid(int val1, int val2) {

   if (val1 == INVALID_PATH)

      return val2;

   if (val2 == INVALID_PATH)

      return val1;

   return val2<val1 ? val2 : val1;

}


int lives_cost(int curr_height, int next_height) {

   int diff_h = next_height-curr_height;

   if (diff_h>-4)

      return diff_h>-3 ? 0 : 1;

   return MAX_LIVES;

}
```

```
bool move_is_valid(int drm[N][N], int i, int j, int next_i, int next_j) {
   if ((next_i==i && next_j==j) || (next_i!=i && next_j!=j))
      return false;
   if (next_i<0 || next_i>=N || next_j<0 || next_j>=N || STEP==drm[next_i][next_j])
      return false;
   if (JAFFAR == drm[next_i][next_j])
      return true;
   int curr_h = drm[i][j],
   next_h = drm[next_i][next_j],
   diff_h = next_h-curr_h;
   return diff_h>-4 && diff_h<2;
}
int prince_of_persia_aux(int drm[N][N], int i, int j, int lives_remain)
{
   if (JAFFAR == drm[i][j])
      return 0;
   if (lives_remain <= 0)
      return INVALID_PATH;
   int drm_bck = drm[i][j];
   int min=INVALID_PATH;
   for (int di=-1; di<=1; ++di) {
      for (int dj=-1; dj<=1; ++dj) {
         if (!move_is_valid(drm, i, j, i+di, j+dj))
            continue;
         drm[i][j] = STEP;
         int lives_lost = lives_cost(drm[i][j], drm[i+di][j+dj]);
```

```
            min = min_check_invalid(min, prince_of_persia_aux(drm, i+di, j+dj, lives_remain-
lives_lost));

            drm[i][j] = drm_bck;

        }

    }

    return INVALID_PATH==min ? INVALID_PATH : min+1;

}

int prince_of_persia(int drm[N][N], int prince_i, int prince_j) {

    return prince_of_persia_aux(drm, prince_i, prince_j, MAX_LIVES);

}
```