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

סמסטר אביב תשע"ז

מבחן מסכם מועד א', 12 ליולי 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. <u>ניתן בהחלט להשתמש בעיפרון ומחק</u>, 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' מירלה בן-חן (מרצה אחראית), דר' יחיאל קמחי, גב' יעל ארז **מתרגלים:** גב' דניאל עזוז, גב' צופית פידלמן, מר תומר לנגה, מר יובל בנאי, דר' יוסי ויינשטיין, מר מוחמד טיבי, מר דמיטרי רבינוביץ', מר יאיר ריעאני, מר איתי הנדלר

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.

```
int f1(int n) {
    int temp = n, m = n;
    while(temp)
    {
        n += m;
        temp /= 2;
    }

    for (int i=0; i <n; i++)
        for (int j=0; j < 8191; j++)
            printf("0");

    return n;
}
```

סיבוכיות זמן: Θ (n log n) סיבוכיות מקום: Θ (1)

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.
הניחו שסיבוכיות הזמן של malloc(n) היא Θ(1), וסיבוכיות המקום של malloc(n) היא Θ(n).

```
int f2(int n) {
    for (int tmp=n; tmp>0; tmp/=2)
    {
        int* p = malloc(tmp);
        free(p);
    }

    for (int i=1; i*i<n; i++)
        for (int j=0; j*i<n; j++)
            printf("0");

    return n;
}
```

סיבוכיות זמן: Θ (n log n) סיבוכיות מקום: Θ (n)

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.

```
#define PARTS 4

void f3(int n) {
    if (n < 4) return;

    for (int i=0; i*i<n; i++)
        printf("%d", i);

    for (int i=0; i<PARTS; i++)
        f3(n/PARTS);
}
```

סיבוכיות זמן: Θ (n) סיבוכיות מקום: Θ (log n)

שאלה 2 (25 נק')

מטריצה ממוינת שורות היא מטריצה בה כל שורה ממוינת מהקטן אל הגדול.

למשל המטריצה הבאה היא ממוינת שורות:

1	5	7	7	9
-4	-3	0	100	101
10	100	200	201	305
0	1	2	3	5

ממשו פונקציה שחתימתה:

```
void sort(int mat[M][N], int sorted[SIZE])
```

אשר מקבלת מטריצה ממוינת שורות וממיינת את הערכים ב – mat במערך **חד-מימדי** באורך SIZE=N*M בו יהיו כל איברי המטריצה ממוינים מהקטן אל הגדול. בסיום ריצת הפונקציה על המטריצה בדוגמא למעלה, תוכן מערך sorted צריך להיות (משמאל לימין):

-4	-3	0	0	1	1	2	3	5	5	7	7	9	10	100	100	101	200	201	305

ניתן להניח ש M הוא חזקה שלמה של 2, וש – M, N, SIZE מוגדרים ב - define.

תוכן מערך sorted לא ידוע בתחילת ריצת הפונקציה.

דרישות:
סיבוכיות זמן $O(NMlogM)$
סיבוכיות מקום נוסף $O(MN)$
כאשר M **מספר השורות** במטריצה ו N **מספר העמודות** (בדוגמא N=5 ,M=4).

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

```
void sort(int mat[M][N], int sorted[SIZE])

{
   int *tmp_array = malloc(sizeof(int) * M*N);
   for (int i = 0; i< M; i++)
      for (int j = 0; j< N; j++)
         sorted[i*N + j] = mat[i][j];

   internal_msort(sorted, M*N, tmp_array);
   free(tmp_array);
}

void merge(int a[], int na, int b[], int nb, int c[])
{
   int ia, ib, ic;
   for(ia = ib = ic = 0; (ia < na) && (ib < nb); ic++)
   {
       if(a[ia] < b[ib]) {
           c[ic] = a[ia];
           ia++;
       }
       else {
           c[ic] = b[ib];
           ib++;
       }
   }
   for(;ia < na; ia++, ic++) c[ic] = a[ia];
   for(;ib < nb; ib++, ic++) c[ic] = b[ib];
}

void memcpy(int dest[], int src[], int n)
{
    for (int i = 0; i<n; i++) {
        dest[i] = src[i];
    }
}

void internal_msort(int a[], int n, int helper_array[])
{
    int left = n / 2, right = n - left;
    if (n <= N)
        return;
    internal_msort(a, left, helper_array);
    internal_msort(a + left, right, helper_array);
    merge(a, left, a + left, right, helper_array);
    memcpy(a, helper_array, n);
}
```

שאלה 3 (25 נקודות) :

יש לכתוב פונקציה שחתימתה

```
int find_single(char *str_arr[], int n)
```

שמקבלת מערך מחרוזות מיוחד שמכיל n מחרוזות, וידוע שכל מחרוזת מופיעה פעמיים ברצף פרט למחרוזת אחת שמופיעה רק פעם אחת, למשל:

```
char *str_arr[] = {"hello", "hello", "world", "world", "CS
spring", "234114", "234114"};
```

לא ניתן להניח הנחות לגבי הכתובות בהן נשמרות המחרוזות גם אם התוכן שלהן זהה. הפונקציה צריכה להחזיר את האינדקס של המחרוזות שמופיעה פעם אחת בלבד, כלומר עבור הדוגמא הנתונה הפונקציה תחזיר 4.

דרישות:

סיבוכיות זמן: $O(m \log n)$ כאשר m מסמן את אורך המחרוזת הארוכה ביותר במערך.

סיבוכיות מקום נוסף: $O(1)$.

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

```
int find_single(char *str_arr[], int n)
{
   int l = 0, r = n-1, mid;
   while (l <= r)
   {
      mid = (l+r) / 2;
      if (mid==0) return 0;
      if (mid==n-1) return n-1;
      if (strcmp(str_arr[mid],str_arr[mid+1]) &&
          strcmp(str_arr[mid],str_arr[mid-1]))
          return mid;
      if (strcmp(str_arr[mid],str_arr[mid+1])) {
         if (mid % 2 == 0)
            r = mid - 2;
         else
            l = mid + 1;
      }
      else {
         if (mid % 2 == 0)
            l = mid + 2;
         else
            r = mid - 1;
      }
   }
   return -1; // should not get here
}
```

```
int strcmp(char str1[], char str2[])
{
   while (*str1 && *str2 && *str1==*str2) {
      str1++;
      str2++;
   }
   return *str1-*str2;
}
```

שאלה 4 (25 נקודות) :

בתחרות עם n משתתפים ו – m שופטים, בשלב הראשון נבחרים המשתתפים אשר ימשיכו לשלב הבא. בחירת המשתתפים נעשית באופן הבא: כל שופט מבצע 3 הצבעות, כל הצבעה קובעת אם משתתף מסוים ימשיך לשלב הבא או לא. לדוגמא, שופט יחיד יכול לבצע את ההצבעות הבאות:

1. משתתף 4 ימשיך לשלב הבא
2. משתתף 2 לא ימשיך לשלב הבא
3. משתתף 19 לא ימשיך לשלב הבא

יש לבחור את המשתתפים שיעלו לשלב הבא בתחרות באופן כזה שהצבעה **אחת לפחות** של **כל אחד מהשופטים** תתקיים.

ממשו את הפונקציה:

```
int select_players(int judges[][3], int nj, int np,
                   bool chosen[])
```

מערך **judges** מכיל את הצבעות השופטים, בכל שורה יש 3 מספרי משתתפים (מספר המשתתף המינימלי הוא 1) שיכולים להיות חיוביים, כדי לסמן שההצבעה היא שהמשתתף ימשיך לשלב הבא, או שליליים, כדי לסמן שההצבעה היא שהמשתתף לא ימשיך לשלב הבא.
לדוגמא, אם יש 4 שופטים ו - 5 משתתפים:

1	2	3
2	3	4
-1	-2	3
1	2	3

judges

במערך משמאל השופט הראשון והרביעי רוצים שמשתתפים 1,2,3 ימשיכו לשלב הבא, השופט השני רוצה שמשתתפים 2,3,4 ימשיכו לשלב הבא, והשופט השלישי רוצה שמשתתפים 1,2 **לא** ימשיכו לשלב הבא ושמשתתף 3 כן ימשיך. אף שופט לא הצביע בעד או נגד משתתף 5.

הפרמטר **nj** הוא מספר השופטים (כלומר מספר השורות במערך **judges**),
והפרמטר **np** הוא מספר המשתתפים. מערך **chosen** הוא מערך בגודל **np**,
ניתן להניח שכל התאים בו הם false כשהפונקציה נקראת לראשונה. בסיום ריצת הפונקציה, מערך chosen צריך להכיל true באינדקס i במידה ומשתתף i+1 ימשיך לשלב הבא, ו – false אחרת (שימו לב: האינדקסים במערך מתחילים מ – 0 ומספר המשתתף המינימלי הוא 1, לכן אינדקס i במערך מתאים למשתתף מספר i+1).
ערך החזרה:מספר המשתתפים שימשיכו לשלב הבא, או 1- במידה ולא ניתן לרצות את כל השופטים.

דוגמאות לפתרונות חוקיים:

true	true	true	true	true

chosen בסיום ריצת הפונקציה
ערך החזרה: 5

true	true	true	true	false

chosen בסיום ריצת הפונקציה
ערך החזרה: 4

true	false	true	true	false

chosen בסיום ריצת הפונקציה
ערך החזרה: 3

true	false	true	false	true

chosen בסיום ריצת הפונקציה
ערך החזרה: 3

דוגמא לפתרון שגוי:
הפתרון שגוי כיוון שאף לא בחירה אחת של השופט השלישי מתקיימת.

true	true	false	false	false

chosen בסיום ריצת הפונקציה
ערך החזרה: 2

הערות:

- 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 שהקלט תקין, כלומר שמערך judges מכיל רק מספרי משתתפים תקינים חיוביים או שליליים.
- ניתן ומומלץ להשתמש בפונק' עזר (ויש לממש את כולן).

```
#define ILLEGAL -1

#define N 3


int select_players(int judges[][n], int nj, int np, bool
                   chosen[])

{

      return select_players_aux(judges, nj, np, chosen, 1);

}


int cnt_players(bool chosen[], int np)

{

      // count number of players that continue to the next
      // level.

      int cnt = 0;

      for (int i = 0; i < np; i++)

            cnt += chosen[i];

      return cnt;

}
```

```
int select_players_aux(int judges[][n], int nj, int np, bool
                       chosen[], int player)

{

      // finished all players

      if (player > np)

            return is_legal(judges, nj, chosen) ?
                   cnt_players(chosen, np) : ILLEGAL;


      // try without this player

      chosen[player - 1] = false;

      int res = select_players_aux(judges, nj, np, chosen,
                                   player + 1);

      if (res != ILLEGAL)

            return res;

      // try with this player

      chosen[player - 1] = true;

      return select_players_aux(judges, nj, np, chosen,
                                player + 1);

}


int abs(int a)

{

      return a > 0 ? a : -a;

}


int selection(int a)

{

      return a > 0 ? 1 : 0;

}
```

```
bool check_judge(int judge[], bool chosen[])
{
      for (int i = 0; i < N; i++)
      {
            int player = abs(judge[i]);
            bool sel = selection(judge[i]);

            if (chosen[player-1] == sel)
                  return true;  // 1 condition is satisfied
      }
      // None condition is satisfied
      return false;
}

bool is_legal(int judges[][n], int nj, bool chosen[])
{
      for (int j = 0; j < nj; j++)
      {
            if (check_judge(judges[j], chosen))
                  continue;
            // if we got here this judge's conditions are not
            // satisfied at all
            return false;
      }
      // All judges are satisfied
      return true;
}
```