אוניברסיטת חיפה **החוג למדעי המחשב**

מבוא למדעי המחשב
סמסטר א' תשע"ו
בחינת סיום, מועד א', 27.1.2016

מרצה: שולי וינטנר

מתרגלים: סמאח גזאווי, ראמי עילבוני

הנחיות:

1. משך הבחינה: 120 דקות.
2. היציאה מהכיתה במהלך הבחינה אסורה.
3. אין להשתמש בכל חומר עזר.
4. קראו היטב כל שאלה וודאו שאתם מבינים אותה לפני שתתחילו לענות עליה. אם יש שאלות, פנו למרצה או למתרגלים.
5. ניתן לכתוב הערות בעברית, גם בגוף פונקציות C.
6. אם לא נכתב אחרת, מותר להגדיר פונקציות עזר.
7. אם לא נכתב אחרת, מותר להשתמש בפונקציות וקבועים שנלמדו בכיתה מתוך הספריות stdio ו-stdlib בלבד. לא ניתן להשתמש בפונקציות אחרות.

בהצלחה!

שאלה 1 (35%)

בהינתן מערך של מספרים שלמים a בגודל n, נאמר ש-k הוא **אינדקס ממצה** אם כל אברי המערך מופיעים במקומות 0 עד k.

למשל, במערך הבא שאורכו 9, 6 הוא אינדקס ממצה כי האברים במקומות ה-7 וה-8 כבר מופיעים קודם לכן:

4, 6, 7, 2, 0, 3, 8, 4, 7

במקרה הקיצוני, אם כל אברי המערך זהים, 0 הוא אינדקס ממצה, ואם האבר האחרון במערך מופיע רק פעם אחת, אז n-1 הוא האינדקס הממצה המינימלי.

הגדירו פונקציה שחתימתה:

```
int covering_index(int a[], int n)
```

הפונקציה מקבלת מערך של שלמים a ואת אורכו, n, כך שכל אברי המערך הם בתחום שבין 0 ל-n-1. הפונקציה מחזירה את האינדקס הממצה המינימלי במערך.

דרישת **סיבוכיות** זמן: O(n). סיבוכיות מקום O(n) (כלומר, ניתן להגדיר מספר קבוע של מערכים נוספים בגודל n).

הבהרות:

1. ניתן להניח שהקלט תקין ואין צורך לבדוק זאת.
2. לא ניתן להשתמש בפונקציה במשתנים חיצוניים או סטטיים.
3. אין להגדיר יותר מפונקציה אחת.

```
int covering_index (int a[], int n)
{
  int i, max = 0;
  int *indices;

  if ((indices=(int *)malloc(n*sizeof(int))) == NULL)
    return -1;
  for (i=0; i<n; i++)
    indices[i] = 0;
  for (i=0;i<n;i++)
    if (indices[a[i]] == 0) {
      indices[a[i]]=1;
      max = i;
    }
  free(indices);
  return max;
}
```

שאלה 2 (35%)

מערך דו-ממדי (ריבועי) של שלמים הוא **ממוין-שורות** אם לכל שורה i>0 מתקיים שכל אבר בשורה ה-i גדול (או שווה) מכל אברי השורה ה-i-1.

לדוגמה, המערך הבא הוא ממוין-שורות:

0	2	1	6
6	10	6	8
20	12	10	12
54	27	29	30

הגדירו פונקציה שכותרתה

```
int search(int a[][N], int x)
```

הפונקציה מקבלת מערך ממוין-שורות בגודל N*N (כש-N הוא קבוע) ומספר שלם x. הפונקציה מחזירה את אינדקס השורה שבה נמצא x, או 1- אם x אינו נמצא במערך. אם x מופיע ביותר משורה אחת, ניתן להחזיר את האינדקס של כל אחת מהן.

למשל, במערך שבדוגמה לעיל, אם x הוא 0 על הפונקציה להחזיר 0. אם x הוא 6 הפונקציה יכולה להחזיר 0 או 1. אם x הוא 12 על הפונקציה להחזיר 2. אם x הוא 13 על הפונקציה להחזיר 1-.

דרישת **סיבוכיות** זמן: O(N).

הבהרות:

1. ניתן להניח שהקלט תקין, ואין צורך לבדוק זאת.
2. בפתרון שאלה זו אסור להגדיר מערכים נוספים.
3. לא ניתן להשתמש במשתנים חיצוניים או סטטיים.
4. ניתן להגדיר יותר מפונקציה אחת.
5. אין להשתמש בפונקציות חיצוניות, אלא אם כן הגדרתם אותן כאן.

הפתרון הבא מקבל גם את ממדי המערך המעשיים (כלומר, הוא לא מניח שהמערך אכן ריבועי). ראשית, מעבר על השורות ב-O(n) כדי למצוא את השורה המתאימה (ייתכנו שתיים כאלו), ואז מעבר ב-O(n) על לכל היותר שתי שורות כדי למצוא את x.

```
int search (int a[][N], int rows, int cols, int x)
{
  int row, col;

  /* find the relevant row(s) */
  for (row=0; row<rows-1 && a[row][0]<x; row++)
    ;
  /* search one or two rows */
  for (col=0; col<cols; col++) {
    if (a[row][col] == x)
      return row;
    if (row>0 && a[row-1][col] == x)
      return row-1;
  }
  return -1;
}
```

שאלה 3 (35%)

עץ טרנארי מלא הוא עץ בו לכל צומת או שלושה בנים (**צומת פנימי**) או אף בן (**עלה**). נתון עץ טרנארי מלא שכל צומת בו כולל נתון אחד, "צבע", שיכול להיות כחול או אדום. מבנה הרשומה הוא כזה:

```
typedef enum {BLUE=0, RED=1} Color;
typedef struct tnode *Tnodep;
typedef struct tnode {
  Color color; /* contents of the node */
  Tnodep left, middle, right; /* three children */
} Tnode;
```

עץ טרנארי כזה הוא **חוקי** אם לכל צומת פנימי v מתקיים שצבע הצומת v הוא הצבע של רוב בניו (מאחר שלכל צומת פנימי בדיוק שלושה בנים, הרוב מוגדר תמיד).

הגדירו פונקציה שחתימתה:

```
int color_tree (Tnodep t);
```

הפונקציה מקבלת מצביע לעץ טרנארי מלא שערכי השדה color בכל הצמתים שלו נתונים. על הפונקציה להחזיר 1 אם העץ חוקי ו-0 אחרת.

הבהרות:

1. ניתן להניח שהעץ הוא אכן עץ טרנארי מלא ואין צורך לבדוק זאת.
2. לא ניתן להשתמש בפונקציה במשתנים חיצוניים או סטטיים.
3. אין להגדיר יותר מפונקציה אחת.

```
int valid (Tnodep p)
{
  int sum;

  if (p == NULL)
    return 1;
  if ((p->left == NULL) ||
      (p->middle == NULL) ||
      (p->right == NULL))
    return 1;
  sum = (p->color == p->left->color) +
        (p->color == p->middle->color) +
        (p->color == p->right->color);
  return (sum > 1 && valid (p->left) &&
          valid (p->middle) && valid (p->right));
}
```