פרק 12 – תבניות אלגוריתמיות

12.1 מערך דו-ממדי

מערך דו-מִמַדִי מייצג מבנה מתמטי הנקרא מטריצה, ובוודאי מוכר לכם מתשבצים, מלוח שחמט ומסודוקו. המבנה מורכב משורות ומעמודות. הנה דוגמה למערך דו-ממדי ובו ארבע שורות וחמש עמודות:

סוס	מתנה	כוכב	רכבת	נחש
חתול	כלב	ציפור	מכונית	ילד
ילדה	בית	עציץ	מטוס	שמש
עלה	פרח	גבעול	שמש	ירח

המערכים שהכרנו עד כה היו מערכים חד-ממדיים.
מערך חד-ממדי הוא בעצם מקרה פרטי של מערך דו-ממדי ובו שורה אחת בלבד. במערך דו-ממדי ניתן לפנות לכל איבר בציון השורה והעמודה שלו.

כיצד סורקים מערך דו-ממדי?

כזכור מפרק 10, לצורך פנייה לאיבר במערך חד-ממדי ציינו את מספר התא שאליו היינו מעוניינים לפנות. כדי לעבור על כל איבריו של מערך חד-ממדי נוח להשתמש בלולאה שמשתנה הבקרה שלה משמש כמציין של תא תורן במערך. כדי לפנות לאיבר במערך **דו-ממדי** יש לציין הן את מספר ה**שורה** והן את מספר ה**עמודה** של האיבר. למשל על מנת לציין את התא שמכיל את המילה "כלב" נפנה למערך בשורה השנייה בטור הרביעי (משמאל). כיצד נציין את התא שמכיל את המילה "ילדה"?

כדי לעבור על כל איבריו של מערך דו-ממדי עלינו להשתמש בלולאות מקוננות (לולאה בתוך לולאה): הלולאה החיצונית תעבור על פני השורות, והלולאה הפנימית תסרוק עבור כל שורה את כל איבריה. בסריקה כזאת נשתמש ב**שני** משתני הבקרה, אחד של הלולאה החיצונית והאחר של הלולאה הפנימית, לצורך ציון התא התורן.

התבוננו באלגוריתם הבא שבו אנו קולטים ערכים במערך דו-ממדי ובו N שורות ו-M עמודות:

1. עבור כל i שווה בין 1 ל-N עשה
   1.1 עבור כל j שווה בין 1 ל-M עשה
      1.1.1 קלוט ערך והשם את הערך במשבץ בשורה ה-i ובעמודה ה-j

כיצד מצהירים על מערך דו-ממדי?

הצהרה על מערך דו-ממדי והקצאת זיכרון עבורו נעשים בדומה להצהרה ולהקצאת זיכרון עבור מערך חד-ממדי. כדי להצהיר על מערך דו-ממדי בשם `numbers` המכיל מספרים שלמים נכתוב:

```
int[][] numbers;
```

שני הזוגות של הסוגריים המרובעים מציינים כי `numbers` הוא מערך דו-ממדי ו-**int** מציין שאיבריו הם מטיפוס שלם.

כדי להקצות זיכרון עבור מערך דו-ממדי יש להשתמש בהוראה **new**. להקצאת זיכרון עבור 3 שורות ו-5 עמודות, תתבצע באופן הבא:

```
numbers = new int[3][5];
```

כעת המשתנה numbers **מפנה** לשטח הזיכרון שהוקצה עבור המערך. כתמיד, ניתן לאחד את ההצהרה ואת ההקצאה להוראה אחת, כך:

```
int[][]numbers = new int[3][5];
```

כפי שציינו בפרק הדן במערכים חד-ממדיים, מומלץ להשתמש בקבועים לצורך הגדרת גודל המערך, למשל נוכל להצהיר על המערך numbers באופן הבא:

```
final int NUM_OF_ROWS = 3;
final int NUM_OF_COLS = 5;
int[][]numbers = new int[NUM_OF_ROWS][NUM_OF_COLS];
```

בדומה לפנייה אל איבר במערך חד-ממדי, גם במערך דו-ממדי הפנייה נעשית באמצעות סוגריים מרובעים, אך במקרה של מערך דו-ממדי, הפנייה צריכה להתייחס גם למספר השורה של התא המבוקש וגם למספר העמודה. גם כאן, בדומה למערך חד-ממדי, מספור השורות והעמודות מתחיל מאפס. למשל, numbers[2][3] מתייחס לתא הנמצא **בשורה** השלישית (שמספרה הוא 2) ובעמודה הרביעית (שמספרה הוא 3).

התכונה length מאפשרת לנו לברר את ממדי המערך הדו-ממדי באופן הבא: הערך של numbers.length מציין את מספר השורות במערך numbers, במקרה זה 3, ואילו הערך של numbers[i].length מציין את מספר האיברים בשורה i. במקרה זה, ערכו של numbers[0].length הוא 5 כי בשורה שמספרה 0 יש 5 איברים.

התבוננו בקטע התוכנית הבא המציג את איברי המערך הדו-ממדי numbers בצורת טבלה:

```
for(int i = 0; i < numbers.length; i++)
{
     for(int j = 0; j < numbers[i].length; j++)
         System.out.print(numbers[i][j] + " ");
     System.out.println();
}
```

נסכם את המושגים הבסיסיים הנוגעים למערכים דו-ממדיים ולעבודה עמם בשפת Java:

- **מערך דו-ממדי בשפת Java** הוא מבנה נתונים, המורכב משורות ומעמודות.
- **הקצאת זיכרון** עבור מערך דו-ממדי מתבצעת באמצעות ההוראה **new**.
- **ציון** איברי מערך דו-ממדי (בדומה למערך חד-ממדי) מתחיל משורה 0 ומעמודה 0.
- **גישה** לאיבר במערך דו-ממדי נעשית בציון השורה והעמודה של האיבר הרצוי, למשל: matrix[i][j]
- **סריקת** מערך דו-ממדי מתבצעת באמצעות לולאות מקוננות.
- **התכונה length** של מערך דו-ממדי מציינת את מספר השורות במערך, למשל: matrix.length
- **התכונה length** של שורה מסוימת במערך דו-ממדי מחזירה את מספר האיברים באותה השורה, למשל: matrix[0].length

בעיה 1

מטרת הבעיה ופתרונה: הצגת מערך דו-ממדי.

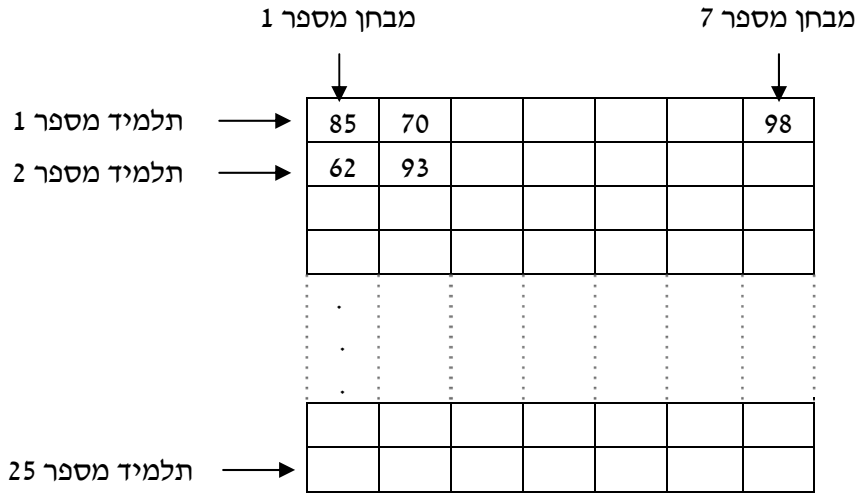
בכיתת מדעי המחשב 25 תלמידים. במהלך השנה התקיימו בדיוק 7 מבחנים. פתחו וישמו אלגוריתם שיקבל כקלט את ציוני שבעת המבחנים של כל תלמיד. בנוסף, האלגוריתם יקלוט מספר המציין מבחן (מספר בין 1 ל-7), ומספר המציין תלמיד (מספר בין 1 ל-25), ויציג כפלט:

1. את ממוצע ציוני שבעת המבחנים של התלמיד שמספרו התקבל כקלט.
2. את ממוצע הציונים של כל 25 התלמידים במבחן שמספרו התקבל כקלט.
3. את הציון הגבוה ביותר שהיה במבחן כלשהו במהלך השנה במדעי המחשב.

כיצד נשמור את המידע הדרוש?

ישנם 25 תלמידים ולכל תלמיד 7 ציונים, ניתן לחשוב על כך בתור טבלה בת 25 שורות – שורה עבור כל תלמיד, ו-7 עמודות – עמודה עבור כל ציון, כך שבכל שורה יהיו שבעת הציונים של תלמיד מסוים.

התבוננו באיור הבא:



סידור זה של הציונים במערך דו-ממדי יאפשר לסרוק את הנתונים של תלמיד מסוים ושל מבחן מסוים. סריקת ציוני תלמיד תתבצע באמצעות סריקת השורה המתאימה לו במערך, ואילו סריקת ציוני מבחן מסוים תתבצע באמצעות סריקת העמודה המתאימה לו במערך.

הגדרת המחלקה *ציוני התלמידים*

נזדקק למחלקה המייצגת את ציוני התלמידים, ובה מערך דו-ממדי.

הגדרת התכונות

- **grades** – מערך דו-ממדי של שלמים המכיל את ציוני התלמידים בכיתה.

הגדרת הפעולות

- **פעולה בונה** – הפעולה תקבל כפרמטר את מספר התלמידים בכיתה, ואת מספר המבחנים ותקצה זיכרון עבור מערך הציונים.
- **עדכון ציון** – הפעולה תקבל את מספר התלמיד, את מספר המבחן ואת הציון במבחן ותעדכן את הציון במערך הציונים.
- **ממוצע תלמיד** – הפעולה תקבל מספר תלמיד (`studentNumber`) ותחזיר את ממוצע המבחנים של התלמיד הנתון.

בפעולה זו אנו נדרשים לעבור רק על שורה אחת מסוימת בטבלה. מעבר על שורה אחת דומה למעבר על מערך חד-ממדי (שבו כזכור יש רק שורה אחת). לכן אנו זקוקים כאן ללולאה אחת בלבד ולא לקינון לולאות. במהלך ביצוע הלולאה מספר השורה יישאר קבוע ואילו מספר העמודה ישתנה, בהתאם למשתנה הבקרה. כמו כן נזכור כי מספור העמודות והשורות ב-Java מתחיל ב-0.

1. *אפס את משתנה סכום הציונים של התלמיד*
2. ***עבור*** *כל* i *שעם בין 0 למספר המבחנים פחות 1* ***בצע:***
   2.1 *הוסף לסכום הציונים את הערך הנמצא במערך הציונים בשורה 1-*studentNumber *ובעמודה* i
3. *חשב את ממוצע התלמיד כסכום הציונים חלקי מספר המבחנים*

- **ממוצע מבחן** – הפעולה תקבל מספר מבחן (`testNumber`) ותחזיר את ממוצע כל התלמידים במבחן הנתון.

בדומה לפעולה הקודמת, כדי לעבור רק על עמודה אחת בטבלה (העמודה `testNumber-1`) נזדקק שוב רק ללולאה אחת, שמשתנה הבקרה שלה נע מ-0 עד למספר התלמידים פחות אחת. במהלך ביצוע הלולאה מספר השורה ישתנה בהתאם למשתנה הבקרה, ומספר העמודה יישאר קבוע:

1. *אפס את משתנה סכום הציונים של המבחן*
2. ***עבור*** *כל* i *שעם בין 0 למספר התלמידים פחות 1* ***בצע:***
   2.1 *הוסף לסכוםהציונים את הערך הנמצא במערך הציונים בשורה* i *ובעמודה 1-*testNumber
3. *חשב את ממוצע המבחן כסכום הציונים חלקי מספר התלמידים*

- **ציון מקסימום** – הפעולה תחזיר את הציון הגבוה ביותר השמור במערך הציונים.

פעולה זו דורשת מעבר על **כל** הטבלה (בלולאה מקוננת), תוך מציאת המקסימום בערכי הטבלה (ראו ביישום האלגוריתם בהמשך).

מימוש המחלקה

```
/*
   מחלקת ציוני התלמידים
*/
public class StudentGrades
{
      // הצהרת מערך הציונים
      private int[][] grades;
```

```
    // פעולה בונה
    public StudentGrades(int students, int tests)
    {
        grades = new int[students][tests];
    }
    // עדכון ציון מבחן של תלמיד נתון
    public void setGrade(int studentNumber, int testNumber,
                                                        int grade)
    {
        grades[studentNumber-1][testNumber-1] = grade;
    }
    // מציאת ממוצע תלמיד נתון
    public double studentAverage(int studentNumber)
    {
        int sumStudent = 0;
        for (int i = 0; i < grades[studentNumber-1].length; i++)
             sumStudent = sumStudent + grades[studentNumber-1][i];
        return (double)sumStudent / grades[studentNumber-1].length;
    }
    // מציאת ממוצע מבחן נתון
    public double testAverage(int testNumber)
    {
        int sumTest = 0;
        for (int i = 0; i < grades.length; i++)
             sumTest = sumTest + grades[i][testNumber-1];
        return (double)sumTest / grades.length;
    }
    // מציאת ציון מקסימלי
    public int maxGrade()
    {
        int max=0;
        for(int i = 0; i < grades.length; i++)
            for(int j = 0; j < grades[i].length; j++)
                if (grades[i][j] > max)
                    max = grades[i][j];
         return max;
    }
} // class StudentGrades
```

שימו ♥: הפעולה הבונה מקבלת כפרמטר את מספר התלמידים ואת מספר הציונים ומקצה מקום למערך דו-ממדי שיכיל את הציונים. ציוני התלמידים ייקלטו בפעולה הראשית ויתעדכנו באמצעות פעולת הגישה setGrade. מיד נציג את מימוש הפעולה הראשית.

הגדרת הפעולה הראשית

פירוק הבעיה לתת-משימות:

משימות הפעולה הראשית הן:

1. יצירת עצם מסוג StudentGrades, קליטת נתוני הציונים ועדכונם.
2. קליטת מספר תלמיד studentNumber ומספר מבחן testNumber.
3. חישוב ממוצע התלמיד שמספרו studentNumber.
4. חישוב ממוצע המבחן שמספרו testNumber.

5. מציאת הציון המקסימלי.

תת-המשימה הראשונה כוללת יצירת עצם מסוג StudentGrades. לצורך כך נגדיר קבועים המייצגים את מספר התלמידים ואת מספר המבחנים ונעביר אותם לפעולה הבונה. לאחר מכן נקלוט את הציונים ונעדכן אותם בעצם ציוני התלמידים.

תת-המשימה השנייה היא קליטת שני ערכים לשני משתנים מטיפוס שלם. בתת-משימה השלישית עד החמישית נבצע את החישובים באמצעות פעולות המוגדרות בעצם StudentGrades.

מימוש הפעולה הראשית

```
/*
   המחלקה הראשית המשתמשת במחלקת ציוני התלמידים
*/
import java.util.Scanner;
public class StudentTest
{
    public static void main(String[] args)
    {
      // הגדרת קבועים - ממדי המערך
      final int NUM_OF_STUDENTS = 25;
      final int NUM_OF_TESTS = 7;
      //    הגדרת משתנים
      double sAverage;
      double tAverage;
      int grade;
      int studentNum;
      int testNum;
      int max;
      // יצירת עצם מסוג מחלקת הציונים
      StudentGrades studentGrades =
                    new StudentGrades(NUM_OF_STUDENTS, NUM_OF_TESTS);
      // קליטת הציונים
      Scanner in = new Scanner(System.in);
      System.out.println(" Enter student grades " );
      for(int i = 1; i <= NUM_OF_STUDENTS; i++)
      {
            for(int j = 1; j <= NUM_OF_TESTS; j++)
            {
                System.out.print("Student number " + i +
                                          " Test number " + j + ": ");
                grade = in.nextInt();
                studentGrades.setGrade(i,j,grade);
            } // for j
      } // for i
      //קליטת מספר תלמיד ומספר מבחן
      System.out.print("Enter student number: ");
      studentNum = in.nextInt();
      System.out.print("Enter test number: ");
      testNum = in.nextInt();
      // חישוב הערכים המבוקשים באמצעות זימון הפעולות המתאימות
      sAverage = studentGrades.studentAverage(studentNum);
      tAverage = studentGrades.testAverage(testNum);
```

```
      max = studentGrades.maxGrade();
      System.out.println("The average of student " + studentNum +
                                                  " is " + sAverage);
      System.out.println("The average of test " + testNum +
                                                  " is " + tAverage);
      System.out.println("The highest grade is: " + max);
    }// main
 }// class StudentTest
```

שימו ♥: מאחר שמספור איברים במערך מתחיל מ-0, ואילו מספור התלמידים והמבחנים מתחיל מ-1, הרי שהתלמיד הרביעי הוא התלמיד במקום 3 במערך. לכן בפעולה setGrade אנו מפחיתים 1 ממספר התלמיד וממספר המבחן:

```
grades[studentNumber-1][testNumber-1] = grade;
```

סוף התרגול בשיעור 1

שאלה 12.1

בנו מחלקה ובה מערך דו-ממדי המכיל מספרים שלמים. המחלקה תכיל את הפעולות הבאות:

א. פעולה בונה המקבלת את ממדי המערך הדו-ממדי ומקצה עבורו מקום.

ב. פעולת גישה לאתחול תא במערך הדו-ממדי. הפעולה מקבלת מספר שורה, מספר עמודה ומספר שלם ומעדכנת את התא המתאים.

ג. פעולה המקבלת מספר שורה ומחזירה את סכום איברי השורה.

ד. פעולה המציגה את ערכי האיברים הנמצאים בשורות **הזוגיות**.

ה. פעולה המקבלת מספר שורה ומחזירה "אמת" אם כל איברי השורה מתחלקים ב-3.

שאלה 12.2

א. פתחו אלגוריתם המאחסן בכל תא של מערך דו-ממדי מספר אשר ספרת העשרות שלו שווה למספר השורה של התא, וספרת האחדות שלו שווה למספר העמודה של התא. למשל, מערך דו-ממדי בגודל 3x4 יראה כך:

0	1	2	3
10	11	12	13
20	21	22	23

ב. הגדירו מחלקה הכוללת מערך דו-ממדי כתכונה ואת הפעולות הבאות:
- פעולה בונה המקבלת את ממדי המערך, מקצה עבורו מקום ומיישמת את האלגוריתם שפיתחתם בסעיף הקודם.
- פעולה להצגת תוכן המערך הדו-ממדי בצורת טבלה.

ג. הגדירו פעולה ראשית הקולטת מהמשתמש את ממדי המערך הרצוי, מייצרת עצם מהסוג שהוגדר בסעיף הקודם ומציגה את המערך שנוצר.

שאלה 12.3

בנו מחלקה ובה מערך דו-ממדי המכיל מספרים ממשיים. המחלקה תכיל את הפעולות הבאות:

א. פעולה בונה המקבלת את ממדי המערך החד-ממדי ומקצה עבורו מקום.

ב. פעולת גישה לאתחול תא במערך הדו-ממדי. הפעולה מקבלת מספר שורה, מספר עמודה ומספר ממשי ומאתחלת את התא המתאים.

ג. פעולה המקבלת מספר שורה ומספר עמודה של איבר במערך ומחזירה את סכום הערכים שמסביב לאיבר (מעליו, מתחתיו, מימינו ומשמאלו).

ד. פעולה המקבלת מספר שורה ומספר עמודה של איבר במערך ומחזירה את סכום הערכים שמסביב לאיבר כולל הערכים הממוקמים אלכסונית לו.

שימו ♥: בשתי הפעולות האחרונות יש להיזהר מחריגה מגבולות המערך! כלומר להתייחס רק לשכנים שבגבולות המערך.

שאלה 12.4

פתחו וישמו אלגוריתם לניהול הזמנת מקומות במטוס. הקלט הוא סדרה של בקשות להזמנת מושבים במטוס. כל בקשה מורכבת ממספר שורה ומאות המייצגת את המושב ('a', 'b', 'c', ...). לאחר כל בקשה תוצג הודעה: "בקשתך התקבלה" או "המקום שביקשת תפוס". לאחר סיום העיבוד של כל בקשה יוצג מצב התפוסה של כל המושבים במטוס, כאשר A מסמן מושב פנוי ו-N מסמן מושב תפוס. הקלט יסתיים כאשר יוקלד המושב A 1 (והוא שייך כידוע לדיילת הראשית).

הדרכה: הגדירו מחלקה המייצגת מטוס. המחלקה תכלול מערך דו-ממדי בוליאני המייצג את המושבים ואת הפעולות הבאות:

א. פעולה בונה המקבלת את ממדי המטוס (מספר שורות ומספר מושבים בכל שורה). הפעולה תקצה את מערך המושבים ותאתחל את כל המושבים כך שיהיו פנויים.

ב. פעולת הזמנה המקבלת מספר שורה ואות המייצגת את המושב. הפעולה מסמנת את המקום כתפוס ומחזירה ערך בוליאני המציין את הצלחת הפעולה או את כישלונה.

זכרו: כדי להמיר את התווים למקומות במערך ('a' הוא 0, 'b' הוא 1 וכו') יש לחסר מהתו המבוקש את התו 'a'. כלומר אם התו מאוחסן במשתנה ch, נכתוב: 'a'-ch. חשבו מדוע? (ראו פרק 4).

ג. פעולה להצגת מצב מושבי המטוס.

הגדירו פעולה ראשית הכוללת את קליטת ממדי המטוס, את יצירת העצם מטוס, את קליטת הזמנות המשתמשים ואת ביצוע ההזמנות.

שאלה 12.5

חגית משחקת שש-בש עם לירון בכל יום בשבוע, בכל יום 10 משחקים. תוצאות המשחקים נשמרות במערך דו-ממדי בוליאני בגודל 10×7: כל שורה במערך מייצגת יום בשבוע, השורה תכיל את תוצאות עשרת המשחקים ששיחקו באותו היום. למשל אם חגית ניצחה במשחק השני ביום חמישי אז התא השני בשורה החמישית יכיל את הערך `true`, אם לירון ניצח באותו משחק, תא זה יכיל את הערך `false`. פתחו אלגוריתם שיחשב ויציג:

א. למי מהשניים כמות הניצחונות הגדולה ביותר.

ב. הודעה המציינת את יום המזל של חגית (**היום** שבו לחגית היו הכי הרבה ניצחונות). אם יש יותר מיום אחד כזה ההודעה תציין את היום הראשון שנמצא.

ג. הודעה המציינת את משחק המזל של לירון (מכיוון שביום מתקיימים עשרה משחקים נמספר אותם מ-1 עד 10. **משחק המזל** הוא זה שמספר הניצחונות בו במהלך השבוע הוא הגדול ביותר). אם יש יותר ממשחק אחד כזה ההודעה תציין את המשחק הראשון שנמצא.

הדרכה: הגדירו מחלקה המייצגת את תוצאות המשחקים. המחלקה תכלול מערך דו-ממדי בוליאני, ואת הפעולות הבאות:

- פעולה בונה ליצירת מערך התוצאות.
- פעולת גישה לעדכון תוצאה של משחק מסוים.
- פעולות הבודקות: למי יש יותר ניצחונות, מהו יום המזל של חגית, ומהו משחק המזל של לירון. הפעולות מחזירות ערכים בהתאם.

מערך דו-ממדי ריבועי

מערך דו-ממדי ריבועי הוא מערך שבו מספר השורות שווה למספר העמודות. במערך כזה, ערך התכונה length של המערך הדו-ממדי שווה לערכה של התכונה length של כל שורה ושורה. כלומר בהינתן מערך דו-ממדי ששמו matrix, הערך של `matrix.length` שווה לערך של `matrix[i].length` עבור כל i המייצג מספר שורה.

במערך דו-ממדי ריבועי (או **מטריצה ריבועית**) אפשר להתייחס למושגים אלכסון **ראשי** ואלכסון **משני**:

אלכסון ראשי:

X		
	X	
		X

אלכסון משני:

		X
	X	
X		

חשבו: מה מאפיין את כל התאים באלכסון הראשי? ובאלכסון המשני? לצורך פתרון הבעיה הבאה נזדקק לאפיון זה.

בעיה 2

מטרת הבעיה ופתרונה: הצגת שימוש באלכסונים במטריצה ריבועית.

פתחו אלגוריתם שיקלוט ערכים ממשיים למטריצה ריבועית בגודל n×n. האלגוריתם יחשב ויציג את סכום הערכים באלכסון הראשי ואת סכום הערכים באלכסון המשני. ישמו את האלגוריתם בשפת Java.

הגדרת המחלקה *Diagonal*

הגדרת התכונות

- **matrix** – מערך דו-ממדי ריבועי, המכיל איברים מטיפוס שלם.

הגדרת הפעולות

בנוסף לפעולה הבונה ולפעולת גישה המאתחלת איבר במטריצה (בדומה לפעולות שהוצגו בבעיה מספר 1), נזדקק לשתי פעולות המחשבות את סכומי האלכסונים המבוקשים.

- **sumMain** – הפעולה תחזיר את סכום איברי האלכסון הראשי. לשם סכימה זו עלינו לבדוק **מה מאפיין איבר** באלכסון הראשי. איבר באלכסון הראשי הוא איבר שמספר השורה שלו ומספר העמודה שלו שווים. שימו לב לאיור הבא שבו מופיעים מצייני התאים:

0,0	0,1	0,2	0,3
1,0	**1,1**	1,2	1,3
2,0	2,1	**2,2**	2,3
3,0	3,1	3,2	**3,3**

- **sumSecond** – הפעולה תחזיר את סכום איברי האלכסון המשני. אם כך, מה מאפיין את איברי האלכסון **המשני**?

התבוננו בסכום המציינים של איברים אלה באיור - הוא קבוע וערכו הוא: N-1. N מייצג את מספר השורות והעמודות במטריצה הריבועית. כדי שהמציינים של איברי האלכסון המשני

ישלימו את הסכום הזה , עלינו לפנות לאיבר במקום ה-matrix[i][N-1-i], חבְּרו את האינדקסים, מה קיבלתם?

במטריצה ריבועית בגודל n×n, המציינים של כל תא באיברי **האלכסון הראשי** שווים.
במטריצה ריבועית בגודל n×n, סכום המציינים של כל תא באיברי **האלכסון המשני** שווה ל-n-1.

מימוש המחלקה:

```
/*
   Diagonal מחלקת
*/
public class Diagonal
{
    private double[][] matrix;
    // פעולה בונה
    public Diagonal(int n)
    {
        matrix = new double[n][n];
    }
    // פעולת גישה לעדכון איבר במטריצה
    public void setVal(int row, int col, double val)
    {
        matrix[row][col] = val;
    }
    // חישוב סכום איברי האלכסון הראשי
    public double sumMain()
    {
        double sumMain = 0;
        for (int i = 0; i < matrix.length; i++)
             sumMain = sumMain + matrix[i][i];
        return sumMain;
    }
    // חישוב סכום איברי האלכסון המשני
    public double sumSecond()
    {
        double sumSecond = 0;
        for (int i = 0; i < matrix.length; i++)
             sumSecond = sumSecond + matrix[i][matrix.length-1-i];
        return sumSecond;
    }
}//class Diagonal
```

מימוש הפעולה הראשית

```
/*
    Diagonal המחלקה הראשית המשתמשת במחלקה
*/
import java.util.Scanner;
public class DiagonalTest
{
    public static void main(String[] args)
    {
      // הגדרת קבועים ומשתנים
      final int N = 5;
```

```
        double val;
        // יצירת עצם מסוג מחלקת האלכסונים
        Diagonal diagonalMat = new Diagonal(N);
        // קליטת האיברים
        Scanner in = new Scanner(System.in);
        System.out.println("Enter matrix values " );
        for(int i = 0; i < N; i++)
        {
           for(int j = 0; j < N; j++)
           {
              System.out.print("row " + i + " col " + j + ": ");
              val = in.nextDouble();
              diagonalMat.setVal(i,j,val);
           } // for j
        } // for i
        //הצגת סכום הערכים באלכסון הראשי ובאלכסון המשני
        System.out.println("Main diagonal sum " +
                                                  diagonalMat.sumMain());
        System.out.println("Secondary diagonal sum " +
                                                diagonalMat.sumSecond());
     }// main
  }// class DiagonalTest
```

סוף פתרון בעיה 2

שאלה 12.6

לפניכם כמה קטעי תוכניות בשפת Java, עבור כל אחד מהקטעים כתבו מה יוצג על המסך ובאיזו צורה.

הניחו ש-mat מוגדרת כמטריצה ריבועית בגודל 5×5 והיא מכילה את הערכים הבאים:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

`for(int i = 0; i < mat.length; i++)`
`{`
`  System.out.print(mat[i][2]);`
`}`
	`for(int i = 0; i < mat.length; i++)`
	`{`
	`  System.out.print(mat[2][i]);`
	`}`

`for(int i = 0; i < mat.length; i++)`
`{`
`  System.out.print(mat[i][i]);`
`}`
	`for(int i = 0; i < mat.length; i++)`
	`{`
	`  System.out.print(mat[5][5]);`
	`}`

שאלה 12.7

פתחו וישמו אלגוריתם הבודק לגבי מטריצה ריבועית אם היא:

א. "שוות שורות" – מטריצה "שוות שורות" היא מטריצה שסכום כל שורה בה שווה.

ב. "שוות עמודות" – מטריצה "שוות עמודות" היא מטריצה שסכום כל עמודה בה שווה.

ג. "ריבוע קסם" – "ריבוע קסם" הוא מטריצה "שוות שורות" ו"שוות עמודות" ובנוסף סכום כל שורה שווה לסכום כל עמודה, וסכום כל אלכסון שווה לסכום שורה ולסכום עמודה. הנה דוגמה לריבוע קסם:

8	1	6
3	5	7
4	9	2

הדרכה: הגדירו מחלקה המכילה מערך דו-ממדי של שלמים ואת הפעולות הבאות: פעולה בונה ליצירת המטריצה הריבועית, פעולת גישה לאתחול תא במטריצה ושלוש פעולות בוליאניות הבודקות אם המטריצה היא שוות שורות, שוות עמודות וריבוע קסם.

שימו ♥: לצורך בדיקת ריבוע קסם, ניתן להיעזר בפעולות הקודמות שמימשתם.
הגדירו פעולה ראשית הקולטת מספרים עבור מטריצה בגודל 3×3 ובודקת אם המטריצה היא שוות שורות, אם היא שוות עמודות ואם היא ריבוע קסם.

שאלה 12.8

לחגית נמאס מהשש-בש. היא המציאה משחק חדש הנקרא: "תפוס את האלכסון". מהלך המשחק: מניחים לוח דמקה על השולחן (כלומר מטריצה בגודל 8×8). בכל תור חגית זורקת מטבע על הלוח. אם המטבע נפל מעל לאלכסון הראשי היא מקבלת נקודה, אם הוא נפל מתחת לאלכסון הראשי היא מפסידה נקודה, ואם המטבע נפל בדיוק על האלכסון הראשי היא מקבלת חמש נקודות! עליכם לפתח אלגוריתם שיקבל כקלט סדרה של זוגות המסתיימת ב-1,-1-. כל זוג מתאר את המשבצת שעליה נפל המטבע באותו תור (מספר שורה ומספר עמודה). פלט האלגוריתם יהיה סכום הנקודות שצברה חגית באותו משחק. ישמו את האלגוריתם בשפת Java.

הדרכה: עליכם לכתוב מחלקה עבור המשחק, המחלקה תכלול תכונה המייצגת את גודל הלוח, פעולה בונה המקבלת את גודל הלוח ומאתחלת אותו, ופעולה המבצעת מהלך במשחק (הפעולה תקבל את מיקום המטבע ותחזיר 1-, 1 או 5 לפי כללי המשחק). הפעולה הראשית תיצור עצם מסוג לוח משחק בגודל 8×8, תקלוט זוגות המציינים את מיקום המטבע, תפעיל את פעולת מהלך המשחק, תסכם את הניקוד ולבסוף תציג את התוצאה.

תרגיל 3

מטרת הבעיה ופתרונה: הצגת התבנית "מעבר על זוגות סמוכים בסדרה", ושימוש בפעולה המזמנת פעולה.

מטריצה "סדרתית" היא מטריצה אשר **כל אחת משורותיה** היא סדרה חשבונית.
סדרה חשבונית היא סדרה שבה ההפרש בין כל שני איברים סמוכים הוא קבוע.
למשל, שתי הסדרות הבאות הן סדרות חשבוניות: 13, 11, 9, 7, 5, 3 ו-25-, 20, 15, 10, 5.
פתחו וישמו אלגוריתם אשר יקלוט מספרים שלמים במטריצה ויציג הודעה אם המטריצה סדרתית או לא.

הגדרת המחלקה *Serial*

הגדרת התכונות

- **matrix** – מערך דו-ממדי, המכיל איברים מטיפוס ממשי.

הגדרת הפעולות

בנוסף לפעולה הבונה ולפעולת גישה המאתחלת איבר במטריצה, נזדקק לפעולות הבאות:

- **isMatrixSeries** – פעולה בוליאנית הבודקת אם המטריצה היא "סדרתית". פעולה זו משתמשת בתבנית ***האם כל הערכים בסדרה מקיימים תנאי*** נבצע זאת כך:

  1. *עבור כל שורה במטריצה נבדוק:*
     1.1 *אם השורה <u>אינה</u> סדירה נשבור ונגיד "שקר"*
  2. *נגיד "אמת"*

  מכיוון שהאלגוריתם דורש שנבדוק את כל שורות המטריצה עד שניתקל בשורה שאינה סדרה חשבונית, נגדיר פעולה נוספת הבודקת אם שורה נתונה היא סדרה חשבונית. כעת נוכל להפעיל פעולה זו שוב ושוב (בכל פעם על השורה הבאה) כל עוד השורות הן סדרות חשבוניות. כיוון שפעולה זו תהיה שימושית עבור המחלקה Serial בלבד, ולא עבור המחלקה הראשית, נוכל להגדיר אותה *כפעולה פרטית*. *פעולה פרטית* היא פעולה שהרשאת הגישה אליה היא **private**, וניתן לגשת אליה (להפעיל אותה) רק מתוך המחלקה שבה היא מוגדרת (כמו תכונות פרטיות). התבוננו בפעולה הבאה:

- **isRowArithmetic** – פעולה בוליאנית המקבלת מספר שורה ובודקת אם היא סדרה חשבונית או לא. פעולה זו שימושית רק לצורך ביצוע הפעולה **isMatrixSeries** ולכן היא תוגדר כפרטית. האלגוריתם למימוש פעולה זו ישתמש בתבנית המוכרת לנו ***מעבר על זוגות סמוכים בסדרה***.

מימוש המחלקה:

```
/*
   Serial המחלקה
*/
public class Serial
{
    private double[][] matrix;
    // פעולה בונה
    public Serial(int n, int m)
    {
        matrix = new double[n][m];
    }
    // פעולת גישה לעדכון איבר במטריצה
    public void setVal(int row, int col, double val)
    {
        matrix[row][col] = val;
    }
    // פעולה פרטית הבודקת אם השורה הנתונה היא סדרה חשבונית
    private boolean isRowArithmetic(int row)
    {
        double distance = matrix[row][1] - matrix[row][0];
        for(int i = 2; i < matrix[row].length; i++)
```

```
            if (matrix[row][i] - matrix[row][i-1] != distance)
                return false;
        return true;
    }
    // "פעולה הבודקת אם המטריצה היא "סדרתית
    public boolean isMatrixSeries()
    {
        for(int i = 0; i < matrix.length; i++)
            if (!isRowArithmetic(i))
                return false;
        return true;
   }
}// class Serial
```

שימו ♥:

הפעולה `isRowArithmetic` היא פעולה פרטית, כלומר ניתן לזמן אותה רק מתוך המחלקה `Serial`. במקרים שמחלקה כלשהי מגדירה פעולות עזר והן משמשות רק פעולות אחרות מאותה המחלקה, נגדיר את הרשאת הגישה לפעולות העזר כ-**private**.

תוכלו לממש בעצמכם את המחלקה הראשית לפי הפירוק הבא:

1. יצירת עצם מסוג `Serial` וקליטת ערכים למטריצה.
2. בדיקה אם המטריצה "סדרתית" באמצעות פעולה מתאימה.
3. הדפסת הודעה מתאימה בהתאם לערך שהוחזר.

סוף פתרון בעיה 3

שאלה 12.9

עלינו לפתח מערכת ממוחשבת לרכישת כרטיסי קולנוע. בבית הקולנוע יש שני אולמות: אולם שביט ואולם ארמון. באולם שביט מוצג בימים אלה הסרט מלחמת הכוכבים ובאולם ארמון מוצג בימים אלה הסרט שרק. בכל אולם יש 10 שורות, ו-20 מושבים בכל שורה.

המערכת מציגה למשתמש את הסרטים שמוצגים בבית הקולנוע בימים אלה ומבקשת ממנו לציין את שם הסרט שהוא רוצה לראות. לאחר מכן המערכת שואלת את המשתמש כמה כרטיסים הוא רוצה לרכוש. המערכת מציגה לפני המשתמש את המושבים הפנויים ומבקשת ממנו לבחור מושבים.

בסוף ההזמנה המערכת מדפיסה: נרכשו <מספר כרטיסים> כרטיסים לסרט <שם הסרט> המוצג באולם <שם האולם שהסרט מציג בו>

לפניכם הגדרת המחלקה Cinema. השלימו את המקומות החסרים, והוסיפו מחלקה ראשית שתטפל בפעולות הקולנוע:

```
public class Cinema
{
    private String name;   // שם אולם הקולנוע
    private String movie;  // שם הסרט המוצג
    private boolean[][] freeSeats;   // רשימת מושבי האולם ומצבם
    private final int ROWS = 10;
    private final int COLS = 20;
    // הפעולה הבונה
    public Cinema (String aName, String aMovie)
    {
        name = aName;
        movie = aMovie;
```

```
        freeSeats = new boolean[ROWS][COLS];
        // מסמנים את כל המושבים כפנויים
        for (int i = 0; i < ROWS; i++)
             for (int j = 0; j < COLS; j++)
                  _____________________________;
    }
    // פעולת גישה: מחזירה את שם הסרט המוצג באולם
    public String getMovie ()
    {
        return ___________________;
    }
    // פעולת גישה: מחזירה את שם אולם הקולנוע
    public String getName()
    {
        return ___________________;
    }
    // פעולת גישה: מעדכנת את שם הסרט המוצג באולם
    public void setMovie(String aMovie)
    {
        ____________________________;
    }
    // פעולה בוליאנית הבודקת אם סרט נתון מוצג באולם הקולנוע
    public boolean isShowing(String aMovie)
    {
        return (____________________________)
    }
    // פעולה המחזירה מחרוזת שמתארת את רשימת המושבים הפנויים
    public String getAvailableSeats()
    {
        String availableSeats = new String("");
        for (int i = 0; i <freeSeats.length; i++)
             for (int j = 0; j <freeSeats[i].length; j++)
                  if (____________________________)
                        availableSeats = availableSeats +
                                          ____________________________________;
         return availableSeats;
    }
    // פעולה המקבלת מספר מושב פנוי ומעדכנת את מצבו כתפוס
    // ומחזירה "אמת" אם הפעולה הצליחה ו"שקר" אחרת
    public boolean setSeatAsTaken(int rowSeat, int colSeat)
    {
            if (___________________)
            {
                  _____________________________ = false;
                  return _________________;
            }
            else
                  return _________________;
     }
} // class Cinema
```

שאלה 12.10

כתבו מחלקה המגדירה את "משחק החיים". משחק החיים הוא משחק סימולציה המתאר את מחזור החיים של יצורים חיים. משחקים במשחק במטריצה שכל תא בה מייצג אתר מחייה: בכל אתר מתקיים אחד משני המצבים:

א. "יש חיים" – אתר מחיה מלא – נסמן בתו '1'

ב. "אין חיים" – אתר מחיה ריק – נסמן בתו '0'

לדוגמא, בהינתן המטריצה הבאה:

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

אפשר לראות שבאתר (2,1) יש חיים, ושלאתר זה יש 4 שכנים חיים והם (1,2), (3,0), (3,1) ו- (3,2). לעומת זאת באתר (3,3) אין חיים, ולאתר זה יש שכן חי אחד שהוא (3,2).

חוקי הגנטיקה של המשחק:

- **לידה** – בכל אתר שבו "אין חיים" ויש לו בדיוק 3 שכנים חיים תהיה לידה בדור הבא. אחרת האתר נשאר "ללא חיים".
- **מוות** – בכל אתר שבו "יש חיים" ויש לו לכל היותר שכן אחד שבו יש חיים יתרחש מוות בדור הבא כתוצאה מבדידות. בכל אתר שבו "יש חיים" ולו יש 4 שכנים חיים או יותר, יתרחש מוות בדור הבא כתוצאה מצפיפות.
- **קיום** – כל אתר שבו "יש חיים" ויש לו 2 או 3 שכנים חיים ימשיך להתקיים גם בדור הבא.

תהליכי הלידה, המוות והקיום מתרחשים בו זמנית בכל האתרים ויוצרים מצב חיים חדש הנקרא דור חדש.

כתבו מחלקה המגדירה את "מטריצת החיים". המחלקה צריכה להכיל את לוח המשחק; פעולה המבצעת מעבר דור אחד במשחק, לפי הכללים הנתונים ופעולה המציגה את מטריצת החיים באותו רגע. הפעילו את משחק החיים באמצעות הפעולה הראשית המקבלת כקלט מספר שלם X, יוצרת עצם מסוג מטריצת חיים ומקיימת עבורו X דורות. עליכם להציג את המטריצה לאחר כל דור.

הדרכה: את הדור החדש יש להכין במטריצה זמנית, ולהעתיק אותה למטריצת המשחק בתום הכנתה. הגדירו במחלקת מטריצת החיים פעולת עזר (פרטית) המקבלת מציינים של תא ומחזירה את מספר השכנים החיים שיש לתא זה.

שאלה 12.11

כתבו מחלקה המגדירה לוח של משחק "בינגו". המחלקה תכיל את לוח המשחק, מטריצה בגודל 4×4 ובה מספרים שלמים בתחום 1-100, ולוח בוליאני נוסף (באותו גודל) ותפקידו לסמן את התאים של המספרים שהוכרזו במהלך המשחק. במחלקה יוגדרו הפעולות הבאות:

- פעולה המבצעת מהלך במשחק: הפעולה מקבלת כפרמטר מספר שבחר מנהל המשחק. אם המספר נמצא על לוח המשחק, יש לסמן ב-`true` את המקום המתאים בלוח הבוליאני (אם המספר לא נמצא על הלוח לא יתבצע דבר).
- פעולה שתבדוק אם ניתן להכריז על "בינגו". ניתן להכריז על "בינגו" כאשר יש שורה או טור או אלכסון שכל מספריו נבחרו.

שאלות נוספות בנושא מערך דו-ממדי תוכלו למצוא בפרק 13 "פתרון בעיות".

12.2 חיפוש בינרי

בסעיף זה נלמד דרך נוספת לחפש ערך, מלבד החיפוש סדרתי – שכבר נתקלנו בו בעבר. הדרך לעשות זאת היא *חיפוש בינרי*.

בפרק 7 למדנו כיצד לבצע חיפוש סדרתי במעבר על איברי המערך בזה אחר זה ובהשוואתם לערך המבוקש. החיפוש נפסק כאשר נמצא הערך הדרוש או כאשר הגענו לקצה המערך. במקרה הגרוע (כאשר האיבר המבוקש לא נמצא במערך) נעשו N השוואות אם N הוא גודל המערך. כעת נלמד לבצע חיפוש יעיל יותר, כלומר כזה המבצע **פחות** השוואות, **ומסתמך על העובדה שהנתונים ממוינים.** את החיפוש הצגנו בפרק 8 אך כעת נדון בו בהרחבה.

נסו לחשוב כיצד מחפשים שם במדריך הטלפונים?

פותחים את הספר בערך באמצע ובוחנים את השמות הכתובים בעמוד זה. אם השם המבוקש אינו נמצא בעמוד זה, נמשיך את החיפוש רק בעמודים שאחריו או רק בעמודים שלפניו, זאת לפי תוצאת ההשוואה לשמות שבעמוד. באותו אופן ממשיכים בחיפוש באמצעות פתיחת הספר במחצית המתאימה עד שמוצאים את השם המבוקש או עד שמשתכנעים כי השם לא נמצא.

שימו ♥ לחשיבות ההנחה שהספר ממוין. הנחה זו מאפשרת לנו להמשיך את החיפוש רק במחצית אחת של הספר ולהתעלם מהמחצית השנייה לחלוטין.

האלגוריתם שתיארנו נקרא "חיפוש בינרי" כי בכל צעד אנחנו מחלקים את הנתונים לשני חלקים. (משמעות המילה בינרי = מורכב משני חלקים).

הנה אלגוריתם המשתמש בחיפוש בינרי כדי לחפש את הערך `key` במערך **ממוין**:

```
1.  כל עוד יש איברים במערך בצע
    1.1  אם key שווה לאיבר האמצעי אז מצאנו את key במערך
    1.2  אחרת אם האיבר האמצעי גדול מ-key אז
         1.2.1  המשך את החיפוש במחצית התחתונה של המערך
    1.3  אחרת
         1.3.1  המשך את החיפוש במחצית העליונה של המערך
```

נתון המערך הממוין A להלן, נעקוב אחרי ביצוע האלגוריתם כאשר ערכו של key הוא 50 במערך.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]
20	35	37	42	49	50	52	60	75

A[middle]

אמצע המערך הוא האיבר A[4] וערכו 49. הערך 50 גבוה מ-49, ולכן אם הערך 50 נמצא במערך הרי הוא נמצא בקטע המערך שבין A[5] ל-A[8]:

A[5]	A[6]	A[7]	A[8]
50	52	60	75

A[middle]

נמשיך לחפש את הערך 50 בקטע המערך. אמצע המערך הוא האיבר A[6], וערכו 52. למעשה כאשר אורך המערך הוא זוגי, יש שני איברים שיכולים להיחשב כאמצע המערך – אין חשיבות באיזה מהם נבחר. הערך 50 קטן מ-52, ולכן נמשיך לחפש בקטע המערך שבין A[5] ל-A[5] (מערך בן איבר אחד!):

A[5]
50

השוואת הערך של key ל-A[5] מסתיימת כמובן בהצלחה וניתן להסיק שהערך key (50) נמצא במערך A.

שאלה 12.12

נסו לעקוב אחרי ביצוע האלגוריתם כאשר ערכו של key הוא 51.

כיצד נוכל להמשיך את החיפוש רק בחלק מהמערך? הרי המערך כולו שמור בזיכרון, ולא רק חלק ממנו! לשם בדיקה של מחצית המערך בלבד נשתמש בשני אינדקסים (מציינים) low ו-high שיציינו את גבולות המערך. בתחילת האלגוריתם יהיו האינדקסים שווים לגבולות האמיתיים של המערך (0 ו-N-1) ובמהלך האלגוריתם נצמצם את השטח שבו אנחנו מחפשים. נצמצם את השטח באמצעות משתנה עזר שייקרא middle.

חשבו: מה נשים במשתנה middle כדי שיציין את אמצע המערך בגבולותיו הנוכחיים?

אמצע המערך יתקבל בחישוב: middle=(low+high)/2. אם נרצה לטפל בקטע המערך העליון נתבונן בגבולות: middle+1..high, ואילו קטע המערך התחתון יהיה בגבולות: low..middle-1.

<u>מהו התנאי לעצירת החיפוש</u>? ברגע שמוצאים את הערך key במערך יש לעצור את החיפוש. במקרה שהערך אינו נמצא במערך, נעצור את החיפוש כאשר אי אפשר להקטין עוד את קטע המערך.

<u>כיצד נדע שאי אפשר להקטין עוד את קטע המערך</u>? קבענו כי גבולות קטע המערך הנוכחי הם מ-low ועד high. אם low>high אי-אפשר להקטין עוד את קטע המערך, כי אין איברים בין A[low] ל-A[high]!

לפניכם קטע תוכנית בשפת Java המממש את האלגוריתם של חיפוש בינרי לחיפוש הערך key במערך ממוין A. האלגוריתם מציג הודעה המציינת אם הערך key נמצא או לא נמצא במערך A:

```
int[] A;
int key;
…
int middle;
int low = 0;
int high = A.length-1;
boolean found = false;
while (low <= high && ! found)
{
      middle = (low + high) / 2;
      if (A[middle] == key)   // האיבר המבוקש נמצא
            found = true;
      else
      {
            if (A[middle]> key)     // מחצית תחתונה
                  high = middle – 1;
            else                    // מחצית עליונה
                  low = middle + 1;
      }
```

```
}
if (found)
      System.out.println(key + "found in the array");
else
      System.out.println(key + "was not found in the array");
```

יעילות

חיפוש בינרי יעיל יותר מחיפוש סדרתי (עבור מערכים ממוינים כמובן). בחיפוש בינרי, לאחר כל בדיקה שבה האיבר המבוקש לא נמצא, מספר האיברים הנותרים במערך קטן בחצי. נניח שלפנינו מערך בגודל 1000 איברים והערך שאנחנו מחפשים **לא** נמצא במערך. **בחיפוש סדרתי**, לאחר הבדיקה הראשונה יישארו לנו עוד 999 איברים לבדוק, לאחר מכן 998 וכך הלאה, בסך הכל נבצע את הלולאה, במקרה הגרוע ביותר, 1000 פעמים. **בחיפוש בינרי**, אחרי הבדיקה הראשונה נותרים לנו 500 איברים, אחרי הבדיקה השנייה 250 איברים, אחרי השלישית 125, וכך הלאה 62, 31, 15, 7, 3, 1. לולאת החיפוש תתבצע לכל היותר 10 פעמים. **שיפור עצום לעומת החיפוש הסדרתי!**

מספר הפעמים שהלולאה מתבצעת תלוי כמובן במיקומו של הערך key במערך. אם הערך key אינו נמצא אז הלולאה תתבצע כמספר הפעמים שאפשר לחלק את גודל המערך לשני חלקים. אפילו כשהמערך גדול מאוד. למשל במערך ובו 1,000,000 איברים תתבצע הלולאה לכל היותר 20 פעמים, יעילות-זמן מרשימה מאוד בהשוואה לחיפוש סדרתי. הטבלה שלהלן מציגה עבור ערכים שונים לגודל המערך, את מספר הפעמים המקסימלי שהלולאה תתבצע בחיפוש בינרי. (מספר הפעמים בפועל תלוי במיקומו של האיבר המבוקש במערך, כאן ההתייחסות היא למקרה הגרוע ביותר).

גודל המערך	מספר הפעמים שהלולאה תתבצע
8	4
10	4
16	5
50	6
100	7
1000	10
5000	13
1000000	20

נקטין את המספר N בחצי ונמשיך כך עד שלא נוכל להקטין אותו יותר (כלומר עד שנגיע ל-1). אם נספור את מספר ההקטנות שעשינו נקבל מספר שנקרא $\log_2 N$. למשל $\log_2 16 = 4$ כי צריך להקטין את 16 4 פעמים בחצי עד שמגיעים ל-1 (8, 4, 2 ו-1).

תרגיל 4

מטרת הבעיה ופתרונה: הדגמת שימוש בחיפוש בינרי

בבואו לעבור טסט, מקבל כל נבחן מספר תלת ספרתי. בסוף היום מפורסמים כל המספרים של הנבחנים שעברו את הטסט והם ממוינים בסדר עולה. הגדירו מחלקה המכילה את כל הרשימה של מספרי הנבחנים שעברו טסט, ממוינת בסדר עולה. המחלקה תכיל את הפעולה "האם עברת?" שתקבל את מספר הנבחן ותחזיר אם הוא נמצא ברשימה או לא.

הגדרת המחלקה *DrivingTest*

הגדרת התכונות

- **pass** – מערך ממוין המכיל את מספרי הנבחנים שעברו.

הגדרת הפעולות

- **פעולה בונה** – הפעולה מקבלת מערך ממוין המכיל את מספרי הנבחנים שעברו את הטסט, ומעתיקה אותו למערך pass. בחרנו להעתיק את המערך כדי להגן עליו מפני שינוי מבחוץ.
- **didIPass** – פעולה המקבלת מספר נבחן ומחזירה "אמת" אם מספר זה נמצא במערך העוברים ו"שקר" אחרת. בפעולה זו נוכל לבצע חיפוש בינרי מכיוון שהמערך ממוין.

מימוש המחלקה

```
/*
   DrivingTest המחלקה
*/
public class DrivingTest
{
      private int[] pass;
      //פעולה בונה
      public DrivingTest(int[] p)
      {
            pass = new int[p.length];
            for (int i = 0; i < p.length; i++)
                  pass [i] = p[i];
      }
      public boolean didIPass(int num)
      {
            int middle;
            int low = 0;
            int high = pass.length - 1;
            while (low <= high)
            {
                  middle = (low + high) / 2;
                  if (pass[middle] == num)          // האיבר המבוקש נמצא
                        return true;
                  else
                  {
                        if (pass[middle] > num)       // מחצית תחתונה
                              high = middle - 1;
                        else                          // מחצית עליונה
                              low = middle + 1;
                  }
            }//while
            return false;
      }
}//class DrivingTest
```

סוף התרגון בעיה 4

שאלה 12.13

לפניכם קטע קוד שמבצע חיפוש בינרי במערך arr:

הניחו כי המשתנים high ו-low מאותחלים בגבולות המערך 0 ו-N.

```
boolean flag = true;
while (low <= high)
{
      middle = (low + high) / 2;
      if (arr[middle] != num)
            flag = false;
      if (arr[middle] > num)
            high = middle - 1;
      else
            low = middle + 1;
}
if (flag)
      System.out.println("The Value Is In The Array");
else
      System.out.println("The Value Is Not In The Array");
```

הקטע שגוי. מצאו את השגיאה ותקנו את הקטע כך שיבצע את הנדרש.

שאלה 12.14

נתון מערך ממוין A. כתבו קטע תוכנית בשפת Java שיבדוק כמה פעמים מופיע הערך 25 במערך A. כתבו את הקטע יעיל ככל שתוכלו.

שאלה 12.15

נתון מערך A באורך 16 שאיבריו הם מספרים שלמים מ-1 עד 16, ממוינים בסדר עולה. עבור כל אחד מהערכים של key מ-1 ועד 16 כמה פעמים מתבצעת הלולאה בחיפוש בינרי?

שאלה 12.16

נתון מערך ממוין A בגודל N ובו מספרים שלמים חיוביים (בסדר עולה). כתבו קטע תוכנית שהפלט שלו הוא הודעה אם **רוב** איברי המערך גדולים מן הממוצע של האיבר הראשון והאחרון.

שאלה 12.17

נתון מערך ממוין A בגודל N ובו מספרים שלמים חיוביים עוקבים, מלבד שניים מן המספרים.

א. כתבו קטע תוכנית שהפלט שלו הוא הודעה אם זוג הערכים הלא רצופים נמצאים בחצי הראשון של המערך, בחצי השני של המערך או בדיוק במעבר בין החצי הראשון לחצי השני (N זוגי).

ב. כתבו קטע תוכנית שהפלט שלו הוא **המציינים** של שני איברי המערך שמצאתם בסעיף הקודם, כלומר מיקומם במערך.

שאלה 12.18

מטריצה "ממוינת למחצה" היא מערך דו-ממדי שכל שורה בה ממוינת בסדר עולה. הגדירו מחלקה ובה מטריצה "ממוינת למחצה". במחלקה הגדירו פעולה שתקבל כפרמטר ערך X ותדפיס את מיקומו של X במטריצה, כלומר את מספר השורה והעמודה שהערך X נמצא בהן. נתון שהערך X מופיע בדיוק פעם אחת במטריצה.

הדרכה: השתמשו בפעולת עזר פרטית שתחפש בכל שורה בנפרד.

שאלה 12.19

"סדרת הפרשים עולה" היא סדרת ערכים שבה ההפרשים בין כל זוג ערכים סמוכים עולים-ממש. כלומר ההפרש הגדול ביותר הוא בין האיבר האחרון בסדרה וזה שלפניו, וההפרש הקטן ביותר הוא ההפרש בין האיבר הראשון לשני. דוגמה ל"סדרת הפרשים עולה": 97, 66, 40, 29, 20, 15.
עליכם לכתוב אלגוריתם שיקלוט "סדרת הפרשים עולה" במערך ומספר נוסף k. האלגוריתם יבדוק אם קיים זוג ערכים סמוכים במערך שהפרשם הוא k. ממשו את האלגוריתם בשפת Java.
עליכם לכתוב את האלגוריתם יעיל ככל שתוכלו מבלי להשתמש במערך עזר.

12.3 מיונים

הצורך במיון נתונים מופיע שוב ושוב: מדריך טלפונים ממוין לפי שמות, טבלת הליגה בכדורגל ממוינת לפי מספר הנקודות ועוד... בעיית המיון היא בעיה מרתקת במדעי המחשב, מספר רב של אלגוריתמים פותחו לשם כך, ולכל אלגוריתם יתרונות וחסרונות משלו. בסעיף זה נציג שלושה אלגוריתמים שונים למיון.

מיון בחירה

חשבו: מונחת לפניכם ערמה של מטבעות. איך אפשר למיין אותם לפי ערכן?

השיטה הטבעית היא לאסוף את כל המטבעות הקטנות ביותר של 5 אגורות, אחר כך את המטבעות של 10 אגורות, וכך הלאה עד שנותרת רק ערימת מטבעות של 10 שקלים. שיטה טבעית זו היא הבסיס לאלגוריתם המיון שנקרא "מיון בחירה".

הנה אלגוריתם המשתמש במיון בחירה כדי למיין סדרת מספרים:

עבור כל i *מ-1 עד אורך הסדרה בצע*

*השם במיקום ה-*i *את הערך הקטן ביותר מבין הערכים שבהם לא טיפלנו*

נניח שמצאנו את הערך הקטן ביותר בסדרה. האלגוריתם דורש לשים אותו במקום הראשון. אבל מקום זה תפוס בידי ערך אחר, ואם נשים במקומו את הערך הקטן ביותר נאבד את הערך שנמצא במקום הראשון. **חשבו**: היכן ניתן לשמור את הערך שנמצא במקום הראשון? נשמור אותו במקום שהתפנה מהאיבר הקטן ביותר. אם האיבר הקטן ביותר נמצא במקום j, נחליף בין האיבר הנמצא במקום הראשון לאיבר הנמצא במקום ה-j.

נתבונן לדוגמה במערך A הבא:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
2	8	20	4	7	0	15	18	1	11

הערך הקטן ביותר 0, נמצא ב-A[5], אם ברצוננו להעביר את 0 ל-A[0] אז נחליף את A[5] ב-A[0]:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
0	8	20	4	7	2	15	1	18	11

דוגמה זו מראה את הצעד הראשון באלגוריתם למיון. הצעד השני יהיה מציאת הערך הקטן ביותר בשארית המערך A. כלומר מבין הערכים שנמצאים בתאים A[1] עד A[9]. הערך העונה על הדרישה הוא 1 השמור ב-A[7], ולכן נחליף אותו עם הערך 8 הנמצא ב-A[1]:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
0	1	20	4	7	2	15	8	18	11

שאלה 12.20

שרטטו תרשים של המערך לאחר שתי ההחלפות הבאות.

עתה אפשר להשלים את האלגוריתם למיון המערך A:

עבור כל i *מ-0 עד* A.length-2 *בצע:*
מצא את הערך הקטן ביותר בקטע המערך A[i]..A[A.length-1]
*ושמור את המציין שלו ב-*iMin
החלף את האיבר A[i] *עם האיבר* A[iMin]

שיטת המיון שתיארנו כאן נקראת "מיון בחירה" (Selection Sort). בכל שלב באלגוריתם בוחרים את האיבר הקטן ביותר בשארית המערך ומכניסים אותו למקום המתאים.

פרויקט 5

מטרת הבעיה ופתרונה: הצגת מיון בחירה בשפת Java.

מנהלת בית הספר החליטה כי בספריית בית הספר, הספרים יהיו מסודרים לפי עוביים. עזרו לספרן המבולבל לסדר את הספרים. הגדירו מחלקה ובה מערך ובכל תא בו יש מספר עמודים של ספר אחד. במחלקה תהיה הפעולה **selectionSort** שבעת הצורך תמיין את המערך מהערך הקטן לגדול.

הגדרת המחלקה *Library*

הגדרת התכונות

- **bookSizes** – מערך חד-ממדי לשמירת עוביו של כל ספר.

הגדרת הפעולות

- **פעולה בונה** – הפעולה מקבלת מערך המכיל את עובי הספרים, ותעדכן את המערך bookSizes.
- **selectionSort** – פעולה הממיינת את המערך שבמחלקה לפי אלגוריתם "מיון בחירה".
- **getBookSizes** – פעולת גישה המחזירה את המערך הממוין.

מימוש המחלקה

```
/*
       Library המחלקה
*/
public class Library
{
    private int [] bookSizes;
    // פעולה בונה
    public Library(int[] books)
```

```
    {
        bookSizes = books;
    }
    public void selectionSort()
    {
        int iMin, temp;
        for(int i = 0; i < bookSizes.length - 1; i++)
        {
            iMin=i;
            // מציאת מינימום
            for(int j = i + 1; j < bookSizes.length; j++)
                if(bookSizes[j] < bookSizes[iMin])
                    iMin = j;
             // החלפת איברים
             temp = bookSizes[iMin];
             bookSizes[iMin] = bookSizes[i];
             bookSizes[i] = temp;
        }
     }
     public int[] getBookSizes()
     {
        return bookSizes;
     }
} // Library
```

סוף פתרון בעיה 5

שאלה 12.21

שנו את פעולת המיון בבעיה 5 כך שהיא תזמן שתי פעולות עזר במקום לטפל בכל תתי-המשימות בעצמה. פעולות העזר הן: פעולה המחזירה את המציין של הערך המינימלי בקטע נתון של המערך ופעולה המחליפה בין שני ערכים במערך. לשתי הפעולות יתקבלו אינדקסים כפרמטרים. את הפעולות יש לממש כפעולות פרטיות במחלקה Library.

יעילות

חשבו: אם N הוא מספר איברי המערך, כמה פעמים מתבצע גוף הלולאה במיון בחירה?

עבור i = 0, הלולאה הפנימית מתבצעת N-1 פעמים

עבור i = 1, הלולאה הפנימית מתבצעת N-2 פעמים

עבור i= 2, הלולאה הפנימית מתבצעת N-3 פעמים

. . . . .

עבור i = N-1 הלולאה הפנימית מתבצעת N-(N-1) = 1 פעמים

לפי הנוסחה לסכום סדרה חשבונית נקבל שמספר הפעמים שהלולאה מתבצעת הוא:

$$1 + 2 + \ldots + (N-1) = \frac{(N-1)\cdot N}{2}$$

מיון בחירה של N מספרים דורש בערך N^2 ביצועים של גוף הלולאה. מספר זה גדל בקצב גבוה ומגיע לחצי מיליון עבור מערך בן 1000 איברים.

שאלה 12.22

האם מספר הפעמים שהלולאה מתבצעת במיון בחירה תלוי בתוכן המערך?

שאלה 12.23

כמה פעולות החלפה מתבצעות במיון בחירה?

שאלה 12.24

התבוננו בפעולה selectionSort (בבעיה 5). כמה פעמים מתבצע המשפט `iMin = j` בפעולה למיון בחירה?

שאלה 12.25

כתבו אלגוריתם המקבל כקלט סדרת מספרים בסדר כלשהו. האלגוריתם יציג את מספר הערכים שמיקומם המקורי הוא גם מיקומם בסדרה **המסודרת** של הערכים הנתונים.

הדרכה: קלטו את האיברים במערך, מיינו במיון בחירה ומנו תוך כדי כך כמה איברים נמצאו במיקומם.

שאלה 12.26

הגדירו מחלקה ובה מערך חד-ממדי בגודל N. הגדירו במחלקה פעולה שתקבל כפרמטר מספר שלם חיובי K קטן מ-N, וערך S. הפעולה תחזיר "אמת" אם קיימים K מספרים במערך שסכומם קטן מ-S ו"שקר" אחרת.

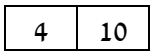
מיון הכנסה

בתת-סעיף זה נציג מיון נוסף הנקרא מיון הכנסה. זמן הביצוע של מיון הכנסה אמנם דומה לזה של מיון בחירה, אך על סוגי קלט רבים מיון הכנסה יעיל יותר.

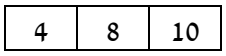
מיון הכנסה מבוסס על הדרך שבה שחקן קלפים מסדר יד קלפים. השחקן מרים את הקלפים אחד-אחד ושומר על הקלפים שבידו באופן ממוין. כשהוא מקבל קלף חדש הוא מפנה לו מקום, ומכניס אותו למקום הנכון ביד.

יד עם קלף אחד:

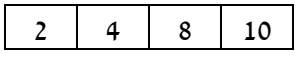
אחרי קבלת הקלף 4:



אחרי קבלת הקלף 8:



אחרי קבלת הקלף 2:



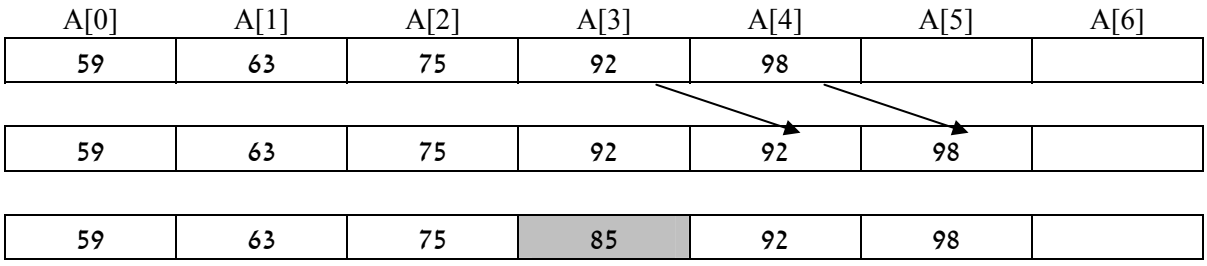
ברור שלאחר ביצוע צעדי ההכנסה עבור כל הקלפים שקיבלנו יהיו הקלפים ממוינים. הפעולה שבבסיס מיון הכנסה היא הצעד "הכנס ערך חדש למערך ממוין". כעת נתחיל בדיון לפיתוח אלגוריתם לביצוע פעולה זו.

הכנסת איבר במערך ממוין

הכנסת איבר חדש למערך תגרום להזזת מספר איברים.

חשבו: מי הם האיברים במערך שצריכים לזוז? האיברים שצריכים לזוז הם האיברים שערכיהם גדולים מהאיבר שנרצה להכניס. התרשימים שלהלן מראים את המערך A ובו 5 ערכים, ואת

הכנסת הערך החדש 85. האיברים שצריכים לזוז "ימינה" הם A[4] ו- A[3]. לאחר ההזזה, אפשר להכניס את הערך החדש ל-A[3] שיהיה המקום המקורי של הערך 92:



התבוננו באלגוריתם הבא להכנסת איבר element למערך ממוין:

1. השם ב-i את מספר האיברים הממוינים במערך פחות אחת
2. כל עוד האיבר במקום ה-i גדול מ-element בצע
   2.1. הצמ את האיבר מקום אחד ימינה
   2.2. הקטן את i ב-1
3. הכנס את element במקום שהתפנה

שאלה 12.27

הכניסו את הערך 6 למערך הנתון, לפי האלגוריתם שתואר למעלה:

5	8	20	22	25		

חשבו: מה יקרה כאשר האיבר להכנסה יהיה קטן יותר מכל איברי המערך? **האלגוריתם לא יעצור ותהיה גלישה מהמערך!** כדי למנוע מצב כזה נשנה את תנאי הכניסה ללולאה כך שיכיל גם את התנאי "אם הגענו לקצה המערך". התנאי החדש בשורה השנייה של האלגוריתם יהיה:

2. כל עוד i >= 0 ו*גם* האיבר במקום ה-i גדול מ-element בצע

תרגיל 6

מטרת הבעיה ופתרונה: הצגת מחלקה ובה הכנסת איבר למערך ממוין.

המורה להיסטוריה צופיה, החליטה לשמור את רשימת תלמידיה בצורה ממוינת לפי ציונם בהיסטוריה. כך אם היא תידרש לאתר את התלמידים המצטיינים בעבור האולימפיאדה בהיסטוריה, תוכל לאתרם בקלות. עליכם לבנות מערכת שתעזור למורה צופיה לנהל את ציוני תלמידיה. לשם כך עליכם להגדיר מחלקת "תלמיד" המכילה את התכונות: שם תלמיד וציונו בהיסטוריה, ומחלקת "מורה" המגדירה מערך של תלמידים הממוין לפי ציוניהם, ופעולה המאפשרת להכניס תלמיד למערך. מערך התלמידים יישאר ממוין גם לאחר הכנסה של תלמידים נוספים. המערכת תאפשר לצופיה להוסיף תלמידים, ולשלוף אינפורמציה שלהם לפי שם תלמיד.

הגדרת המחלקה *Student*

הגדרת התכונות

- **name** – שם התלמיד, מטיפוס מחרוזת.
- **historyGrade** – ציונו של התלמיד בהיסטוריה, מטיפוס ממשי.

הגדרת הפעולות

- **פעולה בונה** – הפעולה מקבלת שם וציון, ומעדכנת את תכונות המחלקה בהתאם.
- **getName** – פעולה המאחזרת את שם התלמיד.
- **getHistoryGrade** – פעולה המחזירה את ציונו של התלמיד.

מימוש המחלקה

```
/*
        Student המחלקה
*/
public class Student
{
    private double historyGrade;
    private String name;
    // פעולה בונה
    public Student(String name, double historyGrade)
    {
        this.historyGrade = historyGrade;
        this.name = name;
    }
    // פעולות גישה
    public double getHistoryGrade()
    {
        return historyGrade;
    }
    public String getName()
    {
        return name;
    }
}
```

הגדרת המחלקה *Teacher*

הגדרת התכונות

- **students** – מערך לשמירת התלמידים, יישאר ממוין לאחר כל הכנסה.
- **numOfStudents** – מספר התלמידים השמורים אצל המורה עד עכשיו (במערך students).

הגדרת הפעולות

- **פעולה בונה** – הפעולה תקצה זיכרון עבור מערך התלמידים, ותאתחל את מספר התלמידים ב-0 (בהתחלה אין תלמידים במערך).
- **insertStudent** – פעולה המקבלת "תלמיד" ומכניסה אותו למקום המתאים במערך. הפעולה תעדכן את מספר התלמידים. פעולה זו תמומש לפי האלגוריתם שהוצג לעיל להכנסת איבר למערך ממוין. הנחות הכרחיות: יש מקום במערך. המערך ממוין.
- **getStudents** – פעולה המחזירה את מערך התלמידים.
- **getStudentByName** – פעולה המקבלת שם תלמיד ומחזירה את התלמיד בשם המבוקש.

מימוש המחלקה

```
/*
       Teacher המחלקה
*/
public class Teacher
{
    final int NUM_OF_STUDENTS = 40;
    private Student[] students;
    private int numOfStudents;
    // פעולה בונה
    public Teacher()
    {
        students= new Student [NUM_OF_STUDENTS];
        numOfStudents=0;
    }
    public Student[] getStudents()
    {
        Student[] studs = new Student[numOfStudents];
        for(int i = 0; i < numOfStudents; i++)
            studs[i] = new Student(students[i].getName(),
                                   students[i].getHistoryGrade());
        return studs;
    }
    public Student getStudentByName(String name)
    {
        for(int i = 0; i < numOfStudents; i++)
            if (students[i].getName().equals(name))
                 return new Student(students[i].getName(),
                                   students[i].getHistoryGrade());
        return null; // לא נמצא תלמיד בשם זה
    }
    public void insertStudent(Student stud)
    {
        int i = numOfStudents - 1;
        while (i >= 0 &&
               students[i].getHistoryGrade() > stud.getHistoryGrade())
        {
                students[i+1] = students[i];
                i--;
        }//while
        students[i+1] = new Student(stud.getName(),
                                          stud.getHistoryGrade());
        numOfStudents++;
    }
}
```

סוף התרגון בציה 6

שאלה 12.28

א. כמה דוגמאות מייצגות שונות של המשתנים `numOfStudents, stud` ו-`students` כדאי לבדוק כדי להשתכנע בנכונות הפעולה `insertStudent`? הראו דוגמאות כאלה.

ב. כמה פעמים תתבצע הלולאה עבור כל דוגמה מייצגת שבחרתם?

ג. בחרו שתי דוגמאות מייצגות מסעיף א והראו באמצעות איור, את המערך `students` לאחר כל ביצוע-חוזר של הלולאה, ובסיום ביצוע הלולאה.

ד. כמה פעמים תתבצע הלולאה עבור כל דוגמה מייצגת שבחרתם?

שאלה 12.29

כתבו פעולה ראשית לבעיה 6. הפעולה הראשית תיצור עצם מסוג `Teacher` שאליו יוכנסו תלמידים באמצעות הפעולה `insertStudent`. לאחר מכן קלטו שם של תלמיד וחפשו אותו באמצעות הפעולה `getStudentByName` והציגו את ציון התלמיד. לבסוף זמנו את הפעולה `getStudents` לקבלת המערך הממוין והציגו את רשימת התלמידים שציוניהם מעל ל-80.

מיון הכנסה במערך שלם

אנו חוזרים לאלגוריתם למיון הכנסה של מערך A. ניתן להתייחס לאיבר הראשון של המערך שנמצא ב-A[0] כאל מערך ממוין באורך 1. האלגוריתם מתעלם משאר איברי המערך ומכניס למקום הנכון את הערך שנמצא ב-A[1]. כתוצאה מכך מתקבל מערך ממוין באורך 2 המורכב מ-A[0] ומ-A[1] . הצעד הבא: הכנסת הערך שנמצא ב-A[2] למקום הנכון וכן הלאה. להלן מערך A לפני תחילת המיון. סימנו באפור את קטע המערך שכבר מוין:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
40	35	15	38	42	17	39

התרשימים שלהלן מראים את המערך לאחר הכנסת האיברים שבמקומות A[1], A[2], A[3], A[4], לפי הסדר. שימו ♥ שהאיבר A[4] נמצא במקומו כבר לאחר הכנסת A[3], ואינו זז כתוצאה מפעולה ההכנסה.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
35	**40**	15	38	42	17	39

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
15	**35**	**40**	38	42	17	39

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
15	**35**	**38**	**40**	42	17	39

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
15	**35**	**38**	**40**	**42**	17	39

נסכם ונאמר שהאלגוריתם של מיון הכנסה מניח שקטע המערך בין A[0] ל-A[k] ממוין, ומכניס למקום הנכון את האיבר שנמצא ב-A[k+1]. כתוצאה מכך מתקבל מערך ממוין בין A[0] ל-A[k+1]. הכנסת האיבר ה-A[k+1] למערך הממוין תיעשה באמצעות האלגוריתם "הכנס איבר למערך ממוין" שכבר פיתחנו בבעיה 6. בכל הכנסה למערך, אנו נעזרים במשתנה המציין את מספר האיברים הממוינים במערך. ערכו של משתנה זה גדל באחת לאחר כל הכנסה.

אם נרצה לכתוב פעולה הממיינת את המערך כולו באמצעות פעולת ההכנסה, נעבור על המערך מהאיבר השני ואילך, כל איבר יישלח בתורו לפעולת ההכנסה, והיא תכניס אותו למקומו בחלק הממוין של המערך. כך נעשה בפתרון הבעיה הבאה.

תרגיל 7

מטרת הבעיה ופתרונה: הצגת אלגוריתם למיון הכנסה.

הגדירו מחלקה ובה מערך של מספרים שלמים. הגדירו במחלקה פעולה בשם **insertionSort** הממיינת את המערך באמצעות האלגוריתם למיון הכנסה. המחלקה תשתמש בפעולה פרטית "הכנס איבר למערך ממוין".

הגדרת המחלקה *SortedArray*

הגדרת התכונות

- **numbers** – מערך של מספרים שלמים.
- **sorted** – מספר האיברים הממוינים במערך עד כה.

הגדרת הפעולות

- **פעולה בונה** – הפעולה מקבלת מערך חד-ממדי, ומעדכנת את המערך numbers בהתאם ומאתחלת ב-1 את מספר האיברים הממוינים (מכיוון שהאיבר הראשון ממוין ביחס לעצמו בלבד).
- **insertElement** – פעולה פרטית המקבלת איבר להכנסה, ומכניסה את האיבר בחלק הממוין של המערך, כך שהמערך יישאר ממוין. הפעולה מקדמת ב-1 את מספר האיברים הממוינים במערך.
- **insertionSort** – פעולה הממיינת את המערך numbers לפי מיון הכנסה. (בשימוש ב-**insertElement**)
- **getNumbers** – פעולת גישה המחזירה את המערך הממוין.

מימוש המחלקה

```
/*
      SortedArray המחלקה
*/
public class SortedArray
{
    private int[] numbers;
    private int sorted;
    // פעולה בונה
    public SortedArray(int[] a)
    {
        numbers = a;
        sorted = 1;
    }
    // הכנס למערך ממוין, פעולה פרטית
    private void insertElement(int element)
    {
        int i = sorted - 1 ;
        while (i >= 0 && numbers[i] > element)
        {
```

```
            numbers[i+1] = numbers[i];
            i--;
        }//while
        numbers[i+1] = element;
        sorted++;
    }
    // מיון הכנסה
    public void insertionSort()
    {
        for(int i = 1; i < numbers.length; i++)
            insertElement(numbers[i]);
    }
    public int[] getNumbers()
    {
       return numbers;
   }
}// class SortedArray
```

יעילות

שימו ♥: אם N הוא מספר איברי המערך, כמה פעמים מתבצע גוף הלולאה הפנימית במיון הכנסה (הלולאה בפעולה `insertElement`)? החישוב דומה לחישוב שעשינו עבור מיון בחירה. גם כאן יש סדרה חשבונית עולה: בזימון הראשון ל-`insertElement` ייתכן שיש להזיז איבר אחד במערך, בזימון השני שני איברים וכן הלאה עד שבזימון האחרון i=N-1 וייתכן שיש להזיז את כל N-1 האיברים. במקרה הגרוע מיון הכנסה דורש בערך N^2 ביצועים של גוף הלולאה, וראינו שמספר זה גדל במהירות ככל ש-N גדל.

נשווה מיון הכנסה למיון בחירה לפי סוגי קלט שונים. למיון בחירה יש יתרון כאשר האיברים הראשונים במערך גדולים, כי בכל מעבר בלולאה יש בדיוק החלפה אחת של איברים. לעומת זאת במיון הכנסה האיברים הגדולים "זוחלים" לאט לאט למקומם. היתרון של מיון הכנסה בולט כאשר מנסים למיין מערך שכבר ממוין או שכמעט ממוין. במקרה זה הלולאה בפעולה `insertElement` תתבצע מספר מועט של פעמים, כי הערך `element` יהיה כמעט תמיד גדול מהאיברים בחלק הממוין של המערך ולכן לא ניכנס ללולאה המזיזה איברים ומפנה לו מקום.

סוף התרגול מציה 7

שאלה 12.30

תארו מה קורה במיון הכנסה כאשר מנסים למיין מערך ממוין.

שאלה 12.31

אפיינו את המערך הגורם לביצועים הגרועים ביותר במיון הכנסה.

שאלה 12.32

מה צריך לשנות במחלקה SortedArray כדי לקבל מיון הכנסה בסדר יורד?

שאלה 12.33

נתון מערך לא ממוין בגודל N, עליכם לבדוק כמה ערכים שונים יש במערך.

הדרכה: הגדירו מחלקה המכילה מערך לא ממויין כתכונה. כתבו פעולה הממיינת את המערך וסופרת תוך כדי המיון את מספר הערכים השונים במערך. הפעולה תחזיר את המספר שנמצא.

שאלה 12.34

נתון מערך לא ממוין בגודל N המכיל מספרים החוזרים על עצמם. עליכם למיין את המערך ולגרום לכך שכל מספר יופיע בו פעם אחת בלבד.

הדרכה: הגדירו מחלקה המכילה מערך לא ממוין כתכונה. כתבו פעולה הממיינת את המערך במיון הכנסה, ללא הכנסת מספרים חוזרים. הוסיפו פעולה המחזירה את המערך הממוין.

מיון בועות

מיון בועות הוא מיון המתבסס על השוואת כל זוג ערכים צמודים, ומחליף ביניהם אם הם לא מסודרים בסדר המבוקש. כלומר אם נרצה למיין בסדר עולה, נשווה תחילה את האיבר הראשון לשני ונחליף ביניהם אם הראשון גדול מהשני. לאחר מכן, נשווה את האיבר השני לשלישי ונחליף ביניהם אם צריך. כך נמשיך עד לזוג האחרון במערך. כתוצאה ממעבר כזה על כל הזוגות הצמודים, האיבר בעל הערך הגבוה ביותר יגיע למקומו – לסוף המערך.

מעבר נוסף על כל הזוגות הצמודים יגרום לאיבר השני בגודלו להגיע למקומו – מקום אחד לפני האיבר האחרון. בכל מעבר, איבר אחד יגיע למקומו. חשבו כמה מעברים צריכים לבצע כדי שכל האיברים יגיעו למקומם?

הנה אלגוריתם הממיין מערך של מספרים שלמים לפי הרעיון הנתון:

```
עבור כל i מ-1 עד A.length-1:
    עבור כל j מ-0 עד A.length-i-1:
        אם a[j] > a[j+1]
            החלף את האיבר A[j] ואת האיבר A[j+1]
```

מיון בועות נקרא כך כיוון שבכל שלב "מבעבעים" את האיבר הגדול למקומו.

שאלה 12.35

לפניכם קטע תוכנית הממיין בשיטת "מיון בועות" את המערך A ובו מספרים שלמים. השלימו את החלקים החסרים במיון: (שימו ♥: המערך צריך להיות ממוין בסדר עולה).

```
int temp;
for (int i = 1 ; i < ____________; i++)
{
      for (int j = 0 ; j < _____________; j++)
      {
            if (A[____] > A[____])
            {
                  temp = ___________;
                  A[____] = ___________;
                  A[____] = _________;
            }
      }
}
```

12.4 מיזוג

בסעיף זה נלמד אלגוריתם לבעיית המיזוג.

> **מיזוג** הוא שילוב של קלט משני מקורות לפלט יחיד.
>
> הקלטים ממוינים והפלט חייב להיות ממוין גם הוא.

נניח שברשותנו שני מערכים ממוינים A ו-B המכילים ערכים מסוג מסוים (תווים, מספרים שלמים, ממשיים או כל טיפוס סדור אחר). נרצה למזג את שני המערכים במערך C. האלגוריתם יסרוק במקביל את המערך A באמצעות המציין iA ואת המערך B באמצעות המציין iB, ויבחר את האיבר המתאים (מבין האיבר ב-A והאיבר ב-B) ויכניס אותו למערך C. המשתנה iC יציין את המיקום שאליו צריך להכניס את האיבר הבא במערך C.

חשבו: מה צריך להיות גודל המערך C אם גודלם של המערכים A ו-B הוא A.length ו-B.length?

כך נראים המערכים A, B ו-C לפני תהליך המיזוג (בדוגמה זו המערכים מכילים איברים מטיפוס תווי):

A:

A[0]	A[1]	A[2]	A[3]
F	G	K	R

iA

B:

B[0]	B[1]	B[2]	B[3]	B[4]
J	L	M	T	X

iB

C:

C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]	C[8]

iC

כדי שהמערך C יהיה ממוין בסוף תהליך המיזוג נבחר בכל שלב להכניס את האיבר הקטן מבין האיברים שב-A וב-B. להלן תרשים של המערכים לאחר הצעד הראשון המעתיק את האיבר הראשון של A לאיבר הראשון של C. שימו לב לקידום המציינים.

A[0]	A[1]	A[2]	A[3]
F	G	K	R

iA

B[0]	B[1]	B[2]	B[3]	B[4]
J	L	M	T	X

iB

C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]	C[8]
F								

iC

המערכים לאחר הצעד השני:

A[0]	A[1]	A[2]	A[3]
F	G	K	R

iA

B[0]	B[1]	B[2]	B[3]	B[4]
J	L	M	T	X

iB

C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]	C[8]
F	G							

iC

ולאחר הצעד השלישי:

A[0]	A[1]	A[2]	A[3]
F	G	K	R

iA

B[0]	B[1]	B[2]	B[3]	B[4]
J	L	M	T	X

iB

C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]	C[8]
F	G	J						

iC

שאלה 12.36

ציירו את המערך C ואת המציין iC אחרי הצעד הרביעי באלגוריתם ואחרי הצעד השישי.

הביצוע החוזר יסתיים כאשר באחד משני המערכים לא יישארו איברים. ייתכן כי במערך השני נשאר "זנב": איברים שלא נכנסו למערך C, ויש להעתיקם למערך C.

נכתוב את האלגוריתם למיזוג מערך:

```
1.  כל עוד נשארו איברים ב-A ובמ ב-B בצע
    1.1  אם האיבר ב-A קטן מהאיבר ב-B אז
         1.1.1  הצמק את האיבר מ-A ל-C
         1.1.2  הבא את המצ'ין של A ב-1
    1.2  אחרת
         1.2.1  הצמק את האיבר מ-B ל-C
         1.2.2  הבא את המצ'ין של B ב-1
    1.3  הבא את המצ'ין של C ב-1
2.  כל עוד נשארו איברים ב-A בצע          // "זנב" של A – אם יש
    2.1  הצמק את האיבר מ-A ל-C
    2.2  הבא את המצ'ין של A ב-1
    2.3  הבא את המצ'ין של C ב-1
3.  כל עוד נשארו איברים ב-B בצע          // "זנב" של B – אם יש
    3.1  הצמק את האיבר מ-B ל-C
    3.2  הבא את המצ'ין של B ב-1
    3.3  הבא את המצ'ין של C ב-1
```

שימו ♥: רק אחת מבין שתי הלולאות שבשורות 2 ו-3 תתבצע. לעולם לא שתיהן. חשבו מדוע.

שאלה 12.37

עקבו אחר ביצוע האלגוריתם שתיארנו עבור המערכים A ו-B הבאים:

A[0]	A[1]	A[2]	A[3]
12	27	70	93

B[0]	B[1]	B[2]	B[3]	B[4]
12	27	70	93	57

שאלה 12.38

מהו ה"זנב" המתקבל ממיזוג המערכים בשאלה הקודמת?

שאלה 12.39

לפניכם קטע תוכנית בשפת Java, למיזוג המערכים A ו-B במערך C, השלימו את הקטע במקומות החסרים:

```
int iA = _______;
int iB = _______;
int iC = _______;
while (iA < A.length && iB < B.length)
{
    if(A[iA] ___ B[iB])
    {
        C[____] = A[____];
        ____++;
    }
    else
    {
        C[____] = B[____];
        ____++;
    }
    ____++;
}
while (_________________)
{
    C[iC] = A[iA];
    iA++;
    iC++;
}
while (_________________)
{
    C[iC] = B[iB];
    iB++;
    iC++;
}
```

שאלה 12.40

תנו דוגמאות של ערכים התחלתיים במערכים A ו-B שעבורן:

א. ערכו של iA נשאר 0 לאורך ביצוע לולאת ה-while הראשונה.

ב. ערכו של iB נשאר 0 לאורך ביצוע לולאת ה-while הראשונה.
ג. ערכי iA ו-iB עולים לסירוגין: ערכו של iA עולה ב-1, אחר כך ערכו של iB עולה ב-1, אחר כך ערכו של iA עולה ב-1, וכן הלאה עד ל"זנב".

שאלה 12.41

כמה פעמים מתבצעות הלולאות במיזוג? חשבו את המספר הכולל של הלולאות, לרבות ה"זנבות".

שאלה 12.42

נתונים שני מערכים ממוינים sortedA ו-soredtB. כתבו קטע תוכנית בשפת Java שיציג את ערכו של האיבר הקטן ביותר המופיע בשני המערכים.
למשל עבור המערכים:

sortedA = 1 ,5 ,6 ,8 ,10
sortedB = 2 ,4 ,6 ,10 ,20

יודפס הערך: 6

שאלה 12.43

"רשימות זרות" הן שתי רשימות ערכים שאין להן אף ערך משותף. כתבו קטע תוכנית הבודק אם שני המערכים הממוינים A ו-B הם רשימות זרות.

שאלה 12.44

נתונים שני מערכים ממוינים, אחד בסדר עולה והשני בסדר יורד. כתבו קטע תוכנית הממזג את שני המערכים כך שהמערך הממוזג יהיה ממוין בסדר עולה.

סיכום

בסעיף 12.1 הוצגו בעיות שפתרונן מצריך שמירה של ערכים מאותו טיפוס בצורה של טבלה. ניתן לשמור ערכים אלה כמערך דו-ממדי.

- **גישה לאיבר במערך דו-ממדי** מתבצעת באמצעות: שם מערך ושני מציינים (אינדקסים), הראשון מציין את מספר השורה והשני מציין את מספר האיבר בשורה (כלומר את מספר העמודה).
- **סריקה מלאה** של כל איברי המערך מתבצעת באמצעות לולאה מקוננת, כלומר לולאה בתוך לולאה. הלולאה החיצונית עוברת על שורות המערך והפנימית על כל האיברים בשורה.
- **מטריצה ריבועית** היא מערך דו-ממדי שמספר השורות שלו שווה למספר העמודות שלו.
- **אלכסון ראשי** במטריצה ריבועית הוא אוסף כל האיברים שמספר השורה שלהם שווה למספר העמודה שלהם.
- **אלכסון משני** במטריצה ריבועית הוא אוסף כל האיברים שסכום ערכי השורה והעמודה שלהם שווה למספר השורות פחות 1.

בסעיפים 12.2, 12.3 ו-12.4 הוצגו אלגוריתם לחיפוש בינרי ושלושה אלגוריתמים למיון: מיון הכנסה, מיון בחירה ומיון בועות והוצג אלגוריתם למיזוג.

- **חיפוש בינרי** מתאים רק למערך ממוין, והוא יעיל בהרבה מהאלגוריתם הסדרתי לחיפוש מכיוון שהוא מצמצם את טווח החיפוש בחצי בכל ביצוע של הלולאה.

- **במיון בחירה** מחפשים את הערך הקטן ביותר בקטע המערך שטרם מוין ומחליפים בינו ובין האיבר שנמצא במיקום הראשון בקטע. עבור מערך בגודל N מתבצעת הלולאה N^2 פעמים ללא תלות בערכים ההתחלתיים של איברי המערך.
- האלגוריתם ל**מיון הכנסה** פועל בצורה דומה להכנסת קלף בצורה ממוינת ליד האוחזת בקלפים. גם כאן במקרה הגרוע מיון הכנסה דורש בערך N^2 ביצועים של גוף הלולאה.
- **מיון בועות** פועל על עיקרון של החלפת זוגות. בכל שלב "מבעבעים" את האיבר הגדול (או הקטן) למיקומו המתאים במערך.
- **מיזוג** הוא פעולה למיזוג של שני מערכים ממוינים במערך אחד ממוין. בכל שלב מכניסים למערך הממוזג את האיבר הקטן ביותר מבין האיברים שבמערכי הקלט. בסוף התהליך יכול להישאר "זנב" באחד ממערכי הקלט, יש להעתיק אותו למערך הממוזג.

סיכום מרכיבי שפת Java שנלמדו בפרק 12

- **הצהרה על מערך** דו-ממדי בשפת Java נכתבת כך:

```
טיפוס [][] שם המערך;
```

ההצהרה מורכבת משם הטיפוס, אחריו **שני זוגות** של סוגריים מרובעים (המציינים הצהרה על מערך דו-ממדי) ולבסוף שם המערך.

- לפני שניתן להשתמש במערך יש לבצע עבורו **הקצאת זיכרון**. הקצאת הזיכרון מתבצעת באמצעות ההוראה **new**:

```
שם המערך = new טיפוס [מספר שורות][מספר עמודות];
```

ניתן לשלב את ההצהרה ואת ההקצאה בהוראה אחת:

```
טיפוס [][]שם המערך = new טיפוס [מספר שורות][מספר עמודות];
```

- **מספור האיברים במערך** דו-ממדי מתחיל בשורה 0 ובעמודה 0, לכן `mat[0][1]` מפנה לאיבר **השני** בשורה **הראשונה** במטריצה ששמה mat.
- לכל מערך דו-ממדי מוגדרות **תכונות אורך** בשם **length**, המציינות את מספר השורות במערך ואת מספר האיברים בכל שורה. למשל, `mat.length` מציין את מספר השורות במטריצה `mat`. ואילו `mat[i].length`, מציין את מספר האיברים בשורה i.
- יש להקפיד על פנייה לאיבר באמצעות מציין שערכו איננו חורג מתחום הערכים המותר, כלומר, מציין של שורה נע בין 0 לבין מספר השורות פחות 1 ומציין של איבר בשורה נע בין 0 לבין מספר האיברים בשורה פחות 1. **חריגה מגבולות המערך** תגרום לשגיאה בזמן ריצה.

שאלות נוספות

שאלות נוספות לסעיף 12.2

1. נתון מערך A בגודל N, אשר ערכיו הראשונים הם 0 והערכים שאחריהם הם 1. כתבו קטע תוכנית המוצא את מציין האיבר האחרון שערכו הוא 0 (ניתן להניח שיש לפחות 0 אחד).

2. כתבו תוכנית המממשת את המשחק "חם קר". שחקן המשחק נגד התוכנית בוחר מספר שלם K חיובי בין 0 ל-N נתון. על התוכנית לגלות את המספר הנבחר במינימום ניחושים. ניחוש הינו מספר שלם בין 0 ל-N ותשובת השחקן לניחוש היא: "חם יותר" אם המרחק (**בערך מוחלט**) בין הניחוש הנוכחי ל-K גדול יותר מן המרחק בין הניחוש הקודם ל-K, "קר יותר" אם ההיפך, ו-"אין שינוי" אם המרחק לא השתנה.

3. נתון מערך ממוין A בגודל N ובו ערכים חוזרים. כתבו קטע תוכנית שהפלט שלו הוא הודעה אם יש ערך המופיע יותר מ-N/2 פעמים.

4. נתון מערך ממוין ובו מספרים שלמים בתחום 100..200 בלבד, כל ערך יכול להופיע יותר מפעם אחת. עליכם לכתוב קטע תוכנית בשפת Java שיציג על המסך אילו מבין הערכים בין 100 ל-200 **לא** נמצאים במערך.

5. נתון מערך A בגודל N והוא "ממוין מעגלית"; כלומר מאיבר מסוים בו שאינו בהכרח האיבר הראשון, הוא ממוין בסדר עולה כאשר מתבוננים בו בצורה מעגלית (למשל המערך A שאיבריו הם: 5 7 9 1 2 3 5). כתבו קטע תוכנית המוצא את מציין האיבר הראשון שה"מיון המעגלי" מתחיל ממנו.

6. נתון מערך ממוין A בגודל N והוא מורכב משתי סדרות עולות, כך שהשנייה המתחילה עם סיום הראשונה היא קצרה יותר, ומהווה רישה של הראשונה (כלומר היא זהה לתחילת הראשונה). כתבו קטע תוכנית המוצא את מציין האיבר הראשון בסדרה השנייה.

שאלות נוספות לסעיף 12.3

1. נתון מערך (לא ממוין) A בגודל N ובו ערכים שונים זה מזה. כתבו קטע תוכנית המסדר את המערך בצורת "זיג-זג", כלומר איברי המערך A יסודרו כך שעבור כל ערך שנמצא במקום זוגי במערך – הערך שלפניו קטן ממנו והערך שאחריו גם קטן ממנו.

הדרכה: בצעו מיון **בחירה** שבו תבחרו לסירוגין את המספר הקטן ביותר ולאחר מכן את המספר הגדול ביותר.

2. נתון מערך לא ממוין A בגודל N זוגי. כתבו קטע תוכנית שהפלט שלו הוא חלוקה של N המספרים לזוגות כך שהסכום המרבי מבין סכומי הזוגות הוא מינימלי.

3. נתון מערך (לא ממוין) ובו N ערכים. כתבו קטע תוכנית שהפלט שלו הוא הודעה אם סדרת ההפרשים של ערכי הרשימה הנתונה היא סדרה יורדת. (סדרת הפרשים היא הסדרה הנוצרת מההפרש בין כל זוג ערכים סמוכים בסדרה המקורית).

4. מיון נקרא "יציב" אם הסדר היחסי בין ערכים **זהים** במערך המקורי הוא גם הסדר ביניהם במערך הממוין. לדוגמה, אם במערך המקורי הערך 4 מופיע פעמיים - (למשל במקום השני ובמקום השמיני) ולאחר המיון, ה-4 הראשון מופיע לפני ה-4 השני המיון ייקרא מיון "יציב". לעומת זאת אם במערך הממוין, הסדר ביניהם התהפך, כלומר ה-4 השני מופיע לפני ה-4 הראשון אז המיון "אינו יציב".

ציין והסבר: האם מיון בחירה הוא יציב? האם מיון הכנסה הוא יציב? האם מיון בועות הוא יציב?

5. נתון מערך באורך N, וערך כל אחד מאיבריו הוא 0 או 1. כתבו קטע תוכנית שהפלט שלו הוא מספר ההחלפות המינימלי הדרוש כך שכל ה-1 יהיו מימין לכל ה-0. ההחלפה אינה חייבת להיעשות בין ערכים סמוכים.

שאלות נוספות לסעיף 12.4

1. הניחו שאין ערכים זהים בתוך מערך A ושאין ערכים זהים בתוך מערך B, אך ייתכן שיש ערכים הקיימים גם ב-A וגם ב-B. שנו את פעולת **המיזוג** כך שערכים זהים יופיעו פעם אחת בלבד במערך הממוזג.

2. נתונים שני מערכים ממוינים ובהם מספרים שונים. כתבו קטע תוכנית שיבדוק אם בכל עשיריית מספרים עוקבים במערך **הממוזג** קיים לפחות נציג אחד מכל אחד ממערכי המקור. קטע התוכנית יציג הודעה מתאימה.

3. נתונים שני מערכים ממוינים. כתבו קטע תוכנית המייצר את המערך **הממוזג** הארוך ביותר האפשרי המתחיל במספר הקטן ביותר (מבין שני המערכים), וכל מספר נוסף במערך הממוזג יהיה גדול לפחות ב-10 מקודמו.

4. נתונים שני מערכים ממוינים ובהם יש ערכים חוזרים. כתבו קטע תוכנית המוצא ומדפיס את הערך המופיע מספר מרבי של פעמים בשתי הרשימות יחדיו (אם יש יותר מאחד כזה, אזי נדפיס את אחד מן הערכים האלה).

5. נתונות שתי רשימות ממוינות. כתבו קטע תוכנית שהפלט שלו הוא הרשימה הממוזגת **הארוכה ביותר** האפשרית המתחילה במספר הקטן ביותר מבין שתי הרשימות ומקיימת את החוק הבא: כל מספר נוסף ברשימה הוא מעשיריית המספרים העוקבת לעשיריית המספרים של קודמו. למשל, אם המספר הראשון הוא 17 אזי המספר הבא אחריו יכול להיות בין 20 ל-29.

6. נתונים שלושה מערכים ממוינים. בתוך כל מערך יש ערכים השונים זה מזה. כתבו קטע תוכנית שהפלט שלו הוא רשימה ממוינת המתקבלת ממיזוג המערכים הנתונים, ללא חזרה של ערכים.