פרק 10 – מערכים

האלגוריתמים שפיתחנו לפתרון בעיות שונות בפרקים הקודמים היו שונים ומגוונים. הם היו שונים זה מזה בפרט בכמות המידע שנקלט בהם, כלומר בגודל הקלט. אבל בכל האלגוריתמים שהצגנו עד כה, גם כאשר כמות המידע הנקלט הייתה גדולה, הרי כמות המידע שהיה צריך **לשמור** או **לזכור** במהלך ביצוע האלגוריתם הייתה קטנה. למשל כאשר רצינו לחשב ממוצע של 100 מספרי קלט, שמרנו אך ורק את סכומם המצטבר ואת נתון הקלט התורן, ולא את כל 100 נתוני הקלט. בכל האלגוריתמים שפיתחנו עד כה השתמשנו במספר משתנים מצומצם, ולכל משתנה הוגדר תפקיד ייחודי משלו.

בפרק זה יוצגו בעיות אשר לצורך פתרונן יש לשמור מספר גדול של נתונים שיש קשר ביניהם: הם בעלי משמעות דומה וניתנים לתיאור כאוסף סדור של איברים מאותו טיפוס. אוסף סדור כזה של איברים נקרא "מערך".

10.1 מערך ואיברי מערך

בעיה 1

מטרת הבעיה ופתרונה: הצגת שימוש במערך לפתרון בעיה אלגוריתמית.

פתחו אלגוריתם המקבל כקלט את זמני ההקפה של כל אחד מארבעים משתתפי מרוץ הקרטינג. האלגוריתם מציג כפלט את מספר המשתתפים שביצעו הקפה בזמן נמוך מהזמן הממוצע של כלל המשתתפים. ישמו את האלגוריתם בשפת Java.

ניתוח הבעיה בעזרת דוגמאות

שאלה 10.1

מהו הפלט כאשר נתונים 40 זמני הקפה, עשרים מהם שווים ל-4.50, עשרה שווים ל-4.20, ועשרה שווים ל-4.60?

פירוק הבעיה לתת-משימות

בפרק 7 כבר ראינו כיצד לחשב ממוצע של רשימת ערכי קלט:

1. קליטת נתוני הקלט וצבירתם
2. חלוקת הסכום המצטבר במספר ערכי הקלט

? במהלך צבירת הנתונים לצורך חישוב הממוצע נוסף לצובר ערכו של כל נתון שנקלט, אך הנתון איננו נשמר. לאחר קליטת כל נתוני הקלט ניתן לחשב את הממוצע, אך ערכי הקלט עצמם אינם שמורים. אבל כדי למנות את מספר הנתונים שמתחת לממוצע יש **לשמור** את ערכי הקלט עד לאחר חישוב הממוצע, משום שרק לאחר חישוב הממוצע, אפשר להשוות כל אחד מערכי הקלט לממוצע. כיצד נרחיב את התת-משימות המתוארות כך שיתייחסו גם למציאת ערכי הקלט הקטנים מהממוצע?

1. קליטת נתוני הקלט, **שמירתם** וצבירתם
2. חלוקת הסכום המצטבר במספר ערכי הקלט
3. השוואת כל ערך קלט לממוצע, ומנייתו אם הוא קטן מהממוצע

בחירת משתנים

יש 40 נתוני קלט, ועלינו לשמור כל אחד מהם בנפרד. האם נצהיר על 40 משתנים, כל אחד בנפרד?

מכיוון שלכל אחד מנתוני הקלט אפיון דומה, אפשר לקשרם יחד. ניתן להתייחס אל נתוני הקלט כאל סדרת ערכים בת 40 איברים דומים, ולפיכך לשמור אותם בסדרת משתנים. הערך שנקלט ראשון יישמר במשתנה הראשון בסדרה, הערך שנקלט שני יישמר במשתנה השני בסדרה וכך הלאה. הקישור בין המשתנים יתבצע על ידי מתן שם לסדרה ופנייה לכל אחד מהמשתנים לפי מיקומו הסידורי בסדרה. סדרה כזאת של משתנים הקשורים זה לזה נקראת **מערך**.

מערך (array) הוא אוסף סדור של איברים מאותו טיפוס. ניתן להתייחס לכל אחד מאיברי המערך כמו למשתנה לכל דבר. כלומר ניתן לשמור בו ערכים ולקרוא את הערכים השמורים בו. מיקומו הסידורי של איבר במערך מצוין ב**מציין** (index).

אם כך בבעיה זו אנו זקוקים למערך בן ארבעים איברים, איבר אחד עבור כל אחד מארבעים זמני ההקפה הנתונים. הטיפוס של כל איבר יהיה ממשי. למערך לשמירת זמני ההקפה ניתן את השם scores.

לכן זוהי רשימת המשתנים שנזדקק לה:

scores – מערך של 40 איברים ממשיים, לשמירת כל זמני ההקפה
sumOfScores – מטיפוס ממשי, ישמור את סכום זמני ההקפות הנתונים בקלט
averageScore – ממוצע זמני ההקפה, מטיפוס ממשי
belowAverageCounter – למניית מספר הזמנים שערכם הוא מתחת לממוצע, מטיפוס שלם

בשפת Java ניתן להגדיר מערך שאיבריו הם מכל טיפוס נתונים שהוא, למשל, מערך שאיבריו הם מטיפוס שלם, או מערך שאיבריו הם מטיפוס תו. לצורך פתרון בעיה זו אנו נדרשים להגדיר מערך מטיפוס ממשי, כלומר שאיבריו הם מטיפוס ממשי. ההצהרה על מערך של מספרים ממשיים נעשית באופן הבא:

```
double[] scores;
```

שימו ♥: ההצהרה דומה להצהרה על משתנה מטיפוס ממשי, ורק תוספת הסוגריים המרובעים (`[]`) מבהירה כי אין הכוונה כאן למשתנה יחיד מטיפוס ממשי אלא לעצם שהוא מערך שאיבריו הם מטיפוס ממשי.

אך ההצהרה אינה מספיקה. עלינו להקצות מקום בזיכרון עבור אוסף האיברים שבמערך באמצעות ההוראה **new**. כמובן בעת ההקצאה עלינו לציין את מספר האיברים במערך, וכך נקבע את גודל שטח הזיכרון שיש להקצות.

אם כך, הגדרת עצם שהוא מערך מטיפוס ממשי, והקצאת מקום בזיכרון עבורו (בציון מספר האיברים שהוא אמור להכיל) תיעשה כך:

```
double[] scores = new double[40];
```

כזכור, הפעולה **new** מחזירה הפניה לשטח הזיכרון שהוקצה. לכן למעשה `scores` מפנה כעת אל השטח שהוקצה עבורו בזיכרון.

נשתמש בקבוע על מנת להגדיר את מספר האיברים, וכך תיראה ההגדרה:

```
final int NUM_OF_RUNNERS = 40;
double[] scores = new double[NUM_OF_RUNNERS];
```

יש להדגיש כי בשפת Java, כאשר מוקצה שטח זיכרון עבור מערך של איברים מטיפוס פשוט, מתבצע גם אתחול אוטומטי של כל איבריו. איברי מערך מספריים (שלמים או ממשיים) מאותחלים ב-0, איברי מערך בוליאני מאותחלים ב-false, ואיברי מערך תווי מאותחלים בתו מיוחד שנקרא תו ריק.

בשפת Java האיבר הראשון במערך הוא במיקום 0 מתחילת המערך ולכן פנייה לאיבר הראשון תיעשה באמצעות המציין 0, כך: scores[0]. האיבר השני נמצא במיקום 1 מתחילת המערך ולכן פנייה אליו תיעשה באמצעות המציין 1, כך: scores[1]. האיבר האחרון ברשימה, איבר מספר 40, נמצא במיקום 39 מתחילת המערך ונפנה אליו בכתיבת scores[39]. באופן כללי, ציון איברי מערך בשפת Java מתחיל תמיד ב-0, ופנייה לאיבר הנמצא במיקום i מתחילת מערך בשם anArray נכתבת כך: anArray[i]. איבר זה הוא האיבר ה-i+1 ברשימת האיברים.

נמחיש את המערך scores בעזרת האיור הבא:

scores[0]	scores[1]		scores[19]		scores[39]
		…		…	

האלגוריתם

1. *אתחל את* sumOfScores *ב-0*
2. *אתחל את* belowAverageCounter *ב-0*
3. ***עבור כל*** i *שווה מ-0 עד מספר הציונים הנקלטים פחות 1* ***בצע:***
   3.1. *קלוט את ציון ההקפה הבא והשם באיבר ה-*i *במערך* scores
   3.2. *הוסף לערך השמור ב-*sumOfScores *את ציון ההקפה הנקלט*
4. *חלק את הערך השמור ב-*sumOfScores *במספר ציוני הקלט והשם ב-*averageScore
5. ***עבור כל*** i *שווה מ-0 עד מספר הציונים הנקלטים פחות 1* ***בצע:***
   5.1. ***אם*** *ציון שם האיבר ה-*i *במערך* scores ***גדול מ-***averageScore
      5.1.1. *הגדל את ערכו של* belowAverageCounter *ב-1*
6. *הדפס כפלט את ערכו של* belowAverageCounter

יישום האלגוריתם

לכל עצם שהוא מערך מוגדרת בשפה תכונה בשם `length` השומרת את גודל המערך, כלומר את מספר האיברים שהוא מכיל. לתכונה זו נוכל לגשת באמצעות סימון הנקודה, בדומה לאופן שבו הפעלנו פעולות על עצם. למשל `scores.length` היא תכונת האורך של המערך `scores`, וערכה שווה ל-40.

תכונת האורך של מערך היא קבועה ואינה ניתנת לשינוי. כלומר נוכל לשלב אותה בתוך ביטויים שונים בתוכנית, למשל `x = scores.length + 1`, אך לא נוכל להציבה בצד שמאל של הוראת השמה. כך למשל, הוראה כמו `scores.length = 3` היא שגויה. אם כך `scores.length` הוא בעצם קבוע לכל דבר (בדומה לקבוע `NUM_OF_RUNNERS`), אך הוא מקושר לעצם `scores`, וניתן לגשת אליו רק דרך העצם `scores` כפי שמביע סימון הנקודה.

שימו ♥: גם כדי להשתמש בפעולה על עצם וגם כדי לגשת לתכונה שלו אנו משתמשים בסימון הנקודה, אך הפנייה לפעולה תלווה תמיד בסוגריים (אולי ריקים, אם הפעולה אינה מצפה לקבל פרמטרים), ואילו פנייה לתכונה, בדומה לפנייה למשתנה רגיל, היא ללא סוגריים.

באלגוריתם לפתרון בעיה 1 כללנו הוראה לביצוע-חוזר מספר פעמים ידוע מראש, למעבר על כל איברי המערך בזה אחר זה. נוכל להשתמש בתכונת האורך של המערך הנסרק כדי לשלוט במספר הסיבובים בלולאה. כלומר נוכל לקבוע מראש את מספר הסיבובים בלולאה ל-scores.length.

מאחר שהתכונה length שומרת את מספר האיברים במערך, ניתן גם לומר שאיברי המערך נמצאים בו מהמקום 0 ועד המקום scores.length-1. אם כך, נאתחל את משתנה הבקרה של הלולאה ב-0, והלולאה תסתיים כאשר ערכו יגיע ל-scores.length-1 (אפשר כמובן גם לקבוע את ערך הסיום של משתנה הבקרה ל-NUM_OF_RUNNERS-1).

התוכנית המלאה

```
/*
קלט: זמני ההקפה של ארבעים משתתפי מרוץ קרטינג
פלט: מספר המשתתפים שביצעו הקפה בזמן נמוך מהממוצע
הקבוצתי
*/
import java.util.Scanner;
public class BelowAverage
{
    public static void main (String[] args)
    {
        Scanner in = new Scanner(System.in);
        // הגדרת קבוע
        final int NUM_OF_RUNNERS = 40;
        // הגדרת משתנים
        double[] scores = new double[NUM_OF_RUNNERS];
                                        // מערך זמני ההקפה
        double sumOfScores = 0;        // צובר זמני ההקפה
        double averageScore;           // ממוצע זמני ההקפה
        int belowAverageCounter = 0;  //  מונה
        // קלט וצבירה
 1.     for (int i = 0; i < scores.length; i++)
        {
 1.1.       System.out.print("Enter score: ");
 1.2.       scores[i] = in.nextDouble();
 1.3.       sumOfScores = sumOfScores + scores[i];
        } // for
        // חישוב ממוצע
 2.     averageScore = sumOfScores / scores.length;
        // מניית הנמוכים מהממוצע
 3.     for (int i = 0; i < scores.length; i++)
 3.1.       if (scores[i] < averageScore)
 3.1.1.           belowAverageCounter++;
        // פלט
 4.     System.out.println(belowAverageCounter +
                                    " participants are below average");
    } // main
} //class BelowAverage
```

מעקב

נעקוב באופן חלקי אחר מהלך ביצוע התוכנית עבור הקלט 4.20 ... 4.10 4.50
נניח שסכום זמני ההקפות הוא 168.8.

משפט לביצוע	i	Scores [0]	…	Scores [39]	sumOf Scores	scores[i] > average Score	average Score	below Average Counter	**פלט**
1	0	?		?	0		?	0	
1.1	0	?		?	0		?	0	Enter score:
1.2	0	4.50		?	0		?	0	
1.3	0	4.50		?	4.50		?	0	
1	1	4.50		?	4.50		?	0	
1.1	1	4.50		?	4.50		?	0	Enter score:
1.2	1	4.50		?	8.60		?	0	
1.3	1	4.50		?	8.60		?	0	
. . .	. . .	. . .	. . .	. . .	. . .		. . .	. . .	
1	49	4.50		?	164.6		?	0	
1.1	49	4.50		4.20	164.6		?	0	Enter score:
1.2		4.50			164.6		?	0	
1.3	49	4.50		4.20	168.8		?	0	
2	49	4.50		4.20	168.8		4.22	0	
3	0	4.50		4.20	168.8		4.22	0	
3.1	0	4.50		4.20	168.8	false	4.22	0	
3	1	4.50		4.20	168.8		4.22	0	
3.1	1	4.50		4.20	168.8	true	4.22	0	
3.1.1	1	4.50		4.20	168.8		4.22	1	
. . .	. . .	. . .	. . .	. . .	. . .		. . .	. . .	
3	39	4.50		4.20	168.8		4.22	1	
3.1	39	4.50		4.20	168.8	true	4.22	1	
3.1.1	39	4.50		4.20	168.8		4.22	2	
4	39	4.50		4.20	168.8		4.22	2	2 parti cipants…

שימו ♥: המעקב אחר הערכים השמורים באיבר של המערך זהה למעקב אחר הערכים השמורים במשתנה.

סוף פתרון בעיה 1

בפתרון הבעיה הכרנו שימוש במערך עבור שמירת נתונים שניתן להתייחס אליהם כאל אוסף סדור של איברים מאותו טיפוס. ניתן להתייחס אל איבר במערך כאל משתנה. כלומר, ניתן לשמור בו ערכים ולקרוא את הערכים השמורים בו. מיקומו הסידורי של איבר במערך מצוין **במציין** (index).

באלגוריתם לפתרון בעיה זו הכרנו דרך נוחה לסרוק מערך ולבצע עיבוד על כל איבריו באמצעות הוראה לביצוע-חוזר, שמספר הסיבובים בה ידוע מראש ושווה למספר האיברים במערך (במילים אחרות, לאורכו של המערך).

נסכם את המושגים הבסיסיים שהכרנו בפתרון בעיה 1, הנוגעים למערכים ולעבודה עמם בשפת Java:

מערך בשפת Java הוא עצם המוגדר בשפה.

משום שזהו עצם, אחרי ההצהרה על מערך צריך לבצע עבורו **הקצאת מקום בזיכרון**, באמצעות ההוראה **new**. בעקבות ההקצאה שם המערך מפנה אל שטח הזיכרון שהוקצה עבורו.

בשפת Java, מיד אחרי הקצאת שטח זיכרון עבור מערך של איברים מטיפוס פשוט, מתבצע גם **אתחול אוטומטי** של כל איבריו. איברי מערך מספריים (שלמים או ממשיים) יאותחלו ב-0, איברי מערך בוליאני יאותחלו ב-false, ואיברי מערך תווי יאותחלו בתו הריק.

ציון איברי מערך מתחיל תמיד ב-0. **פנייה לאיבר** הנמצא במיקום i מתחילת מערך בשם anArray נכתבת כך: anArray[i]. איבר זה הוא האיבר ה-i+1 ברשימת האיברים.

לכל עצם שהוא מערך יש **תכונה השומרת את אורכו** (length). ניתן להשתמש בה כדי לדעת את מספר האיברים במערך אך לא ניתן לשנותה. נוח להשתמש בתכונה זו כדי לקבוע את מספר הפעמים לביצוע בלולאות שסורקות את כל איברי המערך.

פנייה לתכונת האורך נעשית באמצעות סימון הנקודה, למשל כך: `anArray.length`.

בתוכנית שבפתרון בעיה 1 הצהרנו על המערך `scores` והקצינו לו מקום בזיכרון באופן הבא:

```
double[] scores = new double[NUM_OF_RUNNERS];
```

ננסח זאת באופן כללי:

הצהרה על מערך מטיפוס כלשהו נכתבת בשפת Java כך:

```
טיפוס[] שם המערך;
```

שם הטיפוס, לאחריו סוגריים מרובעים ואז שמו של המערך.

הקצאת שטח למערך מתבצעת באמצעות ההוראה **new**, אחריה מופיע טיפוס איברי המערך, ואחריו מופיע בסוגריים מרובעים מספר איברי המערך:

```
new טיפוס[מספר הערכים];
```

כזכור, ניתן לצרף את ההצהרה וההקצאה בהוראה אחת:

```
טיפוס[] שם המערך = new טיפוס[מספר הערכים];
```

וניתן גם לבצען בנפרד:

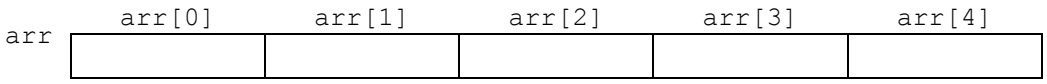
```
טיפוס[] שם המערך;
שם המערך = new טיפוס[מספר הערכים];
```

כדי להמחיש את המושגים החדשים נתבונן בדוגמה הבאה:

בשורה הבאה יש הצהרה על מערך מטיפוס שלם, והקצאת שטח המספיק ל-5 איברים.

```
int[] arr = new int[5];
```

אם כך המערך arr מורכב בעצם מחמישה תאים בזיכרון:



האיבר הראשון במערך arr הוא arr[0], האיבר השני הוא arr[1] וכן הלאה. האיבר האחרון הוא arr[4].

ניתן להתייחס לכל איבר במערך arr כאל משתנה מטיפוס שלם. למשל:

- השמה:

```
arr[4] = num;
```

- קלט:

```
arr[i] = in.nextInt();
```

- פלט:

```
System.out.println(arr[i]);
```

נראה כיצד ייראה המערך arr לאחר ביצוע ההוראות הבאות:

```
arr[0] = 9;
arr[1] = 7;
arr[2] = 2 * 4;
arr[3] = 4 + arr[1];
arr[4] = arr[2] + arr[3];
```

arr	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]
	9	7	8	11	19

שאלה 10.2

נניח כעת שבמרוץ הקרטינג משתתפים רק שלושה מתחרים, כלומר ערכו של הקבוע NUM_OF_RUNNERS הוא 3. הקלט לתוכנית (כלומר זמני ההקפה שלהם) הוא: 4.40 4.80 4.20. בנו טבלת מעקב לתוכנית BelowAverage ועקבו אחר מהלך ביצוע האלגוריתם לפי הנתונים החדשים. מהו הפלט המתקבל?

שאלה 10.3

הוסיפו הוראה או הוראות לתוכנית BelowAverage כך שהפלט יהיה רשימת זמני ההקפות הנמוכים מהממוצע.

שאלה 10.4

נתון קטע התוכנית הבא:

```
int a1,a2;
int[] arr = new int[4];
arr[0] = 2;
a1 = in.nextInt();
arr[2] = in.nextInt();
a2 = in.nextInt();
arr[3] = 2 * arr[2];
arr[1] = a2 + arr[2];
a1 = a[1] + a[2] + a1;
System.out.println(a1 + " " + a2);
for (i = 0; i < arr.length; i++)
      System.out.println(arr[i]);
```

בנו טבלת מעקב אחר מהלך ביצוע קטע התוכנית עבור הקלט: 3 2 1.
מה יהיה פלט קטע התוכנית עבור הקלט הנתון?

שאלה 10.5

נתונה הצהרת המערך הבאה:

```
int[] temp = new int[10];
```

א. כתבו לולאה להשמת הערך 0 בכל אחד מאיברי המערך.
ב. כתבו לולאה לקליטת עשרה נתוני קלט בעשרת איברי המערך.
ג. כתבו לולאה להכפלה ב-2 של ערכו של כל איבר במערך.
ד. כתבו לולאה להצגה של חצי מערכו של כל איבר במערך.

שאלה 10.6

פתחו אלגוריתם אשר הקלט שלו הוא רשימה של עשרה ציונים, והפלט שלו הוא רשימה הכוללת לכל ציון את מרחקו מהציון הממוצע. מרחקו של ציון מהציון הממוצע הוא |ציון ממוצע – ציון|, כלומר הערך המוחלט של ההפרש של הציון והציון הממוצע. ישמו את האלגוריתם בשפת Java.

שימו ♥: חשוב להבחין בין ערכו של **איבר** ובין ערכו של **המציין של איבר**, כפי שמראה הדוגמה הבאה.

נניח שנתון המערך `arr` הבא:

	arr[0]	arr[1]	arr[2]
arr	-1	2	1

ערכו של האיבר הראשון הוא 1-, ערכו של האיבר השני הוא 2 וערכו של האיבר השלישי הוא 1.
נתון משתנה `i`, מטיפוס שלם, ונתון קטע התוכנית הבא:

```
1. i = 0;
2. System.out.println(i + " " + arr[i]);
3. i = i + 1;
4. System.out.println(i + " " + arr[i]);
5. arr[i] = arr[i] + 1;
6. System.out.println(i + " " + arr[i]);
```

הנה טבלת המעקב אחר מהלך ביצוע ההוראות שבקטע התוכנית:

מספר שורה	המשפט הבא לביצוע	i	arr[0]	arr[1]	arr[2]	פלט
1	i = 0	0	-1	2	1	
2	System.out…(i + " " + arr[i])	0	-1	2	1	0 -1
3	i = i + 1	1	-1	2	1	
4	System.out…(i + " " + arr[i])	1	-1	2	1	1 2
5	arr[i] = arr[i] + 1	1	-1	3	1	
6	System.out…(i + " " + arr[i])	1	-1	3	1	1 3

בקטע התוכנית הזה תפקידו של המשתנה `i` הוא להיות מציין של איברי המערך `arr`, כלומר משתנה שבאמצעותו פונים אל איברי המערך. כפי שניתן לראות מטבלת המעקב ערכו של `i` אינו שווה בהכרח לערכו של האיבר שנמצא במקום ה-`i`, כלומר ל-`arr[i]`.

בפרט, שינוי בערכו של האיבר שנמצא במקום ה-`i` אינו משפיע על ערכו של `i`, כפי שמדגימות הוראת ההשמה שבשורה 5 והוראת הפלט שבשורה 6.

שאלה 10.7

כתבו קטע תוכנית המצהיר על מערך מטיפוס שלם בגודל 10, ומציב בכל תא ערך מספרי ששווה לריבוע מקומו הסידורי. למשל, בתא 0 יהיה הערך 0 ובתא 5 יהיה הערך 25.

שאלה 10.8

במערך t שלהלן שמורים ערכים שלמים:

```
int[] t = new int[20]
```

א. מה מטרת משפט ה-for הבא:

```
for (int i = 0; i < t.length; i++)
      if (t[i] > i)
            System.out.println(i);
```

ב. כתבו משפט for אשר יציג כפלט את מיקומם במערך (כלומר, את מציינים) של כל האיברים במערך שערכם כפול ממיקומם הסידורי במערך.

שאלה 10.9

בתוכנית הבאה נשתמש במערך של תווים:

```
/*
קלט: עשר אותיות לועזיות
_____________________________________:פלט
*/
import java.util.Scanner;
public class Letters
{
    public static void main (String[] args)
    {
        char[] letters = new char[10];
        for (int i = 0; i < letters.length; i++)
        {
            System.out.print("Enter a character: ");
            letters[i] = in.nextLine().charAt(0);
        } // for
        for (int i = 0; i < letters.length; i++)
            if (letters[i] == letters[letters.length -1] )
                  System.out.println(i);
    } // main
} // Letters
```

א. מהו פלט התוכנית עבור הקלט: ABBASABABA?

ב. מהי מטרת התוכנית? מלאו את תיאור הפלט בהערה שבראש התוכנית.

ג. האם נחוץ שימוש במערך לצורך השגת המטרה שתיארתם בסעיף ב?

בדוגמאות שראינו עד כה ההצהרה על מערך לוותה בהקצאה מיידית של מקום עבורו, בעזרת הפעולה **new**. עם זאת, הזכרנו שהקצאת המקום בזיכרון יכולה להתבצע גם מאוחר יותר. לעתים, אכן נרצה לדחות את ההקצאה, למשל, כאשר לא ידוע מראש גודל המערך והוא תלוי בקלט לתוכנית, כפי שמדגימה הבעיה הבאה.

בעיה 2

מטרת הבעיה ופתרונה: הצגת מערך שאורכו אינו ידוע מראש.

לכיתה המדעית יתקבלו רק התלמידים אשר ציוניהם במבחן הקבלה גבוה מהציון הממוצע של הניגשים למבחן.
פתחו אלגוריתם אשר קולט את מספר הניגשים למבחן, ואחר כך קולט את רשימת הציונים של הנבחנים. פלט האלגוריתם יהיה מספר התלמידים המתקבלים לכיתה. ישמו את האלגוריתם בשפת Java.
ניתן להניח כי לפחות תלמיד אחד ניגש למבחן.

פירוק הבעיה לתת-משימות

התהליך המבוקש כמעט זהה לזה שבפתרון בעיה 1: עלינו לקלוט נתונים ולשמור אותם, לחשב את הממוצע, ולמנות את מספר הנתונים במערך ה**גדולים** מהממוצע. (שלא כמו בבעיה 1, שם היה עלינו למנות את מספר הנתונים ה**קטנים** מהממוצע).

אבל ההבדל העיקרי בין בעיה זו לבעיה 1 הוא שכעת לא ידוע מראש מספר הנתונים, ולכן צריך קודם כול לקרוא ערך זה מהקלט.

בהתאם לכך נקבל את הפירוק הבא לתת-משימות:

1. קליטת מספר הניגשים למבחן
2. קליטת הציונים, שמירתם וצבירתם
3. חלוקת הסכום המצטבר במספר הנבחנים
4. השוואת כל ציון לממוצע, ומנייתו אם הוא גדול מהממוצע

בחירת משתנים

נשתמש במשתנים הבאים:

numOfStudents – מטיפוס שלם, לשמירת מספר התלמידים הנבחנים
grades – מערך באורך numOfStudents מטיפוס ממשי, לשמירת הציונים
sumOfGrades – מטיפוס ממשי, ישמור את סכום הציונים
averageGrade – ממוצע הציונים, מטיפוס ממשי
aboveAverageGrade – למניית מספר הציונים מעל לממוצע, מטיפוס שלם

האלגוריתם

האלגוריתם כמעט זהה לאלגוריתם שהוצג בפתרון בעיה 1:

שאלה 10.10

כתבו את האלגוריתם לפתרון בעיה 2. היעזרו באלגוריתם שניתן לפתרון בעיה 1, ושנו אותו כך שיכלול הוראת קלט של מספר התלמידים הנבחנים, יתייחס למשתנים שנבחרו וייתן את הפלט המבוקש.

יישום האלגוריתם

הנה קטע התוכנית המטפל בהצהרה על המערך, בקליטת מספר התלמידים, ובהקצאת מקום בזיכרון עבור המערך:

```
int numOfstudents;
int[] grades;
System.out.print("Enter number of students: ");
numOfstudents = in.nextInt();
grades = new int[numOfstudents];
```

מיד אחרי שמוקצה למערך מקום מתאים בזיכרון, התכונה `length` של המערך שומרת את אורכו.

שאלה 10.11

השלימו את התוכנית לפתרון הבעיה.

סוף פתרון בעיה 2

שאלה 10.12

כתבו קטע תוכנית המקבלת כקלט מספר שלם חיובי וזוגי, ומקצה מערך מטיפוס שלם בגודל הערך שנקלט. לאחר מכן התוכנית תשים באיברי המערך הנמצאים במחציתו הראשונה את המספר 0, ובאיברי המערך הנמצאים במחציתו השנייה את המספר 1.

בדוגמאות שראינו עד כה איברי מערך נסרקו באופן רציף, זה אחר זה. לעתים, נרצה לסרוק איברי מערך באופן לא רציף. למשל, נניח ש-`arr` הוא מערך של עשרה איברים ויש להציג כפלט את ערכיהם של כל האיברים שבמקומות הזוגיים במערך. גם עבור סריקה כזאת נוכל להשתמש בהוראה לביצוע-חוזר באורך ידוע מראש, אך נקדם את משתנה הבקרה של הלולאה בדילוגים של 2:

```
for (int i = 0; i < arr.length; i = i + 2)
      System.out.println(arr[i]);
```

שאלה 10.13

נתון המערך `arr` המכיל מאה איברים מטיפוס שלם.

א. תארו את מטרת הלולאה הבאה:

```
for (int i = 0; i < arr.length; i++)
      if (arr[i] % 5 == 0)
            System.out.println(arr[i]);
```

ב. תארו את מטרת הלולאה הבאה:

```
for (int i = 0; i < arr.length; i++)
      if (i % 5 == 0)
            System.out.println(arr[i]);
```

ג. כתבו לולאה יעילה יותר (שמספר הסיבובים בה קטן יותר) להשגת המטרה של סעיף ב.

בבואנו לכתוב הוראה לביצוע-חוזר המבצעת סריקה ועיבוד של איברי מערך, נתאים את סוג ההוראה להגדרת הסריקה שצריכה להתבצע.

אם ידועים מראש הן המציין שממנו צריכה הסריקה להתחיל והן המציין שהיא אמורה להסתיים בו, והסריקה עצמה היא בדילוגים ידועים, נשתמש בהוראה לביצוע-חוזר שמספר הסיבובים בה ידוע מראש (המיושמת בלולאת **for**).

אם סיום הסריקה תלוי בקיום תנאי, נשתמש בביצוע-חוזר-בתנאי (המיושמת בלולאת while).

שאלה 10.14

במערך s יש ערכים מטיפוס תו.

א. תארו את מטרת קטע התוכנית הבא:

```
int i = 0;
int len = s.length;
while (s[i] < s[len - 1])
    i = i + 1;
System.out.println(i);
```

ב. תארו את מטרת קטע התוכנית הבא:

```
int c = 0;
int len = s.length;
for (int i = 0; i < len; i++)
    if ( (i % 10 == 0) && (s[i] < s[len - 1]) )
        c = c + 1;
System.out.println(c);
```

ג. לולאת ה-**for** שבקטע התוכנית בסעיף ב מתבצעת מספר פעמים השווה לאורך המערך s. כתבו לולאה שמבצעת אותה המשימה אך מספר הסיבובים בה יהיה הרבה יותר קטן.

מעיה 3

מטרת הבעיה ופתרונה: שימוש במערך כדי לתאר מבנה כגון לוח משחק, ושימוש בביטויים חשבוניים כמצייני מיקום .

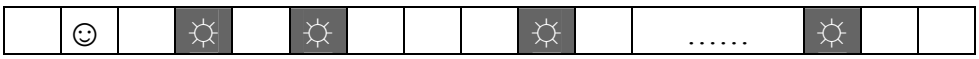
נתאר משחק על לוח ובו שורה של שלושים משבצות ובחלקן מצויים מוקשים. המשבצות ממוספרות מ-1 עד 30.

שחקן מתקדם לאורך הלוח באמצעות הטלת קובייה שעל פאותיה הספרות 1 עד 6. אם המספר המתקבל מהטלת הקובייה מוביל למשבצת שיש בה מוקש, השחקן נשאר במקומו. אם המספר המתקבל מהטלת הקובייה מוביל למשבצת פנויה השחקן יכול להתקדם.

פתחו אלגוריתם אשר הקלט שלו הוא תיאור הלוח המתקבל באמצעות שלושים תווים שערכיהם 'T' (עבור משבצת פנויה) או 'F' (עבור מוקש), ואחריו מספר המציין את מספר המשבצת שהשחקן מוצב עליה. פלט האלגוריתם הוא הטלות הקובייה אשר עבורן יוכל השחקן להתקדם. הניחו שהשחקן מוצב על משבצת שמספרה בין 1 ל-24 ואין בה מוקש. ישמו את האלגוריתם בשפת התכנות Java.

ניתוח הבעיה בעזרת דוגמאות

נתבונן בלוח הבא, כאשר הסימון ☺ מציין את מיקום השחקן והסימון ☼ מציין מוקש:



בלוח זה, השחקן יכול לקדם את הכלי עבור כל אחת מן ההטלות: 1, 3, 5 או 6, אך לא יוכל לקדם אותו עבור ההטלות 2 או 4.

פירוק הבעיה לתת-משימות

הנה חלוקה ראשונית לתת-משימות עבור פתרון הבעיה:

1. קליטה ושמירה של מצב הלוח ושל מקום השחקן על הלוח
2. חישוב הטלות הקובייה המותרות להתקדמות והצגתן כפלט

בחירת משתנים

את לוח המשחק נתאר באמצעות מערך של ערכים בוליאניים בגודל 30, שמצייני איבריו נעים בין 0 ל-29. כל איבר במערך ישמור את מצב המשבצת המתאימה בלוח. הערך true יתאר משבצת פנויה, ואילו הערך false יתאר מוקש. בנוסף נזדקק למשתנה שיציין את מיקום השחקן על הלוח.

אם כך אלה יהיו המשתנים שנשתמש בהם:

board – מערך של 30 איברים מטיפוס בוליאני
pawn – מטיפוס שלם, לשמירת מיקום השחקן על הלוח

למשל עבור הדוגמה שלעיל המערך board ייראה כך:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	…	[27]	[28]	[29]
true	true	true	false	true	false	true	true	true	false	true	…	false	true	true

והמשתנה pawn יכיל את הערך 1 (כיוון שהשחקן נמצא על המשבצת השנייה).

ההתייחסות לאיברי המערך היא בעזרת מציין, לדוגמה הערך בתא board[pawn] מבטא את מצב המשבצת שכלי השחקן מוצב עליה.

שימו ♥: אמנם לוח המשחק ממוספר מ-1 עד 30, אבל המערך המתאר אותו ממוספר מ-0 עד 29. לכן ערכי המשתנה pawn יהיו בתחום 0 עד 29.

האלגוריתם

❓ כיצד נשתמש במערך board כדי לחשב את ההטלות שעבורן יוכל השחקן להתקדם?

כדי לחשב את ההטלות שעבורן יוכל השחקן להתקדם יש לבדוק את מצבן של שש המשבצות העוקבות למשבצת שהשחקן מוצב עליה. כיוון שמשבצת זו מתוארת באמצעות האיבר board[pawn] במערך, שש המשבצות העוקבות מיוצגות על ידי ששת האיברים הבאים לפי הסדר:
board[pawn+1], board[pawn+2], ..., board[pawn+6].

עבור כל איבר מששת האיברים האלה יש לבדוק אם הוא מייצג משבצת פנויה (כלומר אם ערכו הוא true). אם כן, יש להציג כפלט את הטלת הקובייה אשר מביאה אליו את השחקן.

למשל, אם ערכו של board[pawn+3] מבטא משבצת פנויה, יוצג 3 כפלט, כיוון שהטלת 3 בקובייה מביאה את השחקן למשבצת פנויה.

1. עבור כל i שמן מ-0 עד גודל המערך פחות 1 בצע:
   1.1. קלוט את מצב המשבצת ה-i שלו הלוח
2. קלוט את מיקום השחקן
3. עבור כל i שמן מ-1 עד 6 בצע:
   3.1. אם המשבצת במרחק i ממיקום השחקן היא ריקה
      3.1.1. הדפס את ערכו של i

התוכנית המלאה

```
/*
קלט: תיאור לוח משחק בן 30 משבצות
ומיקום משבצת שעליה מוצב שחקן (בין 1 ל-24)
פלט: הטלות קוביה המביאות את השחקן למשבצת פנויה */
import java.util.Scanner;
public class MineBoardGame
{
    public static void main (String[] args)
    {
        Scanner in = new Scanner(System.in);
        // הגדרת קבוע
        final int BOARD_SIZE = 30;
        // הגדרת משתנים
        boolean[] board = new boolean[BOARD_SIZE];
        char cellStatus;
        int pawn;
        // קלט הלוח
        System.out.println("Enter the board details: Type F for a " +
                           "cell with a mine, and T for a free cell: ");
        for (int i = 0; i < board.length; i++)
        {
            System.out.print("Enter status of cell number " + (i+1));
            cellStatus = in.nextLine().charAt(0);
            board[i] = (cellStatus == 'T');
        }
        // קלט מיקום השחקן והתאמתו לאופן שמירת הלוח
        System.out.print("Enter the pawn position: ");
        pawn = in.nextInt();
        pawn = pawn - 1;
        // פלט
        System.out.println("Throws that lead to empty positions: ");
        for (int i = 1; i <= 6; i++)
            if (board[pawn + i])
                System.out.println(i);
    }// main
} //class MineBoardGame
```

שימו ♥: בלולאה הראשונה השתמשנו במציין `i` כדי לעדכן את מצב המשבצת התורנית בלוח. כדי להודיע למשתמש את מספר המשבצת שאת מצבה עליו להקליד השתמשנו בביטוי `i + 1`. הביטוי מתאים את מציניי המערך (הנעים בין 0 ל-29) לאופן הספירה של הלוח (מ-1 עד 30). וכך בסיבוב הראשון של הלולאה, כאשר ערכו של `i` הוא 0, מוצגת ההודעה:

```
Enter status of cell number 1:
```

בלולאה השנייה השתמשנו בביטוי `pawn + i` כמציין. הביטוי `pawn + i` מבטא מיקום במערך שמרחקו מהמיקום `pawn` הוא `i`.

? לאחר קליטת מיקום השחקן, מדוע מושם במשתנה `pawn` הערך הנקלט פחות 1?

המשתמש בתוכנית יקליד את מיקום השחקן על הלוח כמספר שלם בין 1 ל-30. לעומת זאת, ערכי המשתנה `pawn` צריכים להתאים לאופן מספור המערך מ-0 עד 29.

סוף פתרון בעיה 3

שאלה 10.15

בנו טבלת מעקב אחר מהלך ביצוע התוכנית `MineBoardGame` לפתרון בעיה 3 עבור הקלט:

T T F F T F T F T . . . T 2

כלומר יש מוקש במשבצת השלישית והרביעית, ומהמשבצת השישית ואילך יש מוקש בכל משבצת זוגית. השחקן נמצא במשבצת השנייה.

מהו הפלט המתקבל?

שאלה 10.16

הרחיבו את האלגוריתם לפתרון בעיה 3 כך שיציג כפלט לא רק את ההטלות שמאפשרות לשחקן להתקדם, אלא גם את מספרי המשבצות שהטלות אלו מובילות אליהן. למשל עבור הקלט שבשאלה הקודמת יכלול הפלט החדש גם את הרשימה: 7 5.

שאלה 10.17

נניח שכתבנו את משפטי ההצהרה הבאים:

```
int[] arr = new int[100];
int position;
int counter = 0;
```

באיברי המערך `arr` הושמו ערכים ובמשתנה `position` נקלט ערך חיובי שלם בתחום 1 עד 79.

א. תארו את מטרת קטע התוכנית הבא:

```
if (arr[position] == arr[position - 1])
    System.out.println("previous one is equal");
if (arr[position] == arr[position + 1])
    System.out.println("next one is equal");
```

ב. תארו את מטרת קטע התוכנית הבא:

```
for (int i = 0; i < 10; i++)
    if (arr[position] == arr[position + i])
        counter = counter + 1;
System.out.println(counter);
```

ג. כתבו לולאה שמטרתה להציג את מציינים של איברי המערך, מבין 20 האיברים העוקבים ל-`arr[position]`, אשר ערכם גדול מערכו של `arr[position]`.

ד. כתבו לולאה שמטרתה להציג את מציינים של איברי המערך, מבין 20 האיברים העוקבים ל-`arr[position]`, אשר ערכם גדול **בדיוק ב-1** מערכו של `arr[position]`.

שאלה 10.18

פתחו וישמו אלגוריתם אשר הקלט שלו הוא רצף של 20 תווים, והפלט שלו הוא מספר הזוגות של תווים עוקבים ששווים לזוג התווים האחרון.

למשל, עבור הקלט abcabacadcababddaaab הפלט הוא 4 כיוון שהזוג האחרון הוא ab, ויש עוד ארבעה זוגות השווים ל-ab.

שימו ♥: עבור קלט של 20 תווים זהים הפלט צריך להיות 18 (כיוון שכל הזוגות, מהראשון ועד הזוג לפני האחרון, שווים לזוג האחרון).

שאלה 10.19

במשחק נתון לוח הכולל שורה של 25 משבצות אשר בכל אחת מהן רשום מספר שלם בין 0 ל-5. על הלוח מוצב שחקן בין המשבצת העשירית והחמש-עשרה.

השחקן מטיל קובייה כדי לנסות ולהתקדם על הלוח. אם המספר המתקבל בהטלת הקובייה מוביל למשבצת אשר המספר הרשום בה קטן או שווה למספר המתקבל בקובייה, השחקן מקדם את הכלי למשבצת זו. אחרת השחקן נסוג אחורה מספר משבצות השווה למספר המתקבל בקובייה.
פתחו אלגוריתם אשר הקלט שלו הוא תיאור הלוח (25 מספרים שערכיהם נעים בין 0 ל-5), ואחריו מיקום השחקן על הלוח וערכה של הטלת הקובייה. הפלט שלו הוא מיקומו החדש של השחקן על הלוח. ישמו את האלגוריתם בשפת Java.

שימו ♥: יש להתאים בין מיקום השחקן כפי שנקלוט (מספר בין 1 ל-25) לבין מציין המערך (מספר בין 0 ל-24).

שאלה 10.20

מה יהיו ערכי איברי המערך בסיום קטע התוכנית הבא?

```
int[] arr = new int[10];
for (int i = 0; i < arr.length / 2; i++)
    arr[i * 2] = i;
```

שאלה 10.21

המערך `arr` הוא מערך באורך 10 המכיל מספרים שלמים. תארו מה יהיו ערכי איברי המערך בסיום קטע התוכנית הבא.

```
for (int i = 0; i < arr.length; i++)
    arr[i] = arr[arr.length - i];
```

שאלה 10.22

א. המערך `arr` הוא מערך באורך 10 המכיל מספרים שלמים. תארו מה יהיו ערכי איברי המערך בסיום קטע התוכנית הבא.

```
for (int i = 0; i < arr.length - 1; i++)
    arr[i + 1] = arr[i];
```

ב. תקנו את קטע התוכנית וכתבו אותו מחדש כך שיבצע הזזה של ערכי התאים במערך בצורה מעגלית, כלומר, התא השני יכיל את ערכו המקורי של התא הראשון, התא השלישי יכיל את ערכו המקורי של התא השני וכן הלאה, והתא הראשון – יכיל את ערכו המקורי של התא האחרון.

שאלה 10.23

א. חברי קבוצת "סיורי הגליל" הולכים תמיד בטור, זה אחר זה. לאחר עשרים דקות הליכה, הראשון בטור נעצר וממתין עד שהטור כולו עובר אותו, ואז מצטרף לטור שוב כאחרון. פתחו אלגוריתם המקבל כקלט את מספר חברי הקבוצה, ולאחר מכן את רשימת השמות של חברי הקבוצה, לפי הסדר שבו הם מתחילים את הטיול. פלט האלגוריתם יהיה סדר חברי הקבוצה לאחר שעה ורבע של הליכה. ישמו את האלגוריתם בשפת התכנות Java.

ב. חברי קבוצת "סיורי הגליל למתחילים" הולכים תמיד בטור; המדריך צועד בראש, ובסוף נמצאים החובש, המלווה והמאסף. בקדמת הטור צועדות הבנות ואחריהן צועדים הבנים. רק הבנים מחליפים את מקומותיהם כל עשרים דקות כפי שתואר עבור קבוצת "סיורי הגליל". פתחו אלגוריתם המקבל כקלט את מספר הבנים ואת מספר הבנות בקבוצה, ואת הרשימה של שמות חברי הקבוצה לפי הסדר שבו הם מתחילים את הטיול. פלט האלגוריתם יהיה סדר חברי הקבוצה לאחר שעה ורבע של הליכה. ישמו את האלגוריתם בשפת התכנות Java.

שאלה 10.24

פתחו אלגוריתם המקבל כקלט סדרת מספרים בסדר עולה. האלגוריתם מסדר ושומר את הסדרה הנתונה בסדר יורד. ישמו את האלגוריתם בשפת התכנות Java.

10.2 חריגה מגבולות מערך

? בהגדרת בעיה 3 נכללה הנחה כי השחקן מוצב על משבצת שמספרה בין 1 ל-24. מדוע הנחה זו חשובה?

נניח כי השחקן עומד על משבצת שערכה גבוה מהמספר 24, למשל 25. הטלת הקובייה נותנת מספר כלשהו בין 1 ל-6. אם הטלת הקובייה תיתן את המספר 6, השחקן ינסה לנוע 6 צעדים קדימה החל מהמשבצת ה-25. כך יגיע השחקן למשבצת מספר 31. משבצת זו אינה קיימת כמובן ואינה חוקית בתנאי המשחק!

ניסיון להתייחס במהלך הרצת תוכנית בשפת Java לתא שאיננו קיים במערך (למשל `board[30]`) יגרום להפסקת פעולתה של התוכנית ולהצגת הודעת שגיאה על חריגה מגבולות המערך. חריגה מגבולות המערך היא שגיאת ריצה, המתגלה בזמן ריצת התוכנית ולא בזמן ההידור.

שאלה 10.25

נוסיף לפתרון בעיה 3 משתנה בשם `limit` מטיפוס שלם, אשר ישמש כערך סופי למשתנה הבקרה `i` של הלולאה השנייה (לולאת הצגת הפלט).
נשתמש ב-`limit` כדי להרחיב את משפטי התוכנית `MineBoardGame`, כך שהתוכנית תאפשר לקלוט ערך בין 1 ל-29 למיקום השחקן (במקום 24 כמו בבעיה המקורית). יחד עם זאת לא תהיה חריגה מגבולות המערך בלולאה השנייה.
לפני הלולאה השנייה נוסיף הוראה לביצוע-בתנאי:

```
if (_________________)
    _________________
else
    _________________
```

וכותרת הלולאה השנייה תהיה:

```
for (int i = 1; i <= limit; i++)
```

השלימו את ההוראה לביצוע-בתנאי, כך שהתוכנית תבצע את הדרוש.

שאלה 10.26

נתון המערך `arr` שהצהרתו היא:

```
int[] arr = new int[8];
```

וערכי איבריו הם:

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	arr[7]
10	20	40	30	4	5	7	6

א. מה יהיה פלט הלולאה הבאה?

```
for (int i = 0; i < arr.length - 1; i++)
    if ((i == arr[i]) || (i == arr[i + 1]))
        System.out.println(i);
```

ב. מה יהיו ערכי המערך אחרי ביצוע הלולאה הבאה?

```
for (int i = 0; i < arr.length; i++)
{
    if (arr[i] == (10 * i))
        arr[i] = arr[i] + 1;
    else
        arr[i] = arr[i] - 1;
}
```

ג. עבור אילו מארבעת הלולאות הבאות תהיה חריגה מגבולות המערך? תקנו לפי הצורך.

1.
```
for (int i = 0; i < arr.length; i++)
    arr[i] = arr[i + 1];
```

2.
```
for (int i = 0; i < arr.length - 1; i++)
        arr[i] = arr[i - 1];
```

3.
```
for (int i = 0; i < arr.length - 1; i++)
    arr[i] = arr[i] * 2;
```

4.
```
for (int i = 0; i < arr.length - 1; i++)
    arr[i] = arr[i * 2];
```

שאלה 10.27

השאלה מתייחסת למערך `arr` בן 8 האיברים שהוגדר בשאלה הקודמת.

כתבו לולאה אשר תשווה את ערכיהם של כל זוג איברים שכנים במערך (`arr[0]` ל-`arr[1]`, `arr[1]` ל-`arr[2]`, ... , `arr[6]` ל-`arr[7]`), ותחשב את מספר זוגות האיברים השכנים שערכיהם שווים זה לזה.

שימו ♥: לא לחרוג מגבולות המערך.

שאלה 10.28

על תמר הוטל לכתוב תוכנית המגדירה מערך בן 50 תאים, ומציבה בהם את הערכים:

0.5 1 1.5 2 2.5 3 3.5...

תמר כתבה:

```
double[] arr = new double[50];
for (int i = 0; i < arr.length * 2; i++)
    arr[i] = i / 2;
```

האם קטע התוכנית נכון? אם איננו נכון – תקנו את השגיאות כך שתושג המטרה המבוקשת.

שאלה 10.29 (מבגרות 2002)

לפניכם קטע תוכנית:

```
for (int i = 0; i < n - 2; i++)
    if (a[i] + 2 == a[i + 2])
        System.out.println(i + " " + a[i]);
```

`a` הוא המערך הבא בגודל 10:

a:	3	18	5	20	2	4	5	6	1	9

א. עקבו בעזרת טבלת מעקב אחר קטע התוכנית עבור n=10 ועבור המערך `a` הנתון, ורשמו מה יהיה הפלט.

ב. הסבירו מדוע קבע כותב קטע התוכנית שהתנאי להמשך ביצוע הלולאה יהיה i < n-2 ולא i<n.

10.3 קשרים בין מערכים

בדוגמאות שראינו עד כה השתמשנו במערך יחיד. לעתים פתרון בעיה מצריך יותר ממערך אחד. בסעיף זה נראה פתרונות אלגוריתמיים המשתמשים ביותר ממערך אחד. בחלקו הראשון של הסעיף נציג עיבוד של כמה מערכים במקביל בקצב התקדמות זהה, ובחלקו השני נעסוק גם בעיבוד של כמה מערכים שאינו נעשה באותו קצב התקדמות.

עיבוד מערכים במקביל ובקצב התקדמות זהה

השאלה הבאה עוסקת בניתוח קטע תוכנית המעבד במקביל שני מערכים, ופתרון השאלה שאחריה מצריך שימוש בשלושה מערכים.

שאלה 10.30

m1 ו-m2 הם שני מערכים שהוצהרו באופן הבא:

```
final int SIZE = 10;
int m1[] = new int[SIZE];
int m2[] = new int[SIZE];
```

נתונים שני קטעי התוכניות הבאים אשר אמורים להציג כפלט אם שני המערכים מכילים בדיוק את אותם איברים באותו סדר.

I

```
boolean compare = false;
for (int i = 0; i < SIZE; i++)
    if (m1[i] == m2[i])
        compare = true;
if (compare)
    System.out.println("The
                arrays are equal");
else
    System.out.println("The arrays
                  are not equal");
```

II

```
boolean compare = false;
for (int i = 0; i < SIZE; i++)
    compare = (m1[i] == m2[i]);

if (compare)
     System.out.println("The
                arrays are equal");
else
    System.out.println("The
           arrays are not equal");
```

שני קטעי התוכניות שגויים. ענו על הסעיפים הבאים ביחס **לכל אחד** מהקטעים:

א. הביאו דוגמה לערכים של איברים בשני המערכים שקטע התוכנית משיג את מטרתו עבורם.

ב. הביאו דוגמה לערכים של איברים בשני המערכים שקטע התוכנית אינו משיג את מטרתו עבורם, והסבירו מדוע זה קורה.

ג. תקנו את קטע התוכנית כך שישיג את מטרתו.

שאלה 10.31

כיתה י'5 ניגשה למבחן. לאחר המבחן הוחלט כי הכיתה כולה תיגש למבחן נוסף, והציון הגבוה מבין שני ציוני המבחנים לכל תלמיד יהיה הציון הקובע.

פתחו אלגוריתם הקולט את מספר התלמידים בכיתה, ולאחר מכן קולט את ציוני המבחן הראשון למערך בשם grade1 ואת ציוני המבחן השני למערך בשם grade2. האלגוריתם ימלא מערך בשם finalGrade כך שיכיל את הציון הקובע לכל אחד מתלמידי הכיתה, ויציג כפלט עבור כל אחד מהתלמידים את ציון המבחן הראשון, את ציון המבחן השני ואת הציון הקובע בליווי הודעות מתאימות. ישמו את האלגוריתם בשפת Java.

עיבוד מערכים במקביל בקצב התקדמות לא זהה

תרגיל 4

מטרת הבעיה ופתרונה: הצגת עיבוד של כמה מערכים במקביל בקצב התקדמות לא זהה.

פתחו אלגוריתם המקבל כקלט מספר חיובי שלם num, ולאחר מכן קורא מהקלט num מספרים שלמים לתוך מערך numbers. האלגוריתם מעתיק מהמערך numbers את המספרים החיוביים למערך positive ואת המספרים השליליים למערך negative, ושומר על סדר המספרים. ישמו את האלגוריתם בשפת Java.

ניתוח הבעיה בעזרת דוגמאות

נניח כי הקלט הוא 2- 4 3 1- 1 5.

המערך numbers ישמור בתוכו 5 מספרים שלמים. בעקבות ההעתקה הדרושה יכיל המערך positive שלושה איברים: 1, 3 ו-4. המערך negative יכיל שני איברים: 1- ו-2-.

אנו רואים כי לא רק שלכל מערך מועתק מספר שונה של איברים, אלא שקצב ההתקדמות על המערכים הללו שונה: ייתכן כי נסרוק כמה איברים ברצף במערך numbers, נשים אותם באחד המערכים, אך במקביל לא נתקדם כלל בהשמה במערך האחר. למשל בדוגמת הקלט שלעיל, בזמן שבמערך numbers נסרק האיבר השלישי והרביעי בזה אחר זה, אין כלל התקדמות במילוי המערך negative.

פירוק הבעיה לתת-משימות

1. קליטת מספר הערכים לעיבוד
2. קליטת רשימת הערכים ושמירתה במערך numbers
3. ביצוע במקביל של סריקת המערך numbers ומילוי המערכים positive ו-negative

בחירת משתנים

כדי לסרוק את המערכים בקצב התקדמות לא זהה, עלינו לשמור שלושה מציינים שונים, אחד עבור כל מערך: המציין של המערך numbers יסרוק את איברי המערך בזה אחר זה. המציין של המערך positive והמציין של המערך negative יצביעו בכל רגע על המקום הפנוי הבא במערך המתאים להם, כדי שהעתקת האיבר הבא תתבצע למקום הנכון.

המציין של המערך numbers יתקדם ב-1 בכל פעם שנסיים עיבוד של איבר תורן במערך ונתקדם לעבר האיבר הבא.

כל אחד מהמציינים האחרים, של המערך positive ושל המערך negative, יתקדם ב-1 רק אחרי ביצוע העתקה למערך שאליו הוא מתייחס.

נשתמש במשתנים הבאים:

size – מספר הערכים לעיבוד
numbers – מערך מטיפוס שלם, בגודל size
positive – מערך מטיפוס שלם, בגודל size
negative – מערך מטיפוס שלם, בגודל size

indexNum – מטיפוס שלם, יציין את מיקום האיבר התורן במערך numbers
indexPos – מטיפוס שלם, יציין את המקום הפנוי הבא במערך positive
indexNeg – מטיפוס שלם, יציין את המקום הפנוי הבא במערך negative

האלגוריתם

1. *קלוט ערך חיובי שלם ב-*size
2. *עבור כל* i *שלם מ-0 עד* size-*1 בצע:*
   2.1. *קלוט למקום ב-*numbers[i]
3. *אתחל את* indexPos *ב-0*
4. *אתחל את* indexNeg *ב-0*
5. *עבור כל* indexNum *שלם מ-0 עד* size-*1 בצע:*
   5.1. *אם* numbers[indexNum] > *0*
      5.1.1. *השתק את תכולת* numbers[indexNum] *ל-*positive[indexPos]
      5.1.2. *הגדל את תכולת* indexPos *ב-1*
   5.2. *אחרת*
      5.2.1. *השתק את תכולת* numbers[indexNum] *ל-*negative[indexNeg]
      5.2.2. *הגדל את תכולת* indexNeg *ב-1*

שאלה 10.32

ישמו את האלגוריתם לפתרון בעיה 4 בשפת Java.

סוף פתרון בעיה 4

שאלה 10.33

פתחו אלגוריתם אשר מקבל כקלט מספר חיובי שלם, ולאחר מכן קולט רשימת מספרים שאורכה כערך הנתון הראשון. האלגוריתם ישמור את המספרים הנקלטים בשני מערכים שונים, מערך אחד עבור מספרים זוגיים, ומערך אחר עבור מספרים אי-זוגיים. האלגוריתם יציג כפלט את כל הערכים שברשימה: ראשית את כל המספרים הזוגיים, לפי סדר קליטתם ואחר-כך את כל המספרים האי-זוגיים לפי סדר קליטתם. ישמו את האלגוריתם בשפת Java.

שאלה 10.34

נניח ש-arr1 ו-arr2 הם שני מערכים מטיפוס שלם ואורכם אינו בהכרח שווה. כתבו קטע תוכנית אשר מעתיק למערך שלישי arr3, רק את האיברים אשר נמצאים גם ב-arr1 וגם ב-arr2.
למשל, אם ב-arr1 וב-arr2 נמצאים הערכים הבאים:

arr1	36	8	9	73	11	3	4
arr2	4	77	8	15	12		

אז ב-arr3 יהיו הערכים הבאים:

arr3	4	8	0	0	0	0	0

הדרכה: יש להשתמש בקינון של לולאות.

10.4 מערך מונים

בפרק 7 למדנו על משתנה מסוג מונה. למדנו כי מונה הוא משתנה אשר תפקידו למנות מספר של אירועים שמתרחשים (כמו למשל הופעות של נתונים או מספר חישובים שביצענו). כיוון שהשימוש במונה הוא לצורך ספירה, מונה הוא משתנה מטיפוס שלם. לעתים יש למנות במקביל אירועים שונים . לשם כך ניתן להשתמש במערך של מונים, כפי שמדגימה הבעיה הבאה. העבודה עם מערך מונים היא תבנית שימושית לפתרון בעיות אלגוריתמיות.

בעיה 5

מטרת הבעיה ופתרונה: הצגת מערך מונים ושימוש בו

בתוכנית "כוכב נולד" יש N מועמדים ולכל מועמד מספר סידורי בין 1 ל-N. הצופים בבית נתבקשו לבחור את אחד המועמדים ולהצביע עבורו. בכל תוכנית המועמד שמקבל את מספר הקולות הקטן ביותר עוזב את התוכנית. פתחו אלגוריתם אשר מקבל כקלט את מספר המועמדים (מספר חיובי שלם), ואחר כך קולט רשימה של הצבעות של הצופים, המסתיימת במספר 1-. האלגוריתם מודיע מי המועמד אשר קיבל את מספר הקולות הקטן ביותר. ישמו את האלגוריתם בשפת התכנות Java.

פירוק הבעיה לתת-משימות

1. קליטת מספר מועמדים
2. קליטת הצבעות למועמדים תוך מניה של ההצבעות עבור כל מועמד
3. מציאת המונה בעל הערך המינימלי והצגה של המועמד שאליו הוא מתאים

בחירת משתנים

כדי לבצע את תת-משימה 2 עלינו לתפעל במקביל כמה מונים, מונה לכל מועמד. כלומר מספר המונים שווה למספר המועמדים. כמובן שאין אנו יכולים להצהיר על מונה נפרד לכל מועמד: מספר המועמדים אינו ידוע מראש, ואפילו אם היה ידוע, ייתכן כי הוא גדול למדי. בכל מקרה, הצהרה על מונה נפרד לכל מועמד הופכת את התוכנית למסורבלת, לפחות קריאה ולקשה יותר לשינוי (למשל אם מספר המועמדים ישתנה).

לכן ניצור מערך שאורכו כמספר המועמדים, וכל תא בו יְיַצג מונה.

נזדקק למשתנים הבאים:

numOfSingers – מטיפוס שלם, שומר את מספר המתמודדים בתחרות
votes – מערך מטיפוס שלם, מערך מונים באורך numOfSingers למניית ההצבעות עבור כל מועמד ומועמד
vote – מטיפוס שלם, שומר הצבעה תורנית מהקלט
min – מטיפוס שלם, לשמירת ההצבעה המצטברת הנמוכה ביותר
minSinger – מטיפוס שלם, שומר את מספר המתמודד שקיבל את מספר ההצבעות הנמוך ביותר

האלגוריתם

לצורך ביצוע תת-משימה 2 יש להשתמש בלולאה, אשר בכל סיבוב בו תיקלט הצבעה נוספת מהקלט, ובהתאם לערכה יגדל ערכו של המונה המתאים. מאחר שמספר ההצבעות אינו ידוע מראש, ומאחר שסוף רשימת ההצבעות מצוין בזקיף, נשתמש בלולאת while שהתנאי בה מתייחס לזקיף.

לצורך ביצוע תת-משימה 3 נחפש את האיבר הקטן ביותר במערך המונים.
שימו ♥: יש להציג כפלט את **מספרו הסידורי** של המתמודד שקיבל את מספר הקולות הקטן ביותר, ולא את מספר הקולות שקיבל. לכן במקרה זה נשתמש בתבנית של מציאת מקום המינימום.

1. *קלוט את מספר המתמודדים ב-*numOfSingers
2. *קלוט מספר מומחה ב-*vote
3. *כל עוד* vote ≠ -1 *בצע:*
   1. *הגדל ב-1 את המונה (במערך) של המתמודד שמספרו* vote
   2. *קלוט מספר מומחה ב-*vote
4. *אתחל את* min *במרכיב המונה של המתמודד הראשון*
5. *אתחל את* minSinger *ב-1*
6. *עבור כל* i *שמן מ-2 עד* numOfSingers *בצע:*
   1. *אם מרכיב המונה של מתמודד מספר* i *קטן מ-*min
      1. *הצב ב-*min *את מרכיב המונה של המתמודד שמספרו* i
      2. *הצב את* i *ב-*minSinger
7. *הדפס את* minSinger

יישום האלגוריתם

חשוב לשים לב כי המספרים הסידוריים של המועמדים מתחילים ב-1, בעוד שמספור איברי המערך מתחיל מ-0. לכן למשל היישום של הוראה 3.1 באלגוריתם הוא

```
votes[vote - 1]++;
```

והיישום של הוראה 6.1 הוא

```
if (votes[i - 1] < min)
```

התוכנית המלאה

```
/*  קלט: מספר המועמדים בתוכנית "כוכב נולד" והצבות הצופים למועמדים
      פלט: המספר הסידורי של המתמודד המפסיד  */
import java.util.Scanner;
public class IsraeliIdol
{
    public static void main (String[] args)
    {
        Scanner in = new Scanner(System.in);
        // הגדרת משתנים
        int numOfSingers;
        int[] votes;
        int vote;
        int min;
        int minSinger;
```

```
        // קליטת מספר מועמדים והקצאת גודל מערך מתאים
        System.out.print("Enter number of contestants: ");
        numOfSingers = in.nextInt();
        votes = new int[numOfSingers];
        // אתחול מערך המונים
        for(int i = 0; i < numOfSingers; i++)
            votes[i] = 0;
        // קליטת בחירת הצופים בלולאת זקיף
        System.out.print("Enter a vote, end the list with -1 ");
        vote = in.nextInt();
        while (vote != -1)
        {
            // הוספת קול נוסף למונה המתאים
            votes[vote - 1]++;
            System.out.print("Enter a vote, end the list with -1 ");
            vote = in.nextInt();
        }
        // אתחול משתני מינימום ומקום המינימום
        min = votes[0];
        minSinger = 1;
        // חיפוש ערך מינימלי ושמירת ערכו וערך המציין שלו
        for(int i = 2; i <= numOfSingers; i++)
        {
            if (votes[i - 1] < min)
            {
                min = votes[i - 1];
                minSinger = i;
            } // if
        } // for
        // פלט
        System.out.println("contestant number " + minSinger +
                                                    " is going home");
    } //main
} //class IsraeliIdol
```

שימו ♥: כשם שיש לאתחל משתנה מסוג מונה לאפס, כך גם יש לאתחל כל איבר ואיבר במערך מונים ב-0. אנו מבצעים אתחול כזה במפורש, למרות שיכולנו להסתמך על כך שבשפת Java איבריו של מערך מטיפוס מספרי מאותחלים אוטומטית ב-0. אתחול מפורש תורם לבהירות התוכנית.

ועוד שימו ♥: את תבנית מציאת מקום המינימום ניתן גם ליישם בצורה מעט שונה, הנוחה לעבודה עם מערכים. למעשה ביישום זה נשמר רק מקום האיבר המינימלי ולא ערכו, ואנו מסתמכים על כך שבמערך קל לברר את ערכו של איבר כאשר ידוע המציין שלו:

```
// חיפוש ערך מינימלי ושמירת ערך המציין שלו
for(int i = 2; i <= numOfSingers; i++)
    if (votes[i - 1] < votes[minSinger])
        minSinger = i;
System.out.println("contestant number " + minSinger +
                                                    " is going home");
```

סוף פתרון בעיה 5

להעמקה בתבנית ***מערך מונים*** פנו לסעיף התבניות המופיע בסוף הפרק.

שאלה 10.35

עקבו אחר התוכנית `IsraeliIdol` בעזרת טבלת מעקב, עבור הקלט 1- 2 4 4 2 1 1 3 2 4 (משמאל לימין).

שאלה 10.36

בכיתה נערכה הצבעה לנציג הכיתה למועצת תלמידים. כל תלמיד הקליד במחשב את בחירתו על פי המפתח הבא: עבור רותי יוקלד הערך 1, עבור אלי – הערך 2, עבור אביב – הערך 3, ועבור אופיר – הערך 4.

פתחו אלגוריתם המקבל כקלט את מספר התלמידים בכיתה ואת רשימת ההצבעות (הצבעה אחת מכל תלמיד בכיתה) ומציג כפלט את שם המועמד המנצח. ישמו את האלגוריתם בשפת Java.

שאלה 10.37

פתחו אלגוריתם הקולט מספר חיובי שלם, ואחר כך מדמה סדרת הטלות קובייה, שאורכה כערך המספר שנקלט. פלט האלגוריתם הוא מספר הפעמים שהוגרלה כל אחת משש התוצאות האפשריות להטלת קובייה. ישמו את האלגוריתם בשפת התכנות Java.

שאלה 10.38

ששת חברי קבוצה בצופים מצאו דרך חדשה להגרלה. הם מחלקים ביניהם את המספרים מ-1 עד 6, ומטילים קובייה עד אשר מספר כלשהו מוגרל פעמיים. החבר בעל המספר שהוגרל פעמיים הוא הזוכה. פתחו אלגוריתם אשר מדמה את תהליך זריקת הקובייה ומדפיס את המספר הזוכה. ישמו את האלגוריתם בשפת התכנות Java.

שאלה 10.39

בפתרון הבעיה הקודמת השתמשנו במערך מונים, אך בכל מהלך הביצוע ערכיו לא עברו אף פעם את הערך 2. שינוי אחד הערכים במערך ל-2 הביא לסיום האלגוריתם. שנו את הפתרון כך שבמקום מערך מטיפוס שלם, ישתמש במערך מטיפוס בוליאני. ישמו את הפתרון החדש בשפת Java.

שאלה 10.40

במשחק "תפוס את הצבע" כל שחקן בוחר צבע שונה. הקלט הוא מספר השחקנים והצבע שכל שחקן בחר. לאחר מכן עבור כל הקשה על מקש enter מוגרל צבע כלשהו (מתוך הצבעים שנבחרו). המשחק מסתיים כאשר מקישים על האות Q. בסוף המשחק בודקים מהו הצבע שהוגרל הכי הרבה פעמים ומציגים את הצבע הזוכה כפלט. כתבו תוכנית שתדמה את "תפוס את הצבע".

הדרכה: מאתחלים מערך צבעים לפי הקלט שמתקבל מהשחקנים, לדוגמה עבור הקלט: 4, אדום, ירוק, צהוב, סגול, יתקבל המערך הבא:

סגול	צהוב	ירוק	אדום

את הצבעים יש להגריל מתוך הצבעים שנקלטו.

בנוסף, יש להגדיר מערך מונים לספירת ההגרלות לכל צבע.

10.5 מערך צוברים

באותו אופן שמימשנו מערך של מונים, ניתן לממש מערך של צוברים. במערך צוברים כל איבר הוא צובר בפני עצמו, אך יש קשר בין משימות הצבירה שהצוברים השונים במערך אחראים להן. לצורך פתרון השאלה הבאה יש להשתמש במערך צוברים, שהעבודה עמו מהווה אף היא תבנית שימושית.

שאלה 10.41

במשחק קלפים משתתפים ארבעה שחקנים ולהם מספרים סידוריים מ-1 עד 4. בכל סבב מוכרזים השחקנים שזכו במקום הראשון, במקום השני ובמקום השלישי. השחקן הנותר נחשב כמפסיד. המנצח במקום הראשון מקבל 7 נקודות, השחקן שבמקום השני מקבל 3 נקודות, השחקן שבמקום השלישי אינו מקבל נקודות והמפסיד **מאבד** 4 נקודות.

המנצח במשחק כולו הוא זה שקיבל את מרב הנקודות בסיום כל הסבבים.

פתחו אלגוריתם אשר מקבל כקלט את מספר הסבבים, ולאחר מכן מקבל עבור כל סבב את מספרי השחקנים שהגיעו למקום הראשון, השני והשלישי. האלגוריתם מציג כפלט את מספרו של השחקן המנצח במשחק. ישמו את האלגוריתם בשפת Java.

שימו ♥ : מאחר שניתן גם לאבד נקודות, ייתכן ששחקן יגיע למספר נקודות שלילי.

שאלה 10.42

בספריית הווידיאו יש לכל סרט קוד בן 4 ספרות. הספרה השמאלית ביותר מסמנת את קוד המחלקה (בין 1 ל-6), שתי הספרות האמצעיות מסמנות את קוד הסרט (בין 1 ל-99) והספרה הימנית ביותר מסמנת את מספר העותקים בספרייה (בין 1 ל-9).

פתחו אלגוריתם המקבל כקלט רשימה של הקודים של כל הסרטים בספרייה המסתיימת במספר 0. פלט האלגוריתם הוא:

א. קוד המחלקה שיש בה את מספר הסרטים (השונים) הגדול ביותר.

ב. קוד המחלקה שיש בה את מספר עותקי הסרטים הגדול ביותר.

ישמו את האלגוריתם בשפת Java.

שאלה 10.42 עוסקת במציאת איבר שכיח במערך. להעמקה בתבנית ***חישוב שכיח*** פנו לסעיף התבניות המופיע בסוף הפרק.

להעמקה בתבנית ***מערך צוברים*** פנו לסעיף התבניות המופיע בסוף הפרק.

10.6 יעילות מקום

בכל הבעיות שפתרנו בפרק זה נעזרנו במערך, ועבור כל בעיה ראינו את נחיצות השימוש במערך במהלך הפתרון. בסעיף הבא נתמקד באבחנה בין בעיות אשר בפתרונן נחוץ מערך ובין בעיות אשר ניתן לפתור גם ללא מערך, ונבין מדוע לא כדאי להשתמש במערך אם אין בו צורך.

בעיה 6

מטרת הבעיה ופתרונה: הצגת המדד של יעילות מקום, והשוואה של אלגוריתמים ביחס ליעילות מקום. מבט נוסף אל יעילות מבחינת זמן-ביצוע.

ערוצי הכבלים עורכים מדי פעם סקרים כדי לברר את אחוזי הצפייה בתוכניותיהם השונות. פרסום תוצאות סקר כזה כולל רשימת תוכניות יחד עם אחוז צפייה בכל תוכנית.

א. פתחו אלגוריתם אשר הקלט שלו הוא מספר של תוכניות (שלם וחיובי), ולאחר מכן רשימה של אחוזי צפייה בתוכניות ואורכה כערך הנתון הראשון. הפלט הוא מספר התוכניות שאחוז הצפייה בהן גבוה מממוצע אחוזי הצפייה.

ב. פתחו אלגוריתם אשר הקלט שלו הוא מספר של תוכניות (שלם וחיובי), ולאחר מכן רשימה של אחוזי צפייה בתוכניות ואורכה כערך הנתון הראשון. הפלט הוא מספר התוכניות שאחוז הצפייה בהן גבוה מ-20%.

ג. האם תשובתכם לסעיף א תשתנה אם ידוע כי רשימת אחוזי הצפייה נתונה בסדר יורד (כלומר תחילה נתון אחוז הצפייה בתוכנית האהודה ביותר, אחר-כך אחוז הצפייה התוכנית האהודה הבאה וכך הלאה, עד אחוז הצפייה בתוכנית שנצפית הכי פחות).

נתחיל בסעיף א.

הבעיה המוגדרת בסעיף א זהה לחלוטין לבעיה 2. יש לקלוט אורך של רשימת נתונים לא ריקה, לקלוט את הרשימה עצמה, לחשב את ממוצע הערכים, ולמנות את הערכים הגדולים מהממוצע. מאחר שכבר פתרנו בעיה זהה לה, לא נחזור כעת על הפתרון בפירוט, רק נזכיר שהפתרון משתמש במערך לשמירת ערכי הקלט (במקרה זה, רשימת אחוזי הצפייה).

נעבור כעת לסעיף ב.

ניתוח הבעיה בעזרת דוגמאות

נניח כי הקלט הוא 31 18 13 21 4. במקרה זה הפלט הוא 2, כי אחוז הצפייה בשתי תוכניות (הראשונה והרביעית) גבוה מ-20%. אין אנו נדרשים לחשב ממוצע של אחוזי הצפייה.

פירוק הבעיה לתת-משימות

ניתן להציג פירוק הדומה לפירוק המתאים לסעיף א:

1. קליטת מספר התוכניות
2. קליטת אחוזי הצפייה ושמירתם
3. השוואת כל אחוז צפייה ל-20 ומנייתו אם הוא גדול מ-20

פירוק זה אכן מורה על שימוש במערך. קודם כל נשמרים הנתונים, ואחר כך הם מעובדים (במקרה זה, מושווים ל-20 ונמנים בהתאם לתוצאת ההשוואה).

? האם ניתן להציג פירוק פשוט יותר, שאינו מחייב שימוש במערך?

כן! ניתן לעבד כל נתון נקלט מיד עם קליטתו, להשוותו ל-20, ולהחליט אם למנות אותו או לא, ובעצם אין צורך בשמירת ערכי הקלט לאחר מכן. לכן נקבל את הפירוק הבא לתת-משימות:

1. קליטת מספר התוכניות
2. קליטת אחוזי הצפיה, ועבור כל אחוז צפייה נקלט, השוואתו ל-20 ועדכון המונה בהתאם

בחירת משתנים

מאחר שכאמור אין צורך במערך נשתמש במשתנים הבאים:

numOfPrograms – מטיפוס שלם, ישמור את מספר התוכניות.
rate – מטיפוס ממשי, ישמור את אחוז הצפייה התורן בקלט.

above20Counter – מטיפוס שלם, מונה לשמירת מספר אחוזי הצפייה הגבוהים מ-20.

האלגוריתם

1. *קלוט את מספר התוכניות ב-*numOfPrograms
2. *אתחל את ערכו של* above20Counter *ל-0*
3. *עבור כל* i *שמקבל ערכים מ-0 עד* numOfPrograms *בצעד 1:*
   3.1. *קלוט את אחוז הצפייה הבא ב-*rate
   3.2. *אם* rate > 20
      3.2.1. *הגדל את ערכו של* above20Counter *ב-1*
4. *הדפס את ערכו של* above20Counter

אם כך, למרות שהבעיה המוגדרת בסעיף א והבעיה המוגדרת בסעיף ב נראות דומות, הרי לפתרון סעיף א יש צורך במערך ואילו לפתרון סעיף ב אין צורך במערך.

מדוע יש בכך חשיבות? כידוע, בעת ביצוע תוכנית אשר יש בה שימוש במערך, מוקצה שטח זיכרון המספיק עבור כל איברי המערך. מספר תאי הזיכרון המוקצים למערך הינו מרכיב משמעותי בקביעת גודל הזיכרון הכולל הדרוש לביצוע התוכנית. כיוון שמשאבי הזיכרון של המחשב מוגבלים, נשתדל לפתח אלגוריתמים שגודל המקום הדרוש לביצועם הוא מצומצם במידת האפשר. כלומר נשתדל לפתח אלגוריתמים יעילים עד כמה שניתן מבחינת מקום בזיכרון. זאת בנוסף להקפדה על יעילות מבחינת זמן-ביצוע, כפי שהוצג בפרק 8.

כאשר יש שני אלגוריתמים שונים לפתרון אותה בעיה, ובאחד מהם משתמשים במערך ובאחר אין משתמשים במערך, נאמר שהאלגוריתם האחר **יעיל יותר מבחינת מקום בזיכרון**.

נעבור לפתרון סעיף ג.

ניתוח הבעיה בעזרת דוגמאות

גם בסעיף זה יש לפתח אלגוריתם הפותר את הבעיה שהוצגה בסעיף א. אבל שלא כמו בסעיף א, נתון לנו מאפיין נוסף של הקלט: רשימת אחוזי הצפייה נתונה בסדר יורד.

הנה קלט לדוגמה העונה להגדרת הבעיה: 3 15 29 30 41 5.

ממוצע הערכים הנתונים הוא 23.6. הערכים הגבוהים מן הממוצע הם 41, 30 ו-29. ניתן לראות כי מאחר שרשימת הערכים נתונה בסדר יורד, כל הערכים הגבוהים מן הממוצע נמצאים בתחילת הרשימה.

בדיוק כמו בסעיף א, את ההשוואה לממוצע ניתן לבצע רק לאחר חישוב הממוצע, כלומר רק לאחר קליטת כל הערכים.

? האם ניתן לפתור סעיף זה ללא מערך?
לא. מאחר שההשוואה לממוצע יכולה להיעשות רק אחרי סיום הקלט, יש לשמור את כל הערכים הנקלטים, ולשם כך נחוץ מערך.

? האם נוכל בכל זאת לשפר את הפתרון בשימוש במאפיין הקלט הנוסף?
בעזרת דוגמת הקלט שניתחנו ראינו כי הערכים הגדולים מן הממוצע נמצאים כולם בתחילת הרשימה, ולכן גם יישמרו ראשונים בשלב שמירת הקלט. לכן, בשלב ההשוואה לממוצע לא נצטרך לעבד שוב את כל ערכי הקלט, אלא רק את אלה המופיעים בהתחלה.

לכן, במקום להשתמש בהוראה לביצוע-חוזר מספר פעמים ידוע מראש שתעבור על **כל** האיברים שנשמרו, נוכל להשתמש בהוראה לביצוע-חוזר-בתנאי. התנאי יאפשר את המשך ביצוע הלולאה כל עוד האיבר התורן גדול מהממוצע.

ניתוח זה מראה כי הפתרון שניתן לא יהיה יעיל יותר מהפתרון של סעיף א מבחינת מקום בזיכרון, אבל יהיה בו שיפור מבחינת יעילות זמן: מניית הערכים הגדולים מהממוצע תעבור רק על חלק מהאיברים שנשמרו ולא על כולם.

פירוק הבעיה לתת-משימות

בהתאם לניתוח שערכנו, נפרק את הבעיה באופן הבא:

1. קליטת מספר התוכניות
2. קליטת אחוזי הצפייה, שמירתם וצבירתם
3. חישוב הממוצע
4. מעבר על הערכים הגדולים מן הממוצע תוך כדי מנייתם
5. הצגת ערך המונה

בחירת משתנים

numOfPrograms – מטיפוס שלם, לשמירת מספר התוכניות

rating – מערך מטיפוס ממשי בן numOfPrograms איברים לשמירת אחוזי הצפייה הנתונים, והם יהיו מסודרים בו בסדר יורד

sumOfRating – מטיפוס ממשי, צובר שישמור את סכום אחוזי הצפייה

averageOfRating – מטיפוס ממשי, לשמירת ממוצע אחוזי הצפייה

aboveAverageCounter – מטיפוס שלם, מונה את אחוזי הצפייה הגבוהים מאחוז הצפייה הממוצע.

התוכנית המלאה

```
/*
(שלם חיובי) קלט: מספר של תוכניות
ורשימת אחוזי צפייה בתוכניות, נתונים בסדר יורד
פלט: מספר אחוזי הצפייה הגבוהים מאחוז הצפייה הממוצע
*/
import java.util.Scanner;
public class AboveAverage
{
    public static void main (String[] args)
    {
        Scanner in = new Scanner(System.in);
        // הגדרת משתנים
        int numOfPrograms;              // מספר התוכניות
        double[] rating;                // מערך אחוזי הצפייה
        double sumOfRating = 0;         // צובר אחוזי הצפייה
        double averageOfRating ;        // ממוצע אחוזי הצפייה
        int aboveAverageCounter = 0;    // מונה הגבוהים מהממוצע
        // קלט וצבירה
        System.out.print("Enter number of programs: ");
        numOfPrograms = in.nextInt();
        rating = new double[numOfPrograms];
```

```
        for (int i = 0; i < numOfPrograms; i++)
        {
            System.out.print("Enter rating percentage");
            rating[i] = in.nextInt();
            sumOfRating = sumOfRating + rating[i];
        } // for
        // חישוב ממוצע
        averageOfRating = sumOfRating / numOfPrograms;
        // מניית הגבוהים מהממוצע
        int i = 0;
        while(rating[i] > averageOfRating)
        {
            i++;
            aboveAverageCounter++;
        } // while
        // פלט
        System.out.println("The rating of " + aboveAverageCounter +
                                          " programs is above average");
    } // main
} //class AboveAverage
```

שימו ♥ העובדה שאחוזי הצפייה מסודרים בסדר יורד מאפשרת לסיים את לולאת ה-while מבלי לעבור על כל איברי המערך. התנאי `rating[i] > averageOfRating` משמש כתנאי כניסה ללולאה ולכן המנייה מסתיימת ברגע שנמצא אחוז צפייה שאינו גבוה מהממוצע.

שאלה 10.43

מדוע מובטח כי ההוראה לביצוע-חוזר-בתנאי, המונה את הערכים הגבוהים מהממוצע, אינה לולאה אינסופית (במילים אחרות מדוע מובטח כי תנאי הכניסה יתקיים בשלב כלשהו)?

סוף פתרון בעיה 6

מפתרון בעיה 6 למדנו לקח חשוב:

יש לנצל עד כמה שניתן את מאפייני הקלט-פלט כדי לנסח אלגוריתם יעיל, הן מבחינת מקום בזיכרון והן מבחינת זמן-ביצוע.

ייתכנו שתי בעיות אלגוריתמיות הנראות דומות, לפחות במבט ראשון, ופתרונות יעילים עבורן נבדלים זה מזה הן מבחינת מקום והן מבחינת זמן. בפרט, ייתכן שעבור פתרון בעיה אחת נחוץ מערך ועבור פתרון הבעיה האחרת לא נחוץ מערך.

שאלה 10.44

נתונה רשימת קלט של 40 ציונים. סמנו אם לביצוע כל אחד מהחישובים הבאים נחוץ מערך או לא:

א. מספר הציונים הגבוהים מ-80 ..... נחוץ / לא נחוץ
ב. מספר הציונים השווים לציון הראשון ברשימה ..... נחוץ / לא נחוץ
ג. מספר הציונים הגבוהים מן הציון האחרון ברשימה ..... נחוץ / לא נחוץ
ד. מספר הציונים הנמוכים מן הציון הלפני אחרון ברשימה ..... נחוץ / לא נחוץ
ה. מספר הציונים הנמוכים מן הציון הראשון ומן הציון האחרון ..... נחוץ / לא נחוץ
ו. מספר הציונים הנמוכים מממוצע הציונים ..... נחוץ / לא נחוץ
ז. מספר הציונים הגבוהים מהממוצע של שני הציונים הראשונים ..... נחוץ / לא נחוץ

ח. מספר הציונים הגבוהים מהממוצע של שני הציונים האחרונים ..... נחוץ / לא נחוץ

שאלה 10.45

בבעיה 3 בפרק, הקלט כלל רשימה של 30 תווים ('T' או 'F') אשר ציינו מצב לוח, ואחר כך מספר שציין מקום של שחקן על הלוח. בפתרון הבעיה נעשה שימוש במערך בן 30 איברים לשמירת מצב הלוח.

נניח שמקום השחקן על הלוח יינתן כקלט **לפני** הרשימה המתארת את מצב הלוח (ולא אחריה). האם גם במקרה זה נחוץ מערך? הסבירו את תשובתכם.

שאלה 10.46

עודד המציא משחק חדש. הוא מציג לפני חבריו קופסת מטבעות סגורה. כל אחד מחבריו רושם ניחוש של מספר המטבעות בקופסה. לאחר מכן עודד סופר את מספר המטבעות בקופסה. המטבעות מחולקים לכל מי שניחש את המספר המדויק של המטבעות. שארית המטבעות נשארת אצל עודד.

א. פתחו אלגוריתם אשר מקבל כקלט מספר המציין את מספר החברים המשתתפים במשחק, אחר כך את רשימת הניחושים, ניחוש עבור כל חבר המשתתף במשחק, ולבסוף את מספר המטבעות שבקופסה. פלט האלגוריתם הוא מספריהם הסידוריים ברשימה של החברים שניחשו את מספר המטבעות המדויק, וכן מספר מטבעות שיקבל כל זוכה.
ישמו את האלגוריתם בשפת Java.
שימו ♥: היזהרו מחלוקה ב-0!

ב. ענו על סעיף א כאשר סדר נתוני הקלט שונה: תחילה נתון מספר המטבעות שבקופסה, אחריו מספר המשתתפים במשחק, ולבסוף רשימת הניחושים.

שאלה 10.47

בכספת של בנק שמורים יהלומים על מדפים הממוספרים מ-1 עד 100.

נתונה כקלט רשימת ערכים המבטאים את מצב מדפי הכספת, כך שהערך ה-i ברשימה מבטא את מצב המדף ה-i (מכיל יהלומים או לא).

ציינו עבור כל אחד מהחישובים הבאים אם נחוץ מערך לצורך ביצועו או לא. נמקו את תשובותיכם.

א. מספר המדפים המכילים יהלומים, ומספר המדפים ללא יהלומים..... נחוץ / לא נחוץ
נימוק: _____

ב. מספר המדפים שמצבם שווה למצב המדף הראשון ..... נחוץ / לא נחוץ
נימוק: _____

ג. מספר המדפים שמצבם שווה למצב המדף האחרון ..... נחוץ / לא נחוץ
נימוק: _____

ד. מספרם הסידורי של המדפים שמצבם שווה למצב המדף הראשון ..... נחוץ / לא נחוץ
נימוק: _____

ה. מספרם הסידורי של המדפים שמצבם שווה למצב המדף האחרון ..... נחוץ / לא נחוץ
נימוק: _____

בעזרת החומר שלמדנו בפרק 10 ניתן לפתור גם בעיות המשתמשות בתבניות ***הזזה מעגלית בסדרה, הזזה של תת-סדרה*** ו***היפוך של תת-סדרה***. להעמקה בתבניות אלו פנו לסעיף התבניות המופיע בסוף הפרק.

שאלות נוספות

1. במשחק כדורסל ניתן לקלוע סל המזכה את הקבוצה בנקודה אחת, בשתי נקודות או בשלוש נקודות. בכל קבוצה חמישה שחקנים אשר ממוספרים מ-1 עד 5. בעת משחק מוזנות תוצאות הקליעות בזוגות מספרים: מספר השחקן הקולע ומספר הנקודות שקיבל עבור הקליעה. בסיום המשחק מוקלד 0 עבור מספר השחקן. מנהל הקבוצה מעוניין לערוך בדיקות סטטיסטיות על הישגי הקבוצה בעת משחק:
   א. מה מספר השחקן או השחקנים שצברו את מספר הנקודות הגדול ביותר?
   ב. כמה נקודות צברה הקבוצה במשך כל המשחק?
   ג. מה סוג הקליעה הנפוץ ביותר (קליעות שזיכו בנקודה אחת, ב-2 נקודות או ב-3 נקודות)
   ד. איזה שחקן או שחקנים לא צברו שום נקודה במשך כל המשחק?

   לדוגמה: הקלט (משמאל לימין) 0 3 2 2 5 1 3 2 2 מתאר כי שחקן מספר 2 קלע סל של 2 נקודות, שחקן מספר 3 קלע סל של נקודה 1, שחקן מספר 5 קלע סל של 2 נקודות ושחקן מספר 2 קלע סל של 3 נקודות. הפלט עבור קלט זה הוא: שחקן מספר 2 צבר את מספר הנקודות הגדול ביותר, במשך כל המשחק הקבוצה צברה 8 נקודות, סוג הקליעה הנפוץ ביותר הוא 2, ושחקנים מספר 1 ו-4 לא צברו אף נקודה במשך המשחק.

   ? חשבו לגבי כל סעיף אם אתם זקוקים למערך מונים, למערך צוברים או שאינכם זקוקים כלל למערך.

2. נתון המערך `arr` ובו ערכים שלמים ונתון הקטע הבא המשתמש במערך מונים:

```
int[] counts = new int[2];
int element;
for (i = 0; i < arr.length; i++)
{
    element = arr[i];
    counts[element % 2]++;
}
for (i = 0; i < counts.length; i++)
    System.out.println(counts[i]);
```

   שימו ♥: קטע תוכנית זה משתמש במערך מונים.
   א. מה יוצג כפלט עבור המערך `arr` הבא:

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	Arr[7]
90	68	198	5	11	34	89	6

   ב. תנו דוגמה למערך `arr` בגודל 10, שעבורו יוצגו הערכים: 3 7.
   ג. מהי מטרת קטע התוכנית?

3. כתבו אלגוריתם שהקלט שלו הוא סדרת תווים המסתיימת בתו '*' והפלט שלו הוא מספר המופעים של כל אחת מאותיות הא"ב האנגלי הקטנות.

   שימו ♥: לצורך כתיבת האלגוריתם, עליכם להשתמש במערך מונים.
   א. מהו הביטוי החשבוני שהשתמשתם בו להתאמת אותיות הא"ב למצייני המערך?
   ב. ישמו את האלגוריתם בשפת Java.

4. במוסד לביטוח רפואי נעשה מחקר על צריכת 150 תרופות בידי החולים המבוטחים. לכל תרופה יש מספר סידורי בין 1 ל-150.

א. פתחו אלגוריתם שהקלט שלו הוא סדרת שלשות של ערכים, וכל שלשה מייצגת גיל של מבוטח, מספר סידורי של התרופה שנצרכה ואת הכמות שלה (מספר יחידות). סדרת הקלט מסתיימת עם קליטת הזקיף 1- כגיל המבוטח. הפלט של האלגוריתם הוא קבוצת הגילאים הצורכת את כמות התרופות הכוללת הגדולה ביותר, מבין 4 קבוצות הגילאים הבאות: קבוצת בני 0 עד 10, קבוצת בני 11 עד 30, קבוצת בני 31 עד 50 וקבוצת בני 51 ומעלה. ייתכן שיש יותר מקבוצת גיל אחת הצורכת את כמות התרופות הכוללת הגדולה ביותר. ישמו את האלגוריתם בשפת Java.

שימו ♥: לצורך פתרון סעיף זה אין צורך להבחין בין סוגי התרופות השונים.

ב. הרחיבו את האלגוריתם כך שיוצגו כפלט גם מספרי התרופות שלא נצרכו כלל.

ג. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם וכיצד שילבתם ביניהן.

5. יותם המדריך החליט לארגן לחניכיו פעולה בסגנון "חפש את המטמון". המטמון הוחבא בבניין שבט הצופים שבו 50 חדרים הממוספרים מ-0 עד 49. בכל חדר נמצא פתק שבו רשום מספר החדר הבא שבו יש להמשיך את החיפוש או נמצא המטמון עצמו. יותם הכין את הפתקים ואת המטמון מבעוד מועד ורצה לפזר אותם בחדרים. לשם כך הוא הכין מערך בשם rooms בגודל 50 תאים. בכל תא במערך רשום מספר בין 1- ל-49, המספר 1- מציין שהמטמון נמצא בחדר זה. כל מספר אחר מכוון לחדר שיש לעבור אליו להמשך החיפוש. כמו כן יותם אומר לחניכיו את מספר החדר הראשון שיש להתחיל ממנו את החיפושים. לדוגמא: עבור 16 חדרים וחיפוש המתחיל בחדר מספר 3:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
rooms	14	11	10	8	-1	6	-1	2	0	13	4	-1	15	1	7	5

הקבוצה תתחיל בחיפוש בחדר 3, תעבור לחדר 8, משם לחדר 0, וכך הלאה עד אשר תמצא את המטמון בחדר מספר 4. כתבו קטע תוכנית המשתמש במערך rooms ובמספר החדר שבו יש להתחיל את החיפוש ומציג כפלט את מיקומו של המטמון. ניתן להניח שיותם פיזר את הפתקים בצורה כזו שהאלגוריתם מסתיים. יש לסרוק את המערך לפי סדר החיפוש שהכתיב יותם.

סיכום

בפרק זה הוצגו בעיות שפתרונן מצריך שמירה של סדרת ערכים מאותו טיפוס. ניתן לשמור סדרה כזאת כמערך (array).

מערך הוא אוסף סדור של משתנים מאותו טיפוס, הקשורים זה לזה ולהם שם משותף.

איבר במערך הוא משתנה לכל דבר, כלומר ניתן לשמור בו ערכים ולקרוא את הערכים השמורים בו.

שמו של איבר במערך מורכב משם המערך וממספרו הסידורי של האיבר במערך. למשל arr[3]. במקרה זה, שם המערך הוא arr ו-3 הוא ה**מציין** (אינדקס) שמפנה אותנו לאיבר שנמצא במקום מספר 3 במערך.

באלגוריתמים רבים נוח לבצע **סריקה של מערך**, תוך עיבוד איבריו, באמצעות לולאה.

אם נדרשת **סריקה מלאה** של כל איברי המערך, הרי מספר הסיבובים ידוע מראש ושווה לאורך המערך. גם סריקה לא רציפה, אבל במרווחים ידועים מראש, ניתנת לביצוע באמצעות לולאה שמספר הסיבובים בה ידוע מראש.

אם הסריקה מתבצעת בדילוגים שאינם קבועים ואינה רציפה, או אם ביצועה תלוי בתנאי, נשתמש בהוראה לביצוע-חוזר-בתנאי.

בטבלת מעקב אחר מהלך ביצוע תוכנית הכוללת מערך נכלול עמודות נפרדות עבור איברים שונים של המערך.

מערך יכול לתפקד כ**מערך מונים** או כ**מערך צוברים**, במקרים שיש צורך לתפעל במקביל כמה מונים או צוברים בעלי אופי משותף.

בעת הרצת תוכנית המשתמשת במערך, מוקצים תאים עבור כל איבר במערך. כיוון שמשאבי הזיכרון של המחשב מוגבלים, נשתדל לפתח אלגוריתמים אשר גודל המקום הדרוש לביצועם הוא מצומצם ככל האפשר.

ייתכנו שני אלגוריתמים שונים לפתרון אותה בעיה, אשר באחד יש שימוש במערך ובשני אין. נאמר שהאלגוריתם השני יותר **יעיל מבחינת מקום.**

יש לנצל עד כמה שניתן את מאפייני הקלט-פלט כדי לנסח אלגוריתם יעיל, הן מבחינת מקום בזיכרון והן מבחינת זמן-ביצוע.

ייתכנו שתי בעיות אלגוריתמיות הנראות דומות, לפחות במבט ראשון, אשר פתרונות יעילים עבורן נבדלים זה מזה הן מבחינת מקום והן מבחינת זמן. בפרט, יתכן שעבור פתרון בעיה אחת נחוץ מערך ועבור פתרון הבעיה האחרת לא נחוץ מערך.

סיכום מרכיבי שפת Java שנלמדו בפרק 10

הצהרה על מערך בשפת Java נכתבת כך:

```
טיפוס[] שם המערך;
```

ראשית שם הטיפוס, אחריו סוגריים מרובעים ואז שמו של המערך.

לפני שימוש במערך מוצהר יש לבצע עבורו **הקצאת שטח זיכרון**. הקצאת שטח למערך מתבצעת באמצעות הפעולה **new** שאחריה מופיע טיפוס איברי המערך, ואחריו מופיע בסוגריים מרובעים מספר איברי המערך:

```
new טיפוס[מספר הערכים];
```

ניתן לצרף את ההצהרה ואת ההקצאה להוראה אחת:

```
טיפוס[] שם המערך = new טיפוס[מספר הערכים];
```

בשפת Java כאשר מוקצה שטח זיכרון עבור מערך של איברים מטיפוס פשוט, מתבצע גם **אתחול אוטומטי** של כל איבריו. איברי מערך מספריים (שלמים או ממשיים) יאותחלו ב-0, איברי מערך בוליאני יאותחלו ב-false, ואיברי מערך תווי יאותחלו בתו הריק.

בשפת Java **מספור האיברים במערך** מתחיל ב-0, ולכן `arr[0]` מפנה אותנו לאיבר הראשון במערך, ו-`arr[3]` מפנה אותנו לאיבר הרביעי במערך.

לכל מערך מוגדרת **תכונת אורך** בשם `length` השומרת את אורכו. תכונה זו מקבלת את ערכה בעת הקצאת שטח זיכרון עבור המערך ואינה ניתנת לשינוי לאחר מכן. הפנייה לתכונה נעשית באמצעות סימון הנקודה, למשל כך: `arr.length`. אין להוסיף סוגריים משום שזו אינה פעולה!

יש להקפיד על פנייה לאיבר באמצעות מציין שערכו איננו חורג מתחום הערכים המותר, כלומר נע בין 0 לבין אורך המערך פחות אחת. **חריגה מגבולות המערך** תגרום לעצירת התוכנית עקב שגיאת ריצה.

סריקת איברי מערך יכולה להיעשות באמצעות לולאת **`for`**, אם ידועות מראש נקודות ההתחלה והסיום של הסריקה, וידועים מרווחי הדילוג (עבור סריקה לא רציפה). בלולאה כזאת, משתנה הבקרה של הלולאה משמש גם כמציין לציון מיקומו של האיבר הנסרק במערך. משתנה הבקרה מאותחל בתחילת הלולאה בערך של המציין של האיבר שהסריקה מתחילה ממנו. ערכו מתקדם בכל צעד בערך הרצוי (לסריקה רציפה הוא מתקדם ב-1 בכל פעם) עד שיגיע למיקומו של האיבר שהסריקה מסתיימת בו .

לסריקה רציפה של כל איברי המערך ניתן להיעזר בתכונת האורך כך שערכי משתני הבקרה ינועו מ-0 עד לאורך המערך פחות אחת, ויגדלו ב-1 בכל סיבוב.

לסריקה שאינה רציפה ובמרווחים שאינם ידועים מראש, או לסריקה התלויה בתנאי, יש להשתמש בלולאת **`while`**.

תבניות – פרק 10

פירוט מלא של התבניות ושל שאלות שבפתרונן יש שימוש בתבניות ניתן למצוא באתר הספר ברשת האינטרנט.

מערך מונים

בחרנו להציג לפניכם שימוש בביצוע-חוזר-בתנאי בהתבנית **מערך מונים**, ושימוש בביצוע-חוזר מספר פעמים ידוע מראש בתבנית **מערך צוברים**. אפשר כמובן להחיל את המקרה האחד על האחר ולהיפך בשינוי קל.

שם התבנית: **מערך מונים**
נקודת מוצא: תנאי סיום toEnd, סדרת ערכים, ביטוי חשבוני whichCounter המקשר בין ערך בסדרה למציין של המערך
מטרה: בניית מערך מונים עבור ערכי הסדרה, בעוד משך הבנייה תלוי בביטוי toEnd.
אלגוריתם:
1. אתחל את איברי המערך countElements ב-0
2. השמ את הערך הבא בסדרה ב-element
3. כל עוד לא מתקיים התנאי toEnd בצע:
   3.1. הגדל את countElements[whichCounter] ב-1
   3.2. השמ את הערך הבא בסדרה ב-element

מערך צוברים

שם התבנית: **מערך צוברים**

נקודת מוצא: אורך סדרת הערכים limit, סדרת ערכים, ביטוי חשבוני whichSum המקשר בין ערך בסדרה למציין של המערך

מטרה: בניית מערך צוברים עבור ערכי הסדרה שאורכה limit

אלגוריתם:

1. *אתחל את ערכי מערך הצוברים*
2. **בצע** limit **פעמים:**
   2.1. *השם את הערך הבא בסדרה ב-*element
   2.2. *השם את הערך הצבירה ב-*value
   2.3. *הוסף ל-*sumElements[whichSum] *את ערכו של* value

חישוב שכיח

שם התבנית: **חישוב שכיח**

נקודת מוצא: אורך סדרת הערכים limit, סדרת ערכים, ביטוי חשבוני whichSum המקשר בין ערך בסדרה למציין של המערך

מטרה: הצגה כפלט של הערך השכיח או של הערכים השכיחים בסדרת הקלט שאורכה הוא limit

אלגוריתם:

1. *אתחל את ערכי מערך המונים*
2. **בצע** limit **פעמים:**
   2.1. *השם את הערך הבא בסדרה ב-*element
   2.2. *הגדל את* sumElements[whichCount] *ב-1*
3. *אתחל את* max *בערכו של* countElements[0]
4. **עבור כל** i *שמן בתחום מ-*1 *עד אורך המערך* countElements *פעמ ו* **בצע:**
   4.1. **אם** countElements[i] > max
      4.1.1. *השם ב-*max *את הערך של* countElements[i]
5. **עבור כל** i *שמן בתחום מ-*0 *עד אורך המערך* countElements *פעמ ו* **בצע:**
   5.1. **אם ערכו של** countElements[i] *שווה ל-*max
      5.1.1. *הדפס את* i *כפלט*

הזזה מעגלית בסדרה

שם התבנית: **הזזה מעגלית שמאלה בסדרה**
נקודת מוצא: סדרת ערכים במערך elements שאורכו length
מטרה: הזזה מעגלית שמאלה של ערכי המערך
אלגוריתם:
1. השם ב-temp את elements[0]
2. **עבור כל** i שמקבל ערכים מ 0 עד length-2 **בצע:**
   2.1. השם ב-elements[i] את הערך של elements[i+1]
3. השם ב-elements[length-1] את הערך של temp

שם התבנית: **הזזה מעגלית ימינה בסדרה**
נקודת מוצא: סדרת ערכים במערך elements שאורכו length
מטרה: הזזה מעגלית ימינה של ערכי המערך
אלגוריתם:
1. השם ב-temp את elements[length-1]
2. **עבור כל** i שמקבל ערכים מ length-1 עד 1 (בסדר יורד) **בצע:**
   2.1. השם ב-elements[i] את הערך של elements[i-1]
3. השם ב-elements[0] את הערך של temp

הזזה של תת-סדרה

שם התבנית: **הזזה של תת-סדרה שמאלה**
נקודת מוצא: סדרת ערכים במערך elements באורך length, מקום k במערך ($0 \le k <$length-1)
מטרה: הזזה שמאלה של התת-סדרה הנמצאת במקומות k+1..length-1 למקומות k..length-2
אלגוריתם:
1. **עבור כל** i שמקבל ערכים מ k עד length-2 **בצע:**
   1.1. השם ב-elements[i] את הערך של elements[i+1]

שם התבנית: **הזזה של תת-סדרה ימינה**
נקודת מוצא: סדרת ערכים במערך elements באורך length, מקום k במערך ($0 <$ k $\le$ length-1)
מטרה: הזזה ימינה של התת-סדרה הנמצאת במקומות 0 .. k-1 למקומות 1 .. k
אלגוריתם:
1. **עבור כל** i שמקבל ערכים מ k עד 1 (בסדר יורד) **בצע:**
   1.1. השם ב-elements[i] את הערך של elements[i-1]

היפוך סדר האיברים בסדרה

שם התבנית: **היפוך סדר האיברים בסדרה**
נקודת מוצא: סדרת ערכים במערך elements
מטרה: היפוך סדר הערכים במערך שאורכו length
אלגוריתם:
1. *השם ב-*limit *את מנת החלוקה של* length *פריטים לשתי קבוצות*
2. *עבור כל* i *שווה בתחום מ-0 עד* limit-1 *בצע:*
   2.1. *החלף את הערכים של* elements[i] *ושל* elements[length-i-1]