מבחן 9

מבחן בגרות חיצוני - שנת 2002

פרטים כלליים

מועד הבחינה: קיץ תשס"ב, 2002

מספר השאלון: 899222

משך הבחינה: שלוש שעות

חומר עזר בשימוש: כל חומר עזר (חוץ ממחשב הניתן לתכנות)

המלצות: קרא המלצות לפני הבחינה ובדיקות אחרונות לפני מסירה (עמודים 8-10)

מבנה השאלון ומפתח הערכה

פרק ראשון	יש לענות על השאלות 1-5 לכל שאלה – 10 נקודות	סה"כ 50 נקודות
פרק שני	יש לענות על שתיים מהשאלות 6-8 לכל שאלה – 15 נקודות	סה"כ 30 נקודות
פרק שלישי	יש לענות על אחת מהשאלות 9-10 לכל שאלה – 20 נקודות	סה"כ 20 נקודות

תוכן עניינים של פתרון המבחן

פתרון 9 — פתרון מבחן בגרות תשס"ב - 2002

פתרון פרק ראשון - 50 נקודות

פתרון שאלה 1

א. טבלת מעקב עבור הקלט 1 , 2- , 10 , 15 , 8 (משמאל לימין)

	i	number	count	פלט
ערכים התחלתיים			0	
לולאה	1	8	1	
	2	15		
	3	10	2	
	4	-2	3	
	5	1	4	
				4

ב. האלגוריתם קולט 5 מספרים ומציג כפלט כמה מהם קטנים או שווים ל 10.

כאשר בשאלה מצוין "הסבר במשפט אחד מה מבצע האלגוריתם" הכוונה לתת כתשובה את הגדרת המשימה שלו. כלומר אם תקבל את המשפט שתרשום כתשובה תוכל להגדיר באופן מלא את האלגוריתם הנתון. שגוי לרשום תשובה אשר כוללת פירוט מילולי של חלק משורות האלגוריתם הנתון.

פתרון שאלה 2

```
/* קלט: התכנית מייצרת 57 זוגות מספרים אקראיים בתחום בין 1 ל- 18 */
/* פלט: התכנית מציגה את המספר הגדול בכל זוג, אם הם שווים – אחד מהם */
```

התכנית ב C

```
// q2t02.c;
Random randNum = new Random();
public static void Main (string[] args)
{
    int i, num1, num2;
    for ( i = 0; i < 57; i++)              /* לולאה להגרלת 57 זוגות */
    {
        num1 = randNum.Next (18) + 1;  /* הגרלת מספר אחד */
        num2 = randNum.Next (18) + 1;  /* הגרלת מספר שני */
        if (num1 > num2 )                  /* השוואה בין שני המספרים והדפסת הגדול ביניהם */
            Console.WriteLine (num1);
        else
            Console.WriteLine (num2);
    }
}
```

פתרון שאלה 3

א. עבור x=18 , y=13 יתקבל ביטוי אמת. להלן טבלת מעקב:

פלט	התנאי הלוגי המורכב (x>y) && (y>12)	y	x
ערך הביטוי אמת	(18>13) && (13>12) אמת = (אמת) וגם (אמת)	13	18

ב. עבור x=13 , y=10 או x=7 , y=10 יתקבל ביטוי שקר. להלן טבלת מעקב:

פלט	התנאי הלוגי המורכב (x>y) && (y>12)	y	x
ערך הביטוי שקר	(13>10) && (10>12) שקר = (שקר) וגם (אמת)	10	13
ערך הביטוי שקר	(7>10) && (10>12) שקר = (שקר) וגם (שקר)	10	7

שים ❤

מאחר והתנאי הוא תנאי מורכב עם הקשר הלוגי and, כדי שיתקבל הפלט 'ערך הביטוי: אמת' שני תת התנאים חייבים להתקיים – הערך של כל אחד מן הביטויים בנפרד צריך להיות true.

פתרון שאלה 4

א. טבלת מעקב עבור הקלט n=10 ועבור המערך a

פלט i , a[i]	n	הוראת התנאי a[i] + 2 = a[i+2]	i	
	10			ערכים התחלתיים
1 3		a[0] + 2 = a[2] 3 + 2 = 5 אמת	0	לולאה
1 3 2 8		a[1] + 2 = a[3] 18 + 2 = 20 אמת	1	
		a[2] + 2 = a[4] 5 + 2 = 2 שקר	2	
		a[3] + 2 = a[5] 20 + 2 = 4 שקר	3	
		a[4] + 2 = a[6] 2 + 2 = 5 שקר	4	
1 3 2 8 6 4		a[5] + 2 = a[7] 4 + 2 = 6 אמת	5	
		a[6] + 2 = a[8] 5 + 2 = 1 שקר	6	
		a[7] + 2 = a[9] 6 + 2 = 9 שקר	7	

ב. הגבול העליון של הלולאה נקבע להיות n-3 מאחר ובגוף הלולאה יש פנייה אל מצין במערך שערכו הוא i + 2. ללא המגבלה הזו הייתה נוצרת כאן חריגה מן הגבול העליון של המערך.

פתרון שאלה 5

א. דוגמה 1: num = 33 , dig = 3 הפונקציה מחזירה את הערך 2.
דוגמה 2: num = 51 , dig = 3 הפונקציה מחזירה את הערך 0.

ב. קטע תכנית בפסקל המייצג את גוף הפונקציה

גרסת פתרון 1:

שם המשתנה	טיפוס המשתנה	הסבר/תפקיד	תחום ערכים
dig1	שלם	ספרת העשרות במספר הדו-ספרתי	1..9
dig2	שלם	ספרת האחדות במספר הדו-ספרתי	0..9
sum	שלם	מונה את מספר הפעמים שהופיעה הספרה dig במספר הדו-ספרתי num	0..2

```
// q5t02.c
int how_many (int num, int dig)
{
        /*                    dig  וספרה  num  טענת כניסה: הפונקציה מקבלת מספר דו-ספרתי */
      /*  num  טענת יציאה: הפונקציה מחזירה את מספר הפעמים שהספרה dig מופיעה במספר */
    int dig1, dig2, sum;
    dig1 = num / 10;          /* ספרת העשרות */
    dig2 = num % 10;          /* ספרת האחדות */
    sum = 0;                  /* אתחול הערך המוחזר */
    if (dig1 == dig)          /* אם ספרת העשרות זהה ל- dig אזי קדם ב- 1 */
        sum = sum +1;
    if (dig2 == dig)          /* אם ספרת האחדות זהה ל- dig אזי קדם ב- 1 */
        sum = sum +1;
    return sum;               /*  השמת ערך הסכום בשם הפונקציה */
}
```

שים ♥ חובה לאתחל את sum ל- 0, מאחר והוא משמש כצובר, ובנוסף כאשר אף אחד מן התנאים לא מתקיים הערך המוחזר צריך להיות 0 – כי יש 0 פעמים dig בתוך num.

גרסת פתרון 2:

```
int how_many (int num, int dig)
{
   if ((num/10 == dig) && (num %10 == dig))   /*  אם ספרת האחדות וגם ספרת העשרות */
      return 2;                               /*  זהות ל- dig אזי החזר 2 */
   else
      if ((num/10 == dig) ||(num %10== dig))  /*  אם ספרת האחדות או ספרת העשרות */
         return 1;                            /*  זהות ל- dig אזי החזר 1 */
      else
         return 0;                            /* אחרת אף ספרה לא זהה ל- dig, החזר 0 */
}
```

פתרון פרק שני - 30 נקודות

ענה על שתי שאלות מבין השאלות 6-8

פתרון שאלה 6

טבלת משתנים

שם המשתנה	טיפוס המשתנה	הסבר/תפקיד	תחום ערכים
num	שלם	משתנה הקלט	חיובי גדול מאפס
counter	שלם	מונה את מספר המספרים הזוגיים	1..10
all	שלם	מונה את מספר המספרים שנקלטו	חיובי

// q6t02.c

```
/* קלט: התכנית קולטת מספרים שלמים עד קליטת 10 מספרים זוגיים */
/* פלט: התכנית מציגה כפלט את המספרים הזוגיים ואת מספר המספרים הכולל שנקלט */

public static void Main (string[] args)
{
    int num, counter, all;

    counter =0;                 /* אתחול מספר המספרים הזוגיים */
    all = 0;                    /* אתחול מספר המספרים הכולל */
    while (counter <10)         /* כל עוד לא התקבלו 10 מספרים זוגיים בצע הלולאה */
    {
       num = int.Parse(Console.ReadLine()); /* קלוט נתון תורן חדש */
       all ++;                  /* קדם מונה המספרים הכולל */
       if ((num % 2) == 0)      /* אם הקלט התורן הוא זוגי – מתחלק ב- 2 */
         {
         counter+=1;                        /* קדם מונה המספרים הזוגיים */
         Console.WriteLine ("even number: "+num);   /* הדפס את המספר הזוגי שנקלט */
         }
    }
    Console.WriteLine (all+" numbers were entered"); /* הדפס את סך כל המספרים
שנקלטו */
}
```

הסבר: לצורך פתרון הבעיה יש צורך בלולאת תנאי מאחר ולא ידוע מראש כמה נתונים יקלטו עד להשגת המטרה. הלולאה מתבצעת כל עוד לא התקבלו עשרה מספרים זוגיים כנדרש. כלומר, כל עוד מונה המספרים הזוגיים עוד לא הגיע ל-10. חשוב שהתנאי יהיה > ולא =>, מאחר וכאשר ערך המונה יהיה 10 משמעו: כבר נקלטו 10 מספרים זוגיים.

פתרון שאלה 7

מספר פתרונות

גרסת פתרון 1:

טבלת משתנים

שם המשתנה	טיפוס המשתנה	הסבר/תפקיד	תחום ערכים
numClass	שלם	קלט מספר כיתה	1..2
bottle	שלם	קלט מספר בקבוקים	שלם חיובי או 0
battery	שלם	קלט מספר סוללות	שלם חיובי או 0
points1	שלם	נקודות מצטברות של כיתה1	שלם חיובי או 0
points2	שלם	נקודות מצטברות של כיתה2	שלם חיובי או 0
i	שלם	מונה לולאת הקלט של התלמידים	0..67

// q7at02.c

```
/* קלט: התכנית קולטת שלשות של נתונים (מספר כיתה, מספר בקבוקי פלסטיק, מספר סוללות)
/*                עבור כל אחד מ 68  התלמידים בשתי כיתות                          */
/* פלט: התכנית מציגה כפלט את מספר הכיתה שצברה הכי הרבה נקודות, בשיוויון להדפיס - תקו  */
public static void Main(string[] args)
{
   int numClass, bottle, battery;
   int points1, points2, i;

   points1 = 0;                    /* אתחול צוברי הנקודות של שתי הכיתות */
   points2 = 0;
   for (i = 0; i < 68; i++)        /* לולאה לקליטת הנתונים מ- 68 התלמידים */
   {
       Console.WriteLine ("enter your class, number of bottles, number of batteries:");
       numClass = int.Parse(Console.ReadLine());
       bottle = int.Parse(Console.ReadLine());          /* קלט שלשת הנתונים הנדרשת */
       battery = int.Parse(Console.ReadLine());
       if (numClass == 1)                               /* אם התלמיד מכיתה1 */
          points1 += bottle*3 + battery*7;              /* קידום הנקודות של כיתה1 לפי הכללים */
       else                                             /* אחרת התלמיד מכיתה2 */
          points2 += bottle*3 + battery*7;              /* קידום הנקודות של כיתה2 לפי הכללים */
   }
   if ( points1 > points2)                              /* הדפסת הכיתה המנצחת */
       Console.WriteLine ("class 1 won");
   else
       if (points2 > points1)
           Console.WriteLine ("class 2 won");
       else
           Console.WriteLine ("TEKO");
}
```

שים ♥

♥ הגדרת הפלט רק בתום קליטת כל הנתונים וצבירת כל הנקודות – אחרי הלולאה.

- הפלט הוא מספר הכיתה ולא מספר הנקודות.

גרסת פתרון 2:

טבלת משתנים

שם המשתנה	טיפוס המשתנה	הסבר/תפקיד	תחום ערכים
numClass	שלם	קלט מספר כיתה	1..2
bottle	שלם	קלט מספר בקבוקים	שלם חיובי או 0
battery	שלם	קלט מספר סוללות	שלם חיובי או 0
points	מערך חד-ממדי	צובר את הנקודות של כיתה1 במציין 1 ושל כיתה 2 במציין 2	שלם חיובי או 0
i	שלם	מונה לולאת הקלט של התלמידים	0..67

// q7bt02.c

```
/* קלט: התכנית קולטת שלשות של נתונים (מספר כיתה, מספר בקבוקי פלסטיק, מספר סוללות)  */
/*        עבור כל אחד מ 68  התלמידים בשתי כיתות                                    */
/* פלט: התכנית מציגה כפלט את מספר הכיתה שצברה הכי הרבה נקודות                      */

public static void Main(string[] args)
{
    int numClass, bottle, battery, i;
    int points [2];

    points [1] = 0;
    points [2] = 0;
    for (i = 0; i < 68; i++)
    {
       Console.WriteLine ("enter your class, number of bottles, number of batteries:");
       numClass = int.Parse(Console.ReadLine());
       bottle = int.Parse(Console.ReadLine());          /* קלט שלשת הנתונים הנדרשת */
       battery = int.Parse(Console.ReadLine());
       points [numClass] = points [numClass] + bottle*3 + battery*7;
    }
    if ( points[1] > points[2] )
         Console.WriteLine ("class 1 won");
    else
         if ( points [2] > points [1] )
              Console.WriteLine ("class 2 won");
         else
              Console.WriteLine ("TEKO");
}
```

הסבר הפתרון: במקום שני מונים נפרדים אנו משתמשים במערך מונים. מספר הכיתה הוא המציין במערך, כך שהמערך במציין 1 הוא צובר הנקודות של כיתה 1 והמערך במציין 2 הוא צובר הנקודות של כיתה 2. ההבדל המשמעותי הוא בקידום הצובר. אין צורך לשאול על מספר הכיתה, אלא יש פניה ישירה על פי מספר הכיתה לצובר המתאים לכיתה במערך. מאחר ומדובר בשתי כיתות בלבד אין כאן יתרון משמעותי, אך אם מספר הכיתות היה גדול, יש להעדיף פתרון כזה.

פתרון שאלה 8

א. טבלת מעקב אחר ביצוע הפונקציה diff עם m=3 והמערך a

	m	i	sum	avg1	avg2	diff
ערכים התחלתיים	3		0			
לולאה ראשונה		0	1			
		1	4			
		2	6			
				(6/3 =) 2.0		
			0			
לולאה שנייה		2	2			
		3	6			
		4	12			
		5	12			
		6	16			
					(16/4 =) 4.0	(2.0–4.0 =) -2.0

הפונקציה תחזיר את הערך 2-.

ב. ההוראה שיש לשנות היא משפט ה for של הלולאה השנייה.

ג. המשפט צריך להיות: for (i = m; i < 7; i ++)

השינוי הוא הכרחי מאחר ואם הלולאה השנייה מתחילה מ- (m-1) אזי האיבר שמספרו הסידורי (m-1) נצבר בלולאה הראשונה וגם בשניה. הממוצע השני אמור להיות של "שאר אברי המערך" כלומר מהאיבר אחרי האיבר (m-1) במערך, כלומר מהאיבר ה- m ואילך.

פתרון שאלה 9

א. הפונקציה compute

טבלת משתנים

שם המשתנה	טיפוס המשתנה	הסבר/תפקיד	תחום ערכים
a	מערך	פרמטר, מערך שלמים לבדיקה	שלמים
num	שלם	פרמטר, המספר אליו יש להתייחס	שלמים
i	שלם	מציין במערך	0..19
counter	שלם	מונה מספר הערכים במערך הקטנים מ num, עד הופעתו או עד סוף המערך	1..20
found	שלם	האם num נמצא במערך או לא	1 , 0

int compute (arr a, int num)

```
/* טענת כניסה: הפונקציה מקבלת מערך a ומספר שלם num */
/* טענת יציאה: הפונקציה מחזירה את מספר הערכים הקטנים מ- num המופיעים לפני הופעתו */
/* הראשונה במערך. אם אינו מופיע במערך אזי תחזיר את מספר הערכים הקטנים ממנו בכל המערך */
{
    int i, counter;
    int found;

    i = 0;                          /* אתחול מציין המערך */
    found = 0;                      /* אתחול הדגל - num א נמצא עדיין */
    counter = 0;                    /* אתחול מונה מספר המספרים הקטנים מ num */
    while ((i < 20)&&(found == 0))  /* כל עוד num לא נמצא ועדיין לא נבדקו כל ערכי המערך */
    {
        if (a[i] == num)
            found = 1;              /* אם num נמצא אזי סמן שהוא נמצא */
        else
            if (a[i] < num)         /* אם num לא נמצא והאיבר התורן קטן ממנו */
                counter ++;         /* אזי קדם את המונה */
        i ++;                       /* התקדם לאיבר הבא במערך */
    }
    return counter;                 /* החזר את ערך המונה */
}
```

ב. התכנית המבוקשת

טבלת משתנים

שם המשתנה	טיפוס המשתנה	הסבר/תפקיד	תחום ערכים
arr	מערך חד-מימדי	מערך קלט	שלמים
i	מונה הלולאה	סורק את כל אברי המערך	0..19

```
// q9t02.c
                /*          קלט: התכנית קולטת ערכים למערך בגודל 20 , וקולטת 10 נתונים נוספים.  */
                /* פלט: עבור כל נתון מבצעת חישוב בהתאם לנתוני המערך ומציגה את התוצאה כפלט */
typedef int arr [20];
void readArray (arr a)                    /* הפונקציה קולטת ערכים למערך a */
{
      int i;
      for ( i = 0; i < 20; i ++ )
      {
            Console.WriteLine ("enter value for place: "+i);
            a [i] = int.Parse(Console.ReadLine());
      }
}

int compute (arr a, int num)

 /* הפונקציה מסעיף א */

public static void Main(string[] args)
{
      int i, number;
      arr m;

      readArray (m);                      /* קלט ערכים למערך m */
      for (i = 0; i < 10; i ++)           /* ביצוע של העיבוד 10 פעמים */
      {
            Console.Write ("enter number: ");
            number = int.Parse(Console.ReadLine()); /* קלט של ערך תורן */
            Console.WriteLine ("the computed value is: "+compute (m, number));
      }                                   /* הדפסת הערך המחושב על ידי הפונקציה */
}
```

ג1. אם תת-התכנית החזירה 0 יכולים להתקיים אחד משני המקרים הבאים:

(*) num לא נמצא כלל במערך וגם כל הערכים במערך גדולים ממנו.

(*) num נמצא במערך, אך עד אליו כל הערכים במערך גדולים ממנו.

ג2. אם תת-התכנית החזירה 20 אזי num בודאי לא נמצא במערך, מאחר ועד אליו לא יכולים להיות 20 ערכים, וגם כל הערכים במערך קטנים ממנו.

ד. בתחילת גוף הפונקציה ניתן להוסיף תנאי: אם המספר גדול מן הערך האחרון במערך אזי החזר 20, אחרת בצע את כל גוף הפונקציה כפי שהוא כתוב.

```
{
   if  (num > a [19] )
         return  20;
   else
    {
      /* כל גוף הפונקציה הקודם */
     }
}
```

פתרון שאלה 10

מספר פתרונות

א. גרסת פתרון 1:

טבלת משתנים לפונקציה

שם המשתנה	טיפוס המשתנה	הסבר/תפקיד	תחום ערכים
mis	שלם	פרמטר, ערך לבדיקה	שלם חיובי
n1	שלם	מועמד להיות המחלק הקטן	שלם חיובי
n2	שלם	מועמד להיות המחלק הגדול	שלם חיובי
p	שלם	השורש השלם של mis	שלם חיובי

הפונקציה המבוקשת **int find_dividersA (int mis)**

```
/*  טענת כניסה: הפונקציה מקבלת מספר שלם mis
/*  טענת יציאה: אם יש זוג מספרים שלמים שמכפלתם mis והפרשם 3 , תחזיר את הקטן, אחרת תחזיר 0 */
{
    int   p, n1, n2;
    p = (int) Math.Sqrt (mis);
    n1 = p;                 /* אתחול ערך המועמד להיות המחלק הקטן */
    n2 = p + 3;             /*   אתחול  ערך המועמד להיות המחלק השני כך שהפרשם 3   */
                            /* כל עוד לא התקבלה המכפלה mis ועוד יש סיכוי למצוא מחלקים */
    while ((n1 * n2 != mis) && (n2 >= p))
    {
        n1 -- ;             /* יצירת המספר הבא שמועמד להיות המחלק הקטן */
        n2 += 3;            /*  יצירת המחלק השני כך שהפרשם 3  */
    }
    if (n1 * n2 == mis)     /* אם הלולאה הסתיימה כי נמצאו המחלקים הנדרשים של mis */
        return n1;          /* אזי החזר את המחלק הקטן */
    else
        return 0;           /* אחרת החזר 0 */
}
```

הפתרון מבוסס על חיפוש זוג מתאים. מיותר לייצר את כל זוגות המכפלה האפשריים מאחר ואנו מחפשים רק כאלו שההפרש ביניהם 3. לכן למעשה החיפוש מתבצע על פי המועמד להיות המחלק הקטן, והמחלק השני תמיד גדול ממנו ב- 3. גבול מציאת המחלק הוא שורש של mis, מאחר ולא ייתכן שיהיו שני מספרים שמכפלתם mis ושניהם גדולים מן השורש שלו. אין צורך להתקדם עד mis עצמו וגם לא עד חציו.

גרסת פתרון 2: **int find_dividersB (int mis)**

```
/*  טענת כניסה: הפונקציה מקבלת מספר שלם mis
/*  טענת יציאה: אם יש זוג מספרים שלמים שמכפלתם mis והפרשם 3 , תחזיר את הקטן, אחרת תחזיר 0 */
{
    int n1, n2;
    n1 = 1;
    n2 = n1 + 3;
    while ((n1* n2 != mis) && (n1< Math.Sqrt (mis) ))
    {
        n1 = n1 + 1;
        n2 = n1 + 3;
    }
    if (n1*n2 == mis)
        return n1;
    else
        return 0;
}
```

הסבר הפתרון: גרסה זו מתבססת על עקרונות מתמטיים. אם מכפלתם של שני מספרים היא mis אזי אחד מהם חייב להיות קטן מן השורש שלו והשני גדול מן השורש שלו. במקרה זה החיפוש מתחיל "מן האמצע" כלומר מן הערך של שורש mis. המחלק הראשון יקטן בכל פעם באחד, כל עוד המחלק השני גדול מן השורש של p. מבחינת יעילות בהתייחס למשימה המסויימת הזו, פתרון זה יעיל יותר באופן משמעותי. לולאה כזו במקרה הגרוע ביותר תתבצע שלוש פעמים.

ב. פיתוח אלגוריתם

(i) טבלת משתנים של התכנית הראשית

שם המשתנה	טיפוס המשתנה	הסבר/תפקיד	תחום ערכים
all	מערך שלמים	ערך התא במציין i מעיד האם הערך i התקבל ע"י הפונקציה עבור קלט כלשהו	1 , 0
number	שלם	קלט תורן	שלם
n	שלם	ערך הפונקציה המוחזר עבור number	שלם
i	שלם	מציין לולאת הקלט	0..82

(ii) תת-משימות

שם תת-המשימה	סוג הפעולה	מטרת המשימה
find_dividers	פונקציה	הפונקציה מקבלת מספר שלם mis , אם קיימים שני מספרים שהפרשם 3 ומכפלתם mis תחזיר את המספר הקטן בזוג, אחרת תחזיר 0.
all_1_to_10	פונקציה	מקבלת מערך סימון בגודל 10 המציין בהתאמה עבור כל תא האם ערך המציין שלו התקבל על ידי הפונקציה find_dividers, ומחזירה 1 אם כל הערכים מ-1 עד 10 התקבלו ו 0 אחרת.
תכנית ראשית	תכנית ראשית	קולטת 83 מספרים ועבור כל מספר מבצעת: מפעילה את הפונקציה find_dividers מעדכנת את מערך הסימון בהתאם לערך שהוחזר מציגה כפלט את מסקנת הפונקציה all_1_to_10

ג. התכנית המבוקשת

```
// q10t02.c

typedef int arr [10];

int find_dividers (int mis)
{
      int p, n1, n2;

      p = (int) Math.Sqrt (mis);
      n1 = p;
      n2 = p + 3;
      while ((n1 * n2 != mis) && (n2 >= p))
      {
            n1 -= 1;
            n2 += 3;
      }
      if (n1 * n2 == mis)
            return n1;
      else
            return 0;
}
```

```
int all_1_to_10A (arr ten)
{
      int i;
      i = 0;
      while ((i < 10) && (ten [i]))        /* כל עוד לא נסרק כל המערך וגם ערכו של התא הנוכחי */
            i = i + 1;                     /* ten [i] הוא אמת, התקדם לתא הבא */
      if (i < 10)
            return 0;                      /* אם הלולאה הסתיימה כאשר i בתחום המערך, אזי נמצא */
      else                                 /* תא שערכו 'שקר', ולכן יוחזר 'שקר' */
            return 1;
}

public static void Main(string[] args)
{
   int number, n, i;
   arr      all;

   for (i = 0; i < 83; i++)                /* בצע עבור 83 מספרים */
   {
       Console.Write ("enter number: ");
       number = int.Parse(Console.ReadLine());          /* קלוט נתון תורן */
       n = find_dividersA (number);        /* חשב את ערך המחלק עבור הנתון התורן */
       if ( (n > 0) && (n < 11) )          /* אם המחלק הוא בטווח הערכים הנבדק */
            all [n - 1] = 1;               /* עדכן במערך הסימון שהתקבל ערך המחלק */
   }
   if ( all_1_to_10A (all) )               /* פלט האלגוריתם – האם כל הערכים מ-1 עד 10 התקבלו */
      Console.WriteLine ("YES");
   else
      Console.WriteLine ("NO");
}
```

```
int all_1_to_10B (arr ten)
      /*                                   טענת כניסה: הפונקציה מקבלת מערך בוליאני ten */
   /* טענת יציאה: הפונקציה מחזירה 'אמת' אם כל הערכים במערך הם 'אמת' ומחזירה 'שקר' אחרת */
{
      int i;
      int ok;
      ok = 1;
      for (i = 0; i < 10; i++)
            ok = ok && ten [i];
      return ok;
}
```

גרסת פתרון 2 עבור הפונקציה all_1_to_10:

הסבר הפתרון: גירסה זו משתמשת בפעולות הלוגיות בתוך חישוב של ביטוי לוגי. כל עוד ערך התא התורן הוא 1 , אזי משפט ההשמה: ok = ok && ten [i]; ישמור על ערך1 ב ok. כי ערכו של הביטוי (1 וגם 1) הוא 1 .
כאשר הסריקה תגיע לתא הראשון שערכו0 יהפוך ok להיות 0 ולא יוכל לשוב ולהשתנות עד סוף הסריקה. זאת מאחר וערכו של הביטוי (0 וגם 1) הוא 0, וגם ערכו של הביטוי (0 וגם 0) הוא 0.