הבחינה תורגמה לשפות java ו- C#

מדעי המחשב

2 יחידות לימוד

פרק ראשון - 50 נקודות

שאלה 1:

לפניך אלגוריתם:

```
(1)   עבור i מ- 1 עד 5 (כולל) בצע:
(1.1)   קלוט מספר תלת-ספרתי למשתנה num.
(1.2)   חשב את שארית החלוקה של num ב- 10 והצב ב- x
(1.3)   חשב את מנת החלוקה של num ב- 10 והצב ב- y
(1.4)   אם x*x שווה ל- y אז
(1.4.1)   הדפס את y
```

א. עקוב בעזרת טבלת מעקב אחר האלגוריתם עבור הקלטים הבאים (משמאל לימין):

749 , 111 , 164 , 213 , 819

ב. תן דוגמא למספר תלת ספרתי נוסף המקיים את התנאי שבהוראה 1.4

א.

i	i <=5	num	x	y	x*x = = y	פלט
1	כן	749	9	74	false	
2	כן	111	1	11	false	
3	כן	164	4	16	true	16
4	כן	213	3	21	false	
5	כן	819	9	81	true	81
6	לא					

ב. כל אחד ממספרים אלה הינו תשובה נכונה לסעיף ב': 255 , 366 , 497 , 648

שאלה 2:

מערך "סודר" הוא מערך חד-ממדי שמקיים את התנאי הבא: ערך כל תא, החל מהתא השני, שווה לסכום של מציין התא (האינדקס, i) וערך התא הקודם.

א. לפניך מערך a בגודל 6:

	0	1	2	3	4	5
a	2	3	5	8	10	15

האם מערך a הינו מערך "סודר"? אם לא, מהו המציין (האינדקס) של התא הראשון שאינו מקיים את התנאי?

נתונה כותרת הפעולה הבאה:

public static int isOrdinal (**int** [] a)

לפניך טענת הכניסה וטענת היציאה של הפעולה:

ט. כניסה: הפעולה מקבלת a, המכיל מספרים שלמים וחיוביים.
ט. יציאה: הפעולה מחזירה (-1) אם המערך הוא מערך "סודר", אם לא - תחזיר הפעולה את מציין (אינדקס) התא הראשון, שאינו מקיים את התנאי.

ב. השלם את גוף הפעולה, בהתאם לכותרת הפונקציה.

א. לא. התא הראשון שאינו מקיים מערך **סודר** הוא תא מספר 4 $(8 + 4 \neq 10)$.

ב. יש יותר מדרך אחת נכונה לפתור את הבעיה.

אפשר לפתור באחת משתי הדרכים המוצגות להלן, ואפשר גם לשלב מרכיבים משתי הפעולות, וליצור דרכים נוספות לפתרון.

```
public static int isOrdinal (int [] a)
{
    int i = 1, p = -1;
    while (i < 6  &&  p = = -1)
    {
        if (a[i] != i + a[i-1])
            p = i ;
        i ++ ;
    }
    return  p ;
}
```

```
public static int isOrdinal (int [] a)
{
    int i = 1;
    while (i < a.length )
    {
        if (a[i] -  a[i-1]  != i)
            return i ;
        i ++ ;
    }
    return  -1 ;
}
```

שאלה 3

כתוב תכנית מחשב המייצרת 42 זוגות של מספרים אקראיים. הראשון בתחום 25 עד 50 (כולל) והשני בתחום 1 עד 10 (כולל). עבור כל זוג מספרים, יש לבדוק האם המספר הראשון מתחלק בשני ללא שארית. כפלט יש להדפיס כמה מהזוגות קיימו את תנאי ההתחלקות.

: Java

```
//--- תכנית המייצרת זוגות של מספרים אקראיים וסופרת בכמה מהם מתחלק הראשון בשני ---
import java.util.Random;
public static void main (String [] args)
{
        int a, b, counter = 0 ;
        Random rnd = new Random() ;
        for (int i = 1 ; i <= 42 ; i++)
        {
                a = rnd.nextInt (26) + 25 ;     //---     a = (int)(Math.random()*26) + 25;
                b = rnd.nextInt (10) + 1 ;      //---     b = (int)(Math.random()*10) + 1;
                if (a % b = = 0)
                        counter ++ ;
        }
        System.out.println (" זוגות מתחלקים " + counter + " יש ") ;
}
```

: C#

```
//--- תכנית המייצרת זוגות של מספרים אקראיים וסופרת בכמה מהם מתחלק הראשון בשני ---
public static void Main (string [] args)
{
        int a, b, counter = 0 ;
        Random rnd = new Random() ;
        for (int i = 1 ; i <= 42 ; i++)
        {
                a = rnd.Next (26) + 25 ;        //---     a = rnd.Next (25, 51)
                b = rnd.Next (10) + 1 ;         //---     b = rnd.Next (1, 11)
                if (a % b = = 0)
                        counter ++ ;
        }
        Console.WriteLine (" זוגות מתחלקים " + counter + " יש ") ;
}
```

שאלה 4

נתונה כותרת הפעולה midMax :

Java: **static boolean** midMax (**int** a, **int** b, **int** c)
C#: **static bool** MidMax (**int** a, **int** b, **int** c)

טענת כניסה: הפעולה מקבלת כפרמטרים 3 מספרים שלמים a, b ו- c.
טענת יציאה: מוחזר "אמת" אם המספר השני b גדול משני שכניו, ו-"שקר" אחרת

א. כתוב משפט זימון לפעולה midMax/MidMax עבורו יוחזר ערך "אמת".
ב. כתוב משפט זימון לפעולה midMax/MidMax עבורו יוחזר ערך "שקר".
ב. השלם את גוף הפעולה, בהתאם לטענת היציאה.

א. אמת: C#: **bool** x = *MidMax* (4, 6, 2) ; Java: **boolean** x = *midMax* (4, 6, 2) ;
א. שקר: C#: **bool** x = *MidMax* (1, 2, 3) ; Java: **boolean** x = *midMax* (1, 2, 3) ;

ג.

Java:

```
static boolean midMax (int a, int b, int c)
 {
        if (b > a   &&  b > c)
                return  true;
        return  false;
 }
```

C#:

```
static bool MidMax (int a, int b, int c)
{
        if (b > a   &&  b > c)
                return  true;
        return  false;
}
```

שאלה 5

כתוב תכנית שתבצע את הפעולות הבאות:

א. תקלוט סדרה של מחרוזות עד שתיקלט המחרוזת "xxx" ותדפיס את המחרוזת הארוכה ביותר.

ב. התכנית תבדוק ותדפיס כמה פעמים מופיעה האות הראשונה במחרוזת שנמצאה בסעיף א' (הארוכה ביותר).
דוגמא: אם המחרוזת הארוכה היא: ABAACBACD, יהיה הפלט 4
(כי האות A הופיעה במחרוזת 4 פעמים).

הערה: הנח כי נקלטה לפחות מחרוזת אחת.

Java:

```
import java.util.Scanner;

public static void main (String [] args)
{
        Scanner input = new Scanner (System.in);

        String str, maxStr ;
        int strLength, maxLength = 0, count = 0 ;

        System.out.print ("Type a string ("xxx" to stop) → ");
        str = input.next();                          //---     str = input.nextLine();

        while ( str.compareTo ("xxx") != 0)          //--- ( ! str.equals("xxx") )
        {
                strLength = str.length();
                if (strLength > maxLength)
                {
                        maxStr = str ;
                        maxLength = strLength;
                }
                System.out.print ("Type a string ("xxx" to stop) → ");
                str = input.next();
        }
        System.out.println (maxStr);

        //--- סעיף ב' - מספר המופעים של התו הראשון ---//
        for (int  i = 0 ; i < maxLength ; i++)
                if (maxStr.charAt(i) = = maxStr.charAt(0))
                        count ++ ;
        System.out.println (count );

}
```

אפשרות נוספת:

```
if (str.length() > maxStr.length() )
{
        maxStr = str ;
}
```

C#:

```
public static void Main (string [] args)
{
        string str, maxStr ;
        int strLength, maxLength = 0, count = 0 ;

        Console.Write ("Type a string ("xxx" to stop) → ");
        str = Console.ReadLine();

        while ( str.CompareTo ("xxx") != 0)                    //--- ( ! str.Equals("xxx") )
        {
                strLength = str.Length();
                if (strLength > maxLength)
                 {
                        maxStr = str ;
                        maxLength = strLength;
                 }
                Console.Write ("Type a string ("xxx" to stop) → ");
                str = Console.ReadLine();
         }
        Console.WriteLine (maxStr);


        //--- סעיף ב' - מספר המופעים של התו הראשון ---
        for (int  i = 0 ; i < maxLength ; i++)
                if (maxStr.CharAt(i) = = maxStr.CharAt(0))
                        count ++ ;
        Console.WriteLine (maxStr + " מופיע ב- " +  maxStr.CharAt(0) + " התו ");
        Console.WriteLine (" פעמים " +  count );

 }
```

אפשרות נוספת:

```
if (str.Length > maxStr.Length )
{
        maxStr = str ;
}
```

פרק שני - 30 נקודות

שאלה 6:

אתר אינטרנט מסוים דורש שסיסמת המשתמש תהיה בת לפחות 6 תווים (אפשר גם יותר) ותכיל לפחות אות אחת מה- "ABC..." (אותיות גדולות) ולפחות שתי ספרות.

א. כתוב פעולה בשם existUpper המקבלת מחרוזת ומחזירה "אמת" אם קיים בה לפחות תו אחד שהוא אות רישית (אות מה- ABC), ו"שקר" אחרת.

ב. כתוב פעולה בשם countDigits המקבלת מחרוזת ומחזירה את מספר הספרות במחרוזת.

ג. כתוב קטע תכנית הקולט סיסמא ומדפיסה "סיסמא חוקית" אם הסיסמא עומדת בתנאים, ו-"סיסמא לא חוקית" אחרת.

Java:

```
//                                                  סעיף א': - טענת כניסה: מחרוזת
//      טענת יציאה: מוחזר "אמת" אם קיימת אות רישית אחת לפחות במחרוזת, ו-"שקר" אחרת
public static boolean existUpper (String pw)
{
        int i = 0;
        while (i < pw.length()
        {
                if (pw.charAt(i) >= 'A'  &&  pw.charAt(i) <= 'Z')
                        return true;
                i++;
        }
        return false;
}
//                                                  סעיף ב': - טענת כניסה: מחרוזת
//      טענת יציאה: מוחזר מספר הספרות במחרוזת (שים ♥: סיפרה ≠ מספר !!!)
public static int countDigits (String pw)
{
        int count = 0;
        for (int i = 0 ; i < pw.length() ; i++)
                if (pw.charAt(i)  >= '0'  &&  pw.charAt(i)  <= '9')
                        count ++ ;
        return count ;
}
                                                    סעיף ג' - בדיקת הסיסמה
System.out.print ("Type your password → ");
String pw = input.next ();
if (pw.length() >= 6  &&  existUpper|(pw)  &&  countDigits (pw) > 2)
        System.out.println ("סיסמא חוקית");
else    System.out.println ("סיסמא לא חוקית");
```

C#:

```
//                                                    סעיף א': - טענת כניסה: מחרוזת
//      טענת יציאה: מוחזר "אמת" אם קיימת אות רישית אחת לפחות במחרוזת, ו-"שקר" אחרת
public static boolean ExistUpper (string pw)
{
        int i = 0;
        while (i < pw.Length )
        {
                if (pw[i] >= 'A'  &&  pw[i] <= 'Z')
                        return true;
                i++;
        }
        return false;
}
//                                                    סעיף ב': - טענת כניסה: מחרוזת
//      טענת יציאה: מוחזר מספר הספרות במחרוזת (שים ♥: סיפרה ≠ מספר !!!)
public static int CountDigits (string pw)
{
        int count = 0;
        for (int i = 0 ; i < pw.Length ; i++)
                if (pw[i] >= '0'  &&  pw[i] <= '9')
                        count ++ ;
        return count ;
}
                                                      סעיף ג' - בדיקת הסיסמה
Console.Write ("Type your password → ");
string pw = Console.ReadLine ();

if (pw.Length >= 6  &&  ExistUpper|(pw)  &&  CountDigits (pw) > 2)
        Console.WriteLine ("סיסמא חוקית");
else
        Console.WriteLine ("סיסמא לא חוקית");
```

שאלה ד:

"**ספרה ראשונה**" של מספר היא ספרת האחדות שלו.
"**מספר מהיר**" הוא מספר בו כל ספרה, פרט לראשונה, גדולה מקודמתה.
למשל המספרים 541, 932, 853 הינם מספרים מהירים.

א. כתוב פעולה המקבלת כפרמטר מספר תלת סיפרתי ומחזירה "אמת" אם המספר הוא **מספר מהיר** ו"שקר" אחרת.

ב. כתוב תכנית הקולטת מספרים תלת ספרתיים ומדפיסה עבור כל מספר האם הוא **מספר מהיר**. הנחייה: יש לשלב את הפעולה שכתבת בסעיף א' בתוך התכנית ולהשתמש בה לצורך בדיקת המספרים. התכנית תסתיים כאשר ייקלט המספר 999.

:Java

```
import java.util.Scanner;
//--- correct password ---
class FastNumber
{
        static Scanner input = new Scanner (System.in);
        /*      return TRUE if num is a fast-num, otherwise return FALSE.        */
        static boolean isFastNum (int num)
        {
                int d1, d2, d3;
                d1 = num % 10 ;                 // ספרת האחדות
                num = num / 10 ;
                d2 = num % 10 ;                 //  ספרת העשרות
                d3 = num / 10 ;                 //  ספרת המאות
                if (d1 < d2 && d2 < d3)
                        return true ;
                return false ;
        }
        public static void main (String [] args)
        {
                int  num = getNum () ;
                while (num != 999)
                {
                        if (isFastNum (num))
                                System.out.println (num + " is a fast-num ");
                        else
                                System.out.println (num + " is not a fast-num ");
                        num = getNum() ;
                }
        }
```

```
        //--- returns a 3 digit number ---
        static int getNum ()
         {
                int num  ;
                do {
                        System.out.print ("Enter a 3 digit number → ") ;
                        num = input.nextInt() ;
                }  while (num < 100 || num > 999) ;
                return num;
         }
} //--- end of class FastNumber
```

:C#

```
//--- correct password ---
class FastNumber
{
        /*      return TRUE if num is a fast-num, otherwise return FALSE.        */
        static bool IsFastNum (int num)
        {
                int d1, d2, d3;

                d1 = num % 10 ;                 // ספרת האחדות
                num = num / 10 ;
                d2 = num % 10 ;                 //  ספרת העשרות
                d3 = num / 10 ;                 //  ספרת המאות

                if (d1 < d2 && d2 < d3)
                        return true ;
                return false ;
         }
        public static void Main (string [] args)
        {
                int  num = GetNum () ;
                while (num != 999)
                {
                        if (IsFastNum (num))
                                Console.WriteLine (num + " is a fast-num ");
                        else
                                Console.WriteLine (num + " is not a fast-num ");
                        num = GetNum() ;
                }
         }
```

```
        //--- returns a 3 digit number ---
        static int GetNum ()
         {
                int num  ;
                do {
                        Console.Write ("Enter a 3 digit number → ") ;
                        num = int.Parse(Console.ReadLine()) ;
                }  while (num < 100 || num > 999) ;
                return num;
         }
} //--- end of class FastNumber
```

שאלה 8:

נתונה מטריצה mat (מערך דו-ממדי) מגודל 10x10 המכילה מספרים שלמים וחיוביים. (הנח שנתוני המטריצה תקינים).
כתוב תכנית שתמצא בכל עמודה במטריצה את הערך המינימלי ותחליף בינו לבין איבר האלכסון הראשי של אותה עמודה. יש להדפיס את איברי המטריצה בסיום ההחלפה.

דוגמא: עבור מטריצה מגודל 4x4:

לפני ההחלפה

	0	1	2	3
0	5	4	6	2
1	8	1	5	4
2	2	3	8	6
3	9	6	1	8

אחרי ההחלפה

	0	1	2	3
0	2	4	6	8
1	8	1	5	4
2	5	3	1	6
3	9	6	8	2

Java:

```
//--- החלפת איבר מינימלי של כל עמודה באיבר האלכסון הראשי במטריצה ---
class MinMatrix
{
    public static void main (String [] args)
    {
        int [][] mat = matKelet () ;    //--- פעולה הממלאה ומחזירה
                                        //--- את המטריצה הנתונה
        int row;
        int lastCol = mat [0].length;       //--- int lastCol = 10;
        for (int i = 0 ; i < lastCol ; i++)
         {
            row = minPlaceInCol (mat, i) ;
            swap (mat, row, i) ;
         }
        matPelet (mat);
    }
    //--- פעולה הקולטת איברים למטריצה ---
    static void matKelet (int [][] mat)
    {
        //--- כתוב בשאלה כי המטריצה נתונה לכן אין חובה לממש פעולה זו ---
        //--- מספיק להכריז על קיומה. ---
    }
```

```
//--- פעולה המחזירה את מקומו של האיבר המינימלי בעמודה col ---
static int  minPlaceInCol (int [][] mat, int col)
{
        int lastRow = mat.length;               //--- int lastRow = 10;

        int min = mat[0][col] ;
        int minRow = 0 ;

        for (int row = 1 ; row < lastRow ; row ++)
                if ( mat [row][col] < min )
                 {
                        min = mat [row][col];
                        minRow = row ;
                 }
        return minRow;
}

//---   פעולה המחליפה תוכן התאים בשורה y עמודה y (איבר האלכסון) ---
//---                  ובשורה x עמודה y (מינימלי בעמודה), זה בזה ---

static void swap (int [][]mat, int x, int y)
 {
        int  temp = mat[x][y];
        mat[x][y] = mat [y][y];
        mat[y][y] = temp ;
 }

//--- פעולה המדפיסה את איברי המטריצה ---
static void matPelet (int [][] mat)
 {
        int N = mat.length;             //--- int N = 10;

        for (int i = 0 ; i < N ; i++)
         {
                for (int j = 0 ; j < N ; j++)
                        System.out.print (mat[i][j] + "  ") ;    //--- הדפסה במרווח קבוע ---
                System.out.println () ;
         }
 }

} //---  end of class MinMatrix
```

:C# //--- החלפת איבר מינימלי של כל עמודה באיבר האלכסון הראשי במטריצה ---

```
class MinMatrix
{
        public static void Main (string [] args)
        {
                int [,] mat = MatKelet () ;      //---  פעולה הממלאה ומחזירה
                                                 //--- את המטריצה הנתונה
                int row;
                int lastCol = mat.GetLength(1);          //--- int lastCol = 10;
                for (int i = 0 ; i < lastCol ; i++)
                 {
                        row = MinPlaceInCol (mat, i) ;
                        Swap (mat, row, i) ;
                 }
                MatPelet (mat);
        }
        //--- פעולה הקולטת איברים למטריצה ---
        static void MatKelet (int [,] mat)
        {
                //--- כתוב בשאלה כי המטריצה נתונה לכן אין חובה לממש פעולה זו ---
                //---                                  מספיק להכריז על קיומה. ---
        }
        //--- פעולה המחזירה את מקומו של האיבר המינימלי בעמודה col ---
        static int  MinPlaceInCol (int [,] mat, int col)
        {
                int lastRow = mat.GetLength(0);          //--- int lastRow = 10;
                int min = mat[0,col] ;
                int minRow = 0 ;
                for (int row = 1 ; row < lastRow ; row ++)
                        if ( mat [row,col] < min )
                         {
                                min = mat [row,col];
                                minRow = row ;
                         }
                return minRow;
        }
```

```
//---  פעולה המחליפה תוכן התאים בשורה y עמודה y (איבר האלכסון) ---
//---  ובשורה x עמודה y (מינימלי בעמודה), זה בזה ---

static void Swap (int [,]mat, int x, int y)
{
        int  temp = mat[x,y];
        mat[x,y] = mat [y,y];
        mat[y,y] = temp ;
}

//--- פעולה המדפיסה את איברי המטריצה ---
static void MatPelet (int [,] mat)
{
        int N = mat.GetLength(0);          //--- int N = 10;

        for (int i = 0 ; i < N ; i++)
        {
                for (int j = 0 ; j < N ; j++)
                        Console.Write (mat[i,j] + "  ") ;
                Console.WriteLine () ;
        }
}
} //---  end of class MinMatrix
```

פרק שלישי - 20 נקודות

שאלה 9:

עידו המדריך החליט לארגן לחניכיו פעולה בסגנון "חפש את המטמון". בשטח בית-הספר שבו 72 חדרים הממוספרים מ- 0 עד 71 הוחבאו מטמונים. בכל חדר נמצא פתק שבו רשום מספר החדר הבא שבו יש להמשיך את החיפוש, או המטמון עצמו. החניכים התארגנו ב- 8 חוליות.

כל חוליה מתחילה את החיפוש בחדר אחר. עידו הכין את הפתקים והמטמונים מבעוד מועד ורצה לפזר אותם בחדרים. לשם כך הוא הכין שני מערכים:

הראשון - מערך A בגודל 8 ובו ירשם החדר בו תתחיל כל חוליה את החיפוש.
בתא הראשון ירשם החדר בו תתחיל חוליה ראשונה את החיפוש, בתא השני ירשם החדר בו תתחיל חוליה שנייה את החיפוש וכך הלאה, עד התא האחרון.

השני – מערך B בגודל 72 תאים. בכל תא במערך רשום מספר בין (1-) ל- 71, המספר (1-) מציין שהמטמון נמצא בחדר זה, כל מספר אחר אומר לאיזה חדר יש לעבור להמשך החיפוש.

דוגמא: עבור 4 חוליות ו- 16 חדרים:

A	3	7	12	9

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

החוליה הראשונה מתחילה בחדר 3, מחפשת בחדרים 8 ו- 0 ומוצאת את המטמון בחדר 14.

שים ♥: מספר החדרים שצריכה לעבור כל חוליה עשוי להשתנות מחוליה לחוליה.

א. יש לכתוב פעולה שתקבל כפרמטרים את המערך B ומספר חדר k. ותחזיר את מיקומו של המטמון אליו מגיעים אם מתחילים לחפש בחדר k.

ב. לפני הפעלת המשחק החליט עידו לבדוק שכל חוליה מגיעה למטמון אחר, כתוב תכנית שתקלוט את מספרי החדרים ההתחלתיים של כל חוליה למערך A, ותבדוק האם קיים מצב שבו יש יותר מחוליה אחת המגיעה לאותו מטמון, אם כן יודפס: "כפילות בהגעה למטמון", אחרת יודפס: "המסלולים תקינים".
חובה להשתמש בפעולה שכתבת בסעיף א'.

הנחיות:

פרק את הבעיה לתת משימות כך שכל תת משימה תיפתר על ידי תת-תכנית. הגדר את המטרה של כל תת-משימה (הגדר טענת כניסה וטענת יציאה לכל תת-משימה).

בחר במשתנים עיקריים, הגדר את טיפוסיהם, ותאר את תפקידיהם.

תת-משימות:

- קלט מסלולים למערך החדרים. arr2Kelet (B)
- קלט תחילת מסלול לקבוצה למערך הקבוצות. arr1Kelet (A)
- שמירת חדר הסיום לכל קבוצה במערך המטמונים. לולאה בתכנית.
- בדיקת מסלולים מצטלבים. checkTreasureRoom (C)

משתנים עיקריים:

שם משתנה	טיפוס משתנה	תפקיד המשתנה
A	מערך שלמים בגודל 8	שומר את מספר החדר בו מתחילה כל קבוצה.
C	מערך שלמים בגודל 8	שומר את מספר החדר בו מסיימת כל קבוצה.
B	מערך שלמים בגודל 72	שומר את מסלולי הקבוצות.

Java:

```
import java.util.Scanner ;
//--- Looking for treasure  ---
class TreasureGame
{
        static Scanner input = new Scanner (System.in);

        public static void main (String [] args)
        {
                int [] B = new int [72];                //--- מערך החדרים ---
                int [] A = new int [10];                //--- מערך ההתחלה  ---
                int [] C = new int [10];                //--- מערך המטמונים ---

                arr1Kelet (A) ;          //--- מילוי מערך תחילת המסלול ---
                arr2Kelet (B) ;          //--- מילוי מערך החדרים ---

                for (int i = 0 ; i < A.length ; i++)          // --- מילוי מערך סיום המסלול  ---
                        C[i] = treasure (B, A[i]) ;

                if (checkTreasureRoom (C))
                        System.out.println ("Incorrect path ! ") ;
                else

                        System.out.println ("All paths are O.K.") ;
        }

        //---  פעולה המקבלת את מערך החדרים ומספר החדר בו מתחיל המסלול ---
        //---      ומחזירה את מספר החדר בו מסתיים המסלול (בו נמצא האוצר).  ---
        static int treasure (int [] b, int k)
        {
                while (b [k] != -1)
                        k = b [k] ;
                return k ;
        }
```

```
// --- מילוי מערך תחילת המסלול ---
static void arr1Kelet (int [] a)
 {
        for (int i = 0 ; i < a.length ; i++)
         {
                System.out.print (i + " באיזה חדר מתחילה קבוצה מספר→ ");
                a[i] = input.nextInt () ;
         }
 }
// --- מילוי מערך החדרים ---
static void arr2Kelet (int [] b)
 {
        for (int i = 0 ; i < b.length ; i++)
        {
                System.out.print ("לאיזה חדר להמשיך? (1- לאוצר)"+ i
                                + " אתה בחדר מספר→ ");
                b[i] = input.nextInt () ;
        }
}
//--- בדיקה: האם קיימות שתי קבוצות המסיימות באותו חדר ---
static int checkTreasureRoom (int [] c)
 {
        boolean equal = false ;
        for (int i = 0 ; i < c.length - 1 ; i++)
                for (int j = i+1 ; j < c.length ; j++)
                        if (c[i] = = c[j])   equal = true ;
        return equal ;
 }
} //--- end of class TreasureGame
```

:C#

```
//--- Looking for treasure  ---
class TreasureGame
{
        public static void Main (string [] args)
        {
                int [] B = new int [72];                    //--- מערך החדרים ---
                int [] A = new int [10];                    //--- מערך ההתחלה  ---
                int [] C = new int [10];                    //---  מערך המטמונים ---

                Arr1Kelet (A) ;                    //--- מילוי מערך תחילת המסלול ---
                Arr2Kelet (B) ;                    //--- מילוי מערך החדרים ---

                for (int i = 0 ; i < A.Length ; i++)        // --- מילוי מערך סיום המסלול  ---
                        C[i] = Treasure (B, A[i]) ;

                if (CheckTreasureRoom (C))
                        Consol.WriteLine ("Incorrect path ! ") ;
                else

                        Consol.WriteLine ("All paths are O.K.") ;
        }

        //---  פעולה המקבלת את מערך החדרים ומספר החדר בו מתחיל המסלול ---
        //---      ומחזירה את מספר החדר בו מסתיים המסלול (בו נמצא האוצר).  ---
        static int Treasure (int [] b, int k)
        {
                while (b [k] != -1)
                        k = b [k] ;
                return k ;
        }

        // --- מילוי מערך תחילת המסלול ---
        static void Arr1Kelet (int [] a)
         {
                for (int i = 0 ; i < a.Length ; i++)
                 {
                        Consol.Write (i + " באיזה חדר מתחילה קבוצה מספר→ ");
                        a[i] = int.Parse(Consol.ReadLine()) ;
                 }
         }
```

```
        // --- מילוי מערך החדרים ---
        static void Arr2Kelet (int [] b)
         {
                for (int i = 0 ; i < b.Length ; i++)
                {
                        Consol.Write ("לאיזה חדר להמשיך? (1- לאוצר) "+ i
                                        + " אתה בחדר מספר ←");
                        b[i] = int.Parse(Consol.ReadLine()) ;
                }
        }
        //--- בדיקה: האם קיימות שתי קבוצות המסיימות באותו חדר ---
        static int CheckTreasureRoom (int [] c)
         {
                boolean equal = false ;
                for (int i = 0 ; i < c.Length - 1 ; i++)
                        for (int j = i+1 ; j < c.Length ; j++)
                                if (c[i] = = c[j])   equal = true ;
                return equal ;
         }
} //--- end of class TreasureGame
```

שאלה 10:

על מנת למקם תחנת כיבוי-אש שתשרת את יישובי הסביבה, סומנו N (N = 15) היישובים במספרים 0 עד N-1. המרחקים בין היישובים נרשמו במערך דו-מימדי d בגודל N*N, כך שבאיבר שבשורה i עמודה j נרשם המרחק בין יישוב i ליישוב j.

את התחנה יש למקם ביישוב שמרחקו המקסימלי מכל היישובים האחרים באזור הוא הקטן ביותר (כלומר היישוב הקרוב ביותר לכל יישובי הסביבה).

א. פרק את הבעיה לתת משימות כך שכל תת משימה תיפתר על ידי תת-תכנית. הגדר את המטרה של כל תת-משימה (הגדר טענת כניסה וטענת יציאה לכל תת-משימה).

ב. בחר במשתנים עיקריים, הגדר את טיפוסיהם, ותאר את תפקידיהם.

ג. כתוב תכנית ליישום תת המשימות שהגדרת בסעיף א' כך שיודפס מספרו של הישוב המתאים ביותר למיקום התחנה.

דוגמא: עבור המערך הדו-מימדי המיצג ארבעה ישובים:

j \ i	0	1	2	3	**המרחק המקסימלי של היישוב j מכל היישובים:**
0	0	18	10	12	18
1	18	0	9	20	20
2	10	9	0	13	13
3	12	20	13	0	20

המרחק המינימלי

ולכן תמוקם התחנה בישוב מספר 2.

תת-בעיות:

- קלט למטריצה - פעולה הקולטת את המרחקים שבין היישובים השונים.
- מציאת המרחק המינימלי
  - עבור כל אחד מהישובים, מצא את המרחק המקסימלי של היישוב משאר היישובים.
  - מצא את המינימום מבין כל המרחקים שנמצאו.

משתנים:

שם משתנה	טיפוס המשתנה	תפקיד המשתנה
d	מטריצה בגודל N*N	טבלת המרחקים שבין היישובים.
maxDistance	מספרי שלם	המרחק המקסימלי של יישוב משאר היישובים.
min	מספרי שלם	המרחק המינימלי מבין המרחקים המקסימלים.
place	מספרי שלם	מספרו של היישוב בעל מרחק min.

:Java

```
//--- Best Location for Fire Station ---
import java.util.Scanner;
class FireStationLocation
{
    static Scanner input = new Scanner (System.in);
    static final int N = 15;
    public static void main (String [] args)
    {
        int [][] d = new int [N][N];
        int min, maxDistance , place = 0;

        matKelet (d) ;
        min = maxInLine (d, 0) ;            // המרחק המקסימלי של ישוב 0
        for (int i = 1 ; i < N ; I++)
        {
            maxDistance = maxInLine (d, i) ;
            if  (maxDistance < min)
            {
                min = maxDistance ;
                place = i ;
            }
        System.out.println ("Best Location is at " + place) ;
    }
    //--- function receives matrix and line number, returns max value in line ---
    static int maxInLine (int [][] d, int i)
    {
        int max = 0 ;
        for (int j = 0 ; j < N ;  j++)
            if (d[i][j] > max)
                max = d[i][j] ;
        return max ;
    }

    //---  input distance between towns and villages ---
    static void matKelet (int [][] d)
    {
        for (int i = 0 ; i < N ;  i++)
            for (int j = 0 ;  j < N ;  j++)
            {
                System.out.print ("Type distance between " + i + " and " + j + " →");
                d[i][j] = input.nextInt() ;
            }
    }
}  //--- end of class FireStationLocation
```

:C#

```
//--- Best Location for Fire Station ---
class FireStationLocation
{
    static const int N = 15;
    public static void Main (string [] args)
    {
        int [,] d = new int [N,N];
        int min, maxDistance , place = 0;

        MatKelet (d) ;
        min = MaxInLine (d, 0) ;            // המרחק המקסימלי של ישוב 0
        for (int i = 1 ; i < N ; I++)
        {
            maxDistance = MaxInLine (d, i) ;
            if  (maxDistance < min)
            {
                min = maxDistance ;
                place = i ;
            }
        Consol.WriteLine ("Best Location is at " + place) ;
    }
    //--- function receives matrix and line number, returns max value in line ---
    static int MaxInLine (int [,] d, int i)
    {
        int max = 0 ;
        for (int j = 0 ; j < N ;  j++)
            if (d[i,[j] > max)
                max = d[i,j] ;
        return max ;
    }

    //---  input distance between towns and villages ---
    static void MatKelet (int [,] d)
    {
        for (int i = 0 ; i < N ;  i++)
            for (int j = 0 ;  j < N ;  j++)
            {
                Consol.Write ("Type distance between " + i + " and " + j + " →");
                d[i,j] = int.Parse(Console.ReadLine()) ;
            }
    }
}  //--- end of class FireStationLocation
```