

תיקייה לבגרות

תוכן עניינים

3.....	בניית מספר הפוך (פולינדרום)
4.....	char
5.....	סדר אופרטורים
6.....	Math
6.....	Random
7.....	דוגמה לטבלת מעקב של While
8.....	מילון הגדרות
8.....	כלל מתי משתמשים לולאות ומאיזה סוג
9-14.....	string
15-17.....	פונקציות
19-20.....	מערכים
21.....	שלבי בדיקה כללית של מבחן

Palindrome

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp6
8 {
9     class Program
10    {
11        static void Main(string[] args)
12        {
13            Console.Write("Enter a num: ");
14            int num = int.Parse(Console.ReadLine());
15            int b_num = num;
16            int dig = 0;
17            int rev = 0;
18            while (num > 0)
19            {
20                dig = num % 10;
21                rev = rev * 10 + dig;
22                num = num / 10;
23            }
24            if (rev == b_num)
25                Console.WriteLine("This is palindrome");
26            else
27                Console.WriteLine("This is not palindrome");
28        }
29    }
30 }
```

פלינדרום הוא מספר שניתן לקרוא אותו משני הצדדים ויהיה לו את אותו הערך. משתמשים בשאלות על פלינדרומים הרבה במבחנים. לשים דגש על שאלות אלו.

דוגמה עם Char

ערך ה char הוא בעצם קוד ASCII, לכן אפשר לשנות אותו בעזרת מתמטיקה.

כדאי להשתמש ב +1 או -1, חשוב להמיר ממספרים ל-char.

```
char test = 'b';           'b' מקבל test
char x = (char)(test + 1); 'c' מקבל x
char y = (char)(test - 1); 'a' מקבל y
//char c = a + b;         טעות הידור (קומפילציה)
int zz = x + y;           'c' מקבל 196. זה סכום ערך ASCII של 'a' וערך ASCII של 'c'.
char z = (char)(x + y);   'z' מקבל 196 – זה char שערך ASCII שלו 196.
```

ערכו של zz יהיה שווה לסכום ערכי האסקי של chars.

כשעושים על chars פעולה מתמטית + או - על char – מקבלים int.

כשעושים על chars פעולה מתמטית ++ או -- על char – מקבלים char.
להיזהר מ ++ ו -- על char! עדיף לעשות +1, -1. כי:

```
char test = 'g';           'g' מקבל test
char a = test++;           'h' מקבל a, 'g' מקבל test. לא מה שציפינו.
char b = test--;           'g' מקבל test, 'h' מקבל b. לא מה שציפינו.
```

ניתן להמיר char ל int ולהפך:

```
char x = 'a';
int y = 15;
int xx = x;
xx = (int)x;
char yy = (char)y;
```

```
char test = 'g';
int test2 = Convert.ToInt32(test);
Console.WriteLine(test2);

int test3 = 58;
char test4 = Convert.ToChar(test3);
Console.WriteLine(test4);
```

סדר אופרטורים

בשפת C#:

- סימן של כשר AND (גם) הוא &&
- סימן של כשר OR (או) הוא ||
- סימן אופרטור NOT (לא) הוא !

חישוב ביטוי בוליאני מורכב שאינו מכיל סוגריים יעשה לפי סדר הקדימויות:

!	1	עדיפות גבוהה ביותר
!=, <, <=, ==, >, >=	2	
&&	3	
	4	עדיפות נמוכה ביותר

דוגמה במתמטיקה	דוגמה ב-C#	משמעות סימן השוואה	סימן השוואה המקובל במתמטיקה	סימן השוואה ב-C#
$x = 5$	<code>x == 5</code>	שווה	=	==
$x \neq y$	<code>x != y</code>	שונה	\neq	!=
$x < 2$	<code>x < 2</code>	קטן	<	<
$x \leq 2$	<code>x <= 1</code>	קטן או שווה	\leq	<=
$y > 0$	<code>y > 0</code>	גדול	>	>
$y \geq 8$	<code>y >= 8</code>	גדול או שווה	\geq	>=

פונקציות של Math

פעולות שימושיות מהמחלקה המתמטית Math

דוגמה		טיפוס ערך מוחזר	טיפוס פרמטרים	תיאור הפעולה	הפעולה
הערך המוחזר	הפעולה				
63	Math.Abs (63)	שלם	שלם	ערך מוחלט	Abs (num)
12.7	Math.Abs (-12.7)	ממשי	ממשי		
2.5	Math.Sqrt (6.25)	ממשי	ממשי	שורש ריבועי	Sqrt (num)
9.0	Math.Pow (3, 2)	ממשי	ממשי, ממשי	חזקה num1 num2	Pow (num1, num2)
3	Math.Min (3, 8)	שלם	שלם, שלם	הקטן מבין השניים	Min (num1, num2)
8.0	Math.Min (8.0, 8.8)	ממשי	ממשי, ממשי		
8	Math.Max (3, 8)	שלם	שלם, שלם	הגדול מבין השניים	Max (num1, num2)
8.8	Math.Max (8.0, 8.8)	ממשי	ממשי, ממשי		
8	Math.Round (7.9)	שלם	ממשי	עיגול מספר ממשי	Round (num)

Random

כדי לקרוא ל-Random, יש ליצור אחד:

```
Random rnd = new Random();
```

```
int test = rnd.Next(1, 101);
```

```
int a = rnd.Next(-5, 4);
```

```
int b = rnd.Next(4, -5);
```

בין 1 כולל עד 100 כולל

בין -5 כולל עד 3 כולל

נכשל בזמן ריצה:

System.ArgumentOutOfRangeException:
"minValue" cannot be greater than maxValue.

בדוגמה, ה-`random` נמצא בטווח של 1-100 ולא עד 101 מפני שב-`random`, המספר האחרון אף פעם לא נכלל.

אם רוצים להגריל מספר מ-0 עד מס' מסוים, ניתן לרשום רק את המספר הרצוי פחות אחד ללא צורך ב-0.

דוגמה לטבלת מעקב ל while

```

int tip;           // הטיפ מהמשלוח הנוכחי
int sum;           // סכום הטיפים המצטבר
1. sum = 0;
2. Console.WriteLine("Enter your first tip for today." +
                    " End the list of tips with -1: ");
3. tip = int.Parse(Console.ReadLine());
4. while (tip != -1)
   {
4.1.     sum = sum + tip;
4.2.     Console.WriteLine("Enter the next tip." +
                            " End the list of tips with -1: ");
4.3.     tip = int.Parse(Console.ReadLine());
   } // while
5.     Console.WriteLine("You have earned {0} shekels", sum);
} // Main
} // SumOfTips

```

	המשפט לביצוע	tip	sum	tip!=-1	פלט
1	sum = 0;	?	0		
2	Console.WriteLine("Enter your first tip...");	?	0		Enter your first tip ...
3	tip = int.Parse(Console.ReadLine());	12	0		
4	while (tip!= -1)	12	0	true	
4.1	sum = sum + tip;	12	12		
4.2	Console.WriteLine("Enter the next tip...");	12	12		Enter the next tip. ...
4.3	tip = int.Parse(Console.ReadLine());	6	12		
4	while (tip!= -1)	6	12	true	
4.1	sum = sum + tip;	6	18		
4.2	Console.WriteLine("Enter the next tip...");	6	18		Enter the next tip. ...
4.3	tip = int.Parse(Console.ReadLine());	3	18		
4	while (tip!= -1)	3	18	true	
4.1	sum = sum + tip;	3	21		
4.2	Console.WriteLine("Enter the next tip...");	3	21		Enter the next tip. ...
4.3	tip = int.Parse(Console.ReadLine());	-1	21		
4	while (tip!= -1)	-1	21	false	
5	Console.WriteLine("You have earned {0} shekels", sum);	-1	21		You have earned 21 Shekels

מילון הגדרות

בין a ל-b – כולל a וכולל b.

קטן מ- נשאר אותו דבר.

דוג': חמש קטן משבע. חמש לא קטן מחמש.

גדול מ- נשאר אותו דבר.

דוג': חמש גדול משלוש. חמש לא גדול מחמש.

לפחות-קורה כמות הפעמים המצוינת ומעלה. משתמשים בזה בד"כ כשרוצים להשתמש ב do while.

דוג': בצע את הלולאה לפחות פעם אחת – פעם אחת או שתיים, או שלוש, או יותר.

לכל היותר- עד כמות מסוימת של פעמים. מקסימום פעמים. דוג': בצע את הלולאה לכל היותר 6 פעמים. פעם 1 או 2, ... או 6. 7 כבר אסור.

סדרה עולה- סדרה אשר כל איבר הבא גדול או שווה לשני.

דוג': 1,2,3,3,4,5,7,8,8

סדרה עולה ממש- היא סדרה עולה אך ללא כפילויות.

דוג': 1,2,4,5,8

סדרה של מספרים עוקבים- סדרה אשר כל איבר הבא גדול באחד מהאיבר הקודם.

דוג': 2,3,4,5,6

כלל מתי משתמשים בכל לולאה

משתמשים בלולאת for כאשר יודעים מראש כמה פעמים הלולאה תתבצע. לעומת זאת, משתמשים בלולאת while כאשר לא יודעים כמה פעמים היא תתבצע אך יודעים איזה תנאי צריך להתקיים על מנת שהיא תיעצר.

String

ניתן לגשת לכל מקום במחרוזת על ידי כתיבת השם שלה,
וסוגריים מחוברות אשר מכילות את מיקום התו המבוקש:

```
string str = "abcd";
char first = str[0];
char last = str [ str.Length - 1 ];

string str = "abcd";
string str2 = "fdda";
int num = str.IndexOf("bc");           //IndexOf(string)
int num2 = str.IndexOf('c');           //IndexOf(char)
int num3 = str.LastIndexOf("d");       //LastIndexOf(string)
bool equals = str.Equals(str2);        //Equals
int comparison = str.CompareTo(str2);  //CompareTo
string str3 = str.ToUpper();           //ToUpper
string str4 = str3.ToLower();          //ToLower
string str5 = str.Substring(0, 2);      //Substring(int a, int b)
string str6 = str.Substring(1);        //Substring(int a)
string str7 = str.Insert(2, "fsdf");   //Insert
string str8 = str.Remove(0, 2);        //Remove
string str9 = str.Replace("cd", "sd"); //Replace
```

דוגמה	הסבר	קותרת הפעולה	בקצור
<pre>string str = "animal"; int index; index = str.IndexOf("n"); // index 1 יהיה index = str.IndexOf('n'); // index 1 יהיה index = str.IndexOf("ima"); // index 2 יהיה index = str.IndexOf("nnn"); // index 1- יהיה</pre>	<p>פעולה המקבלת מחזורת או תו, ומחפשת בתוך המחזורת שעליה מופעלת הפעולה את המיקום הראשון שבו מופיעה המחזורת או התו שהתקבלו.</p> <p>הפעולה תחזיר את המיקום. היא תחזיר את הערך -1, אם החיפוש נכשל.</p>	<pre>int IndexOf(char value) int IndexOf(string value)</pre>	<p>למצוא בתוך מחזורת</p>
<pre>string s1 = "Dominono"; int a = s1.LastIndexOf('o'); //a=7 int b = s1.LastIndexOf("o"); //b=7 int c = s1.LastIndexOf("no"); //c=6</pre>	<p>מחזיר מספר המסמן מיקום של החל מסוף</p>	<pre>int LastIndexOf(char value) int LastIndexOf(string value)</pre>	<p>מצא מופע אחרון בתוך מחזורת</p>
<pre>string s1 = "Dominono"; bool a = s1.StartsWith("do"); //a=false bool b = s1.StartsWith("Do"); //b=true</pre>	<p>מחזירה true אם בהתאם במקרה והמחזורת מתחילה במחזורת שבפרמטר.</p>	<pre>bool StartsWith(string value)</pre>	<p>האם מחזורת מתחילה במשורה?</p>
<pre>string s1 = "Dominono"; bool c = s1.EndsWith("no"); //c=true bool d = s1.EndsWith("k"); //d=false</pre>	<p>מחזירה true אם בהתאם במקרה והמחזורת מתחילה במחזורת שבפרמטר.</p>	<pre>bool EndsWith(string value)</pre>	<p>האם מחזורת מסתיימת במשורה?</p>

<pre>string s1 = "Olga"; string s2 = "Olga"; string s3 = "Golda"; bool b; b = s1.Equals(s2); //b is true b = s1.Equals(s3); //b is false b = s1.Equals("Olga");//b is true b = (s1 == s2); //b is true b = (s1 == s3); //b is false b = (s1 == "Olga"); //b is true</pre>	<p>פעולה המקבלת מחזורת, ומחזירה true אם המחזורת שעליה הופעלה הפעולה והמחזורת שהתקבלה שוות זו לזו בדיוק. אחרת, היא מחזירה false.</p> <p>פעולה זו זהה לפעולת ההשוואה ==</p>	<p><code>bool Equals (string value)</code></p>	<p>משווה מחזורות מחזיר שווה או לא</p>
<pre>string str1 = "Goodbye"; int x; x = str1.CompareTo("Abba"); // x=-1 x = str1.CompareTo("Zoo"); // x=-1 x = str1.CompareTo("a cat"); // x= 1 חיובי אחר x = str1.CompareTo("goodbye"); // x= 1 חיובי אחר x = str1.CompareTo("Goodbye"); // x= 0 x = str1.CompareTo(str1); // x= 0</pre>	<p>פעולה המקבלת מחזורת, ומשווה אותה למחזורת שעליה הופעלה הפעולה. אם הן שוות זו לזו מוחזר הערך 0. אם המחזורת שעליה מופעלת הפעולה קודמת למחזורת שהתקבלה בסדר מילוני, יוחזר מספר שלם שלילי. אם המחזורת שעליה מופעלת הפעולה, מופיעה אחר המחזורת שהתקבלה בסדר מילוני, יוחזר מספר שלם חיובי.</p>	<p><code>int CompareTo (string s2)</code></p>	<p>משווה מחזורות מחזיר שווה, גדולה מ- או קטנה מ-</p>
<pre>string myString = "Hello World"; string s2 = myString.ToLower(); ערך המשנה לא ישתנה "hello world" יהיה S2</pre>	<p>פעולה שיוצרת מחזורת זהה למחזורת שעליה היא מופעלת, ובה כל האותיות מוחלפות באותיות קטנות. הפעולה מחזירה את המחזורת החדשה.</p>	<p><code>string ToLower ()</code></p>	<p>המרה לאותיות לקטנות.</p>
<pre>string myString = "Hello World"; string s2 = myString.ToUpper(); ערך המשנה לא ישתנה "HELLO WOLRD" יהיה S2</pre>	<p>פעולה שיוצרת מחזורת זהה למחזורת שעליה היא מופעלת, ובה כל האותיות מוחלפות באותיות גדולות. הפעולה מחזירה את המחזורת החדשה.</p>	<p><code>string ToUpper ()</code></p>	<p>המרה לאותיות לגדולות.</p>

<pre>string st = "abcdefg"; string st1 = st.Substring(3); string st2 = st.Substring(6); "defg" יהיה st1 "g" יהיה st2</pre>	<p>פעולה שיצירת מחזורת זהה לתת-מחזורת המתחילה מהמקום ה-k של המחזורת שעליה הפעולה מופעלת ועד סופה. הפעולה מחזירה את המחזורת החדשה.</p>	<pre>string Substring(int k)</pre>	<p>תת מחזורת מקום מסוים עד הסוף</p>
<pre>string st = "abcdefg"; string st1 = st.Substring(2,4); "cdef" יהיה st1 -2 מאיזה מקום, -4 כמה תווים</pre>	<p>פעולה שיצירת מחזורת זהה לתת-מחזורת המתחילה מהמקום ה-k של המחזורת עליה היא מופעלת ואורכה הוא s. הפעולה מחזירה את המחזורת החדשה. לשים לב: לא לוקחים מחזורת ממקום k עד מקום s, אלא לוקחים ממקום k את s תווים.</p>	<pre>string Substring(int k, int s)</pre>	<p>תת מחזורת מקום מסוים באורך מסוים</p>
<pre>string s1 = "12345"; string s2 = s1.Insert(2, "abc"); 12345 נשאר s1 abc34512 נהיה s2</pre>	<p>הפעולה מקבלת מספר שלם $start$ ומחזורת s, ומחזירה מחזורת חדשה בה הוכנסה במחזורת עליה מופעלת הפעולה, המחזורת s החל מהמקום $start$. אם המקום $start$ לא נמצא במחזורת מתקבלת הודעת חריגה.</p>	<pre>string Insert(int start, string s)</pre>	<p>מכניס (מחזיר) מחזורת</p>
<pre>string s1 = "abaabcas"; string s2 = " "; bool b; b = string.IsNullOrEmpty(s1); //b is false b = string.IsNullOrEmpty(s2); //b is true</pre>	<p>הפעולה מקבלת מחזורת s ובודקת אם המחזורת ריקה. אם כן, היא מחזירה $true$. אם לא, היא מחזירה $false$.</p>	<pre>bool IsNullOrEmpty(string s)</pre>	<p>מחזורת ריקה ?</p>

<pre>string s1 = "abaabcas"; string s2 = s1.Remove(1, 4); // "acas" יהיה s2 נחתך חלק ממקום 1 עד מקום 4 (כולל)</pre>	<p>הפעולה מקבלת מספר שלם start ומספר שלם count ומחזירה מחרוזת חדשה המתקבלת על ידי הסרת count תווים מן המקום start מן המחרוזת עליה היא מופעלה.</p> <p>אם המקום start לא נמצא במחרוזת או שאין די תווים באורך המקום count מתקבלת הודעת חריגה.</p> <p>הפעולה מקבלת שני תווים ומחזירה מחרוזת חדשה בה כל מופע של התו c1 מוחלף בתו c2 במחרוזת עליה היא מופעלת.</p>	<pre>string Remove(int start, int count) string Replace(char val1, char val2)</pre>	<p>חותך חלק ממקום עד מקום (מותר גם לחתוך חלק מהאמצע)</p> <p>יוצר מחרוזת חדשה בה כל תו val1 הוחלף בתו val2</p>
<pre>string s1 = "abaabcas"; string s2 = s1.Replace("a", "z"); string s3 = s1.Replace("ab", "fd"); string s4 = s1.Replace("ab", "_"); string s5 = s1.Replace("a", ""); "abaabcas" נשאר s1 "zbzzbczs" מקבל s2 "fdafdcdas" מקבל s3 "bbcs" מקבל s4</pre>	<p>מחזירה מחרוזת כאשר כל מופעים של val1 הוחלפו ל val2</p>	<pre>string Replace(string val1, string val2)</pre>	<p>יוצר מחרוזת חדשה בה כל תו val1 הוחלף ב- val2</p>

<pre>char[] separators = { ' ', '.', '!', ',', '?', ';', ':', ' ' }; string s = "Are you ready? We have to: think, code, enjoy."; string[] words = s.Split(separators); s לא משתנה. words[0] = "Are" words[1] = "you" words[2] = "ready" words[3] = "We" words[4] = "have" words[5] = "to" words[6] = "think" words[7] = "code" words[8] = "enjoy" string text = "An example of split"; string[] words2 = text.Split(' '); s לא משתנה. words2[0] = "An" words2 [1] = "example" words2 [2] = "of" words2 [3] = "split"</pre>	<p>הפעולה מחזירה מערך של מחרוזות המחולקים על ידי .separators</p>	<p><code>string[] split (char[] separators)</code></p>	<p>מחזיר מערך של מחרוזות המחולקים על ידי אחד האיברים בתוך separators</p>
--	--	--	--

פונקציות

הגדרת פעולה:

```
Static (שם פרמטר) [שם פעולה] [טיפוס_ערך_מוחזר]
{
    גוף פעולה
    return ערך מוחזר על ידי פעולה
}
```

פונקציה היא כלי המשמש אותנו על מנת שלא נצטרך לחזור על אותה פעולה כל פעם מחדש.

ישנן כמה סוגי פונקציות, כולן מחזירות ערך מסוים (מספר, מחרוזת, תו) פרט לסוג אחד, פונקציה ריקה, אשר לא מחזירה דבר.

דרך הפונקציה אפשר להחזיר רק ערך אחד (return), אלא אם נעשה שימוש ברפרנס .ref.

```
static double average(int num1, int num2, int num3)
//פעולה מקבלת שלושה מספרים שלמים
//פעולה מחזירה ממוצע שלהם
{
    double avg = (double)(num1 + num2 + num3) / 3;
    return avg;
}
```

```
static bool isMoreFromAvg(double av)
//פעולה מקבלת ממוצע ציונים
//פעולה מחזירה אמת אם ממוצע גדולה מ-80
{
    bool flag = false;
    if (av > 80)
        flag = true;
    return flag;
}
```

טבלת מעקב על פונקציה

```
static void Main(string[] args)
{
    int a = 1, z;
    for (int b = 1; b < 4; b++)
    {
        z = Change(a, ref b);
        Console.WriteLine("a={0}, b={1}", a, b);
    }
}

public static int Change(int x, ref int y)
{
    y = y + x;
    x = x + y;
    return x + y;
}
```


פקודה	a	b	b < 4	פלט
1 int a = 1	1			
2 for (int b=1; b<4; b++)		1	true	
				Change
				x y
				1 1
5 y = y + x;		2		2
6 x = x + y;				3
2.1 Change(a, ref b);				
2.2 Console.WriteLine ("a={0}, b={1}", a, b);				a=1, b=2
2 for (int b=1; b<4;b++)		3	true	
				Change
				x y
				1 3
5 y = y + x;		4		4
6 x = x + y;				5
2.1 Change(a, ref b);				
2.2 Console.WriteLine ("a={0}, b={1}", a, b);				a=1, b=4
2 for(int b=1;b<4;b++)		5	false	

הגדרות של פונקציות

פונקציה מקבלת- הערכים מועברים אל הפונקציה על ידי פרמטרים.

קלט של פונקציה- הערכים מועברים אל הפונקציה על ידי פרמטרים.

הפונקציה קולטת- הפונקציה קולטת מהמשתמש את הקלט.

הפונקציה מחזירה- הערך שמחזירה הפונקציה.

פעולה פנימית – פעולה של מחלקה.

פעולה חיצונית – Main או פעולה static אחרת.

כתוב תוכנית – מתחילים מ: "using".

כתוב קטע קוד – כמה שורות של קוד. אין חובה לכתוב פונקציה.

מערכים

מערך הוא טיפוס נתונים מורכב המכיל אוסף של איברים מאותו טיפוס בעל שם מזהה אחת. ניתן לפנות לכל איבר במערך לפי האינדקס שלו. האינדקס (מצייין) המערך הוא מקומו הסידורי של איבר במערך.

אינדקס	0	1	2	3	4	5	6	7	8
ערך	34	45	23	34	1	12	8	5	2

הגדרת מערך מתבצע על ידי שני שלבים:

1. הגדרת טיפוס מערך.

; שם המערך [] טיפוס איברי המערך

טיפוס איברי המערך – מגדיר את טיפוס של כל אחד מאיברי המערך
הסימון [] – מצביע על כך שמוגדר מערך ולא משתנה
שם מערך – שם משמעותי המזהה את המשתנה

מערך בעל 8 אברים מטיפוס תו

```
char [] characters;  
characters = new char [8];
```

Or

```
char [] characters= new char [8];
```

מערך בעל 25 איברים מטיפוס שלם

```
int[] marks;  
marks = new int[25];
```

Or

```
int[] marks = new int[25];
```

2. בניית מערך

[מספר איברים במערך] טיפוס איברי המערך **new** = שם המערך
new - מילה שמורה שגורמת להקצאת מקום בזיכרון עבור מערך
מספר איברים במערך (גודל מערך) – מספר ערכים שאפשר לשמור במערך

דוגמאות:

מערך בעל 40 אברים מטיפוס ממשי

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

Or

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

כללים לשימוש במערך

- פניה לכל איבר במערך נעשית על ידי ציון שם המערך ובתוך סוגריים מרובעים מספר סידורי (מכונה מציין) של איבר עליו רוצים לפנות. [מציין] שם מערך
- במערך בגודל n (מערך עם n איברים) תא הראשון הוא 0 ותא אחרון הוא $n-1$
- כל איבר במערך הוא משתנה לכול דבר ואפשר לבצע איתו כל אותם פעולות שאפשר לבצע עם משתנים רגילים כמו : לקלוט ערך, לשים בו ערך, להציג ערך, לשתף בפעולות מתמטיות, לשלוח לפעולות כפרמטר.
- ניסיון פניה אל מקום במערך שאינו בגבולות $n-1-0$ נקראת חריגה מגבולות המערך וגורמת לשגיאת קומפילציה.

שלבי בדיקה כללית של המבחן:

- 1) לבדוק = מול == . בכל מקום בו מופיע = לבדוק האם לא אמור להיות ==.
- 2) לבדוק >= ו- > ולהיפך. בכל מקום לבדוק האם > או >=, האם < או <=.
- 3) לבדוק שלכל { יש } ובמקום הנכון. לסמן עם קוים.
- 4) בכל חלוקה טיפוסים מתאימים: int, double.
- 5) בטבלת המעקב יש כל העמודות (פקודות, משתנים, תנאים, פלט), אם לולאה – אז פקודות מופיעות כמה פעמים.
- 6) לקרוא משימה עוד פעם, לסמן עם מרקר מה הפלט, מה הקלט, לבדוק שיש חישוב של כל דבר בדיוק כמו שמבקשים במשימה.
- 7) ט. כניסה ויציאה – שיהיו.
- 8) הערות – שיהיו.