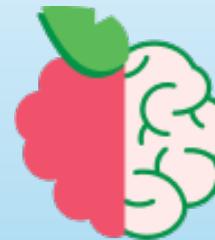


PeTel Tester

C#

ממשק לבדיקת קוד בסיסי שארפ
גַּלְעָד מִרְקָמִן



PeTel



מכון
ויצמן
למדעים

WEIZMANN
INSTITUTE
OF SCIENCE



מכון
ויצמן
למדעים

WEIZMANN
INSTITUTE
OF SCIENCE

כללי

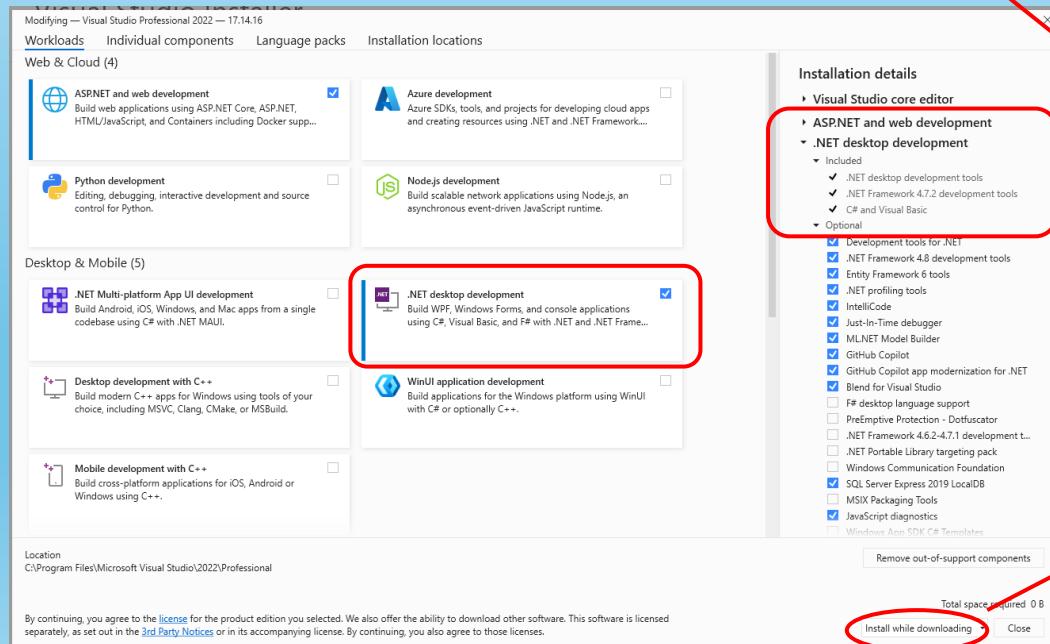
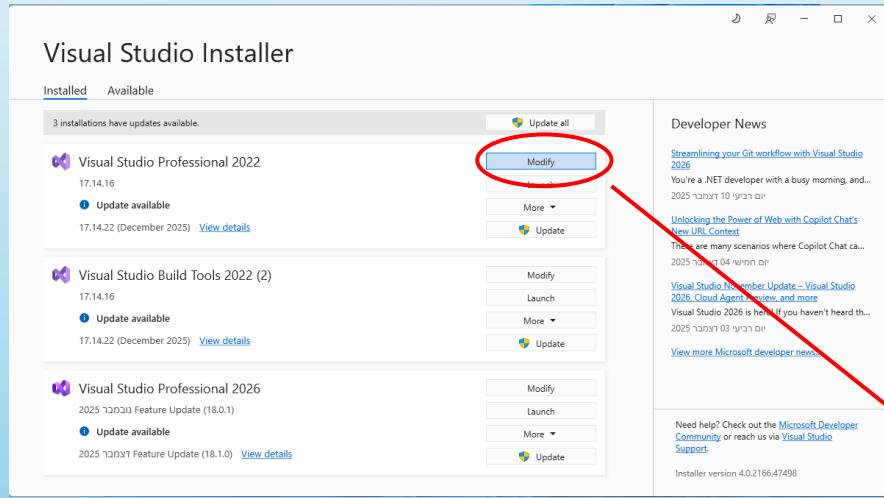
- ממשק בדיקת הקוד כולל שלושה חלקים:
 - **Case Tester** – בדיקה של תוצאות הרצת קוד התלמיד לעומת פתרון המורה.
 - **Code Tester** – בדיקה של תחביר התשובה של התלמיד (אם יש קרייה רקורסיבית, האם הפונקציה סטטית ועוד').
 - **AI Tester** – בדיקת הקוד באמצעות מודל בינה מלאכותית (בפיתוח)
- פיתוח בדיקת הקוד מתבצע בסביבת העבודה **Visual Studio**
 - הרצת בדיקות על פתרונות אפשריים של תלמידים.
 - לאחר סיום הפיתוח העלה של הקבצים לסביבת פטל.



התקנות



הכנה של ויזואל סטודיו



• סביבת הפיתוח שלנו היא Visual Studio. התקינה חינמית: Visual Studio Community

• על מנת לוודא כי הרכיבים הנדרשים מותקנים יש להריץ את Visual Studio .Installer .

- לוודא ש .Net desktop development מותקן.

• לוודא שככל את .Net Framework 4.7.2 .
• אם חסר – להתקין.

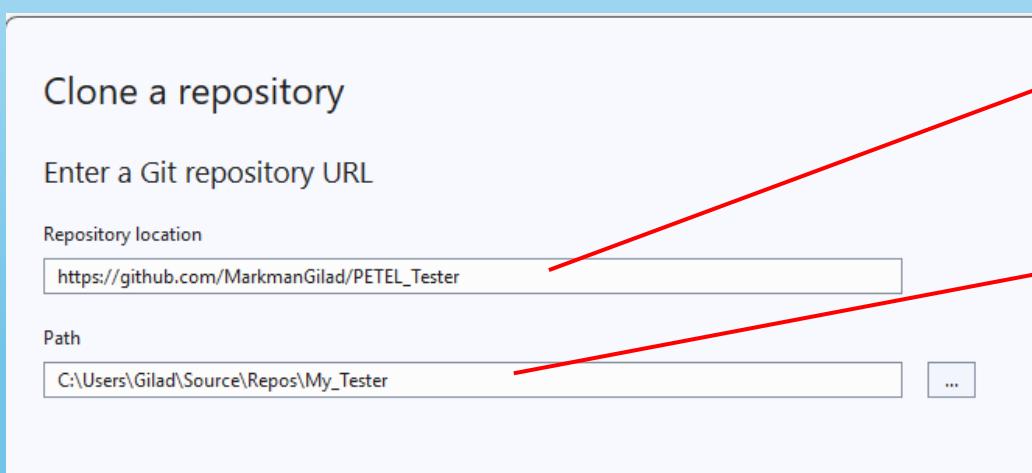
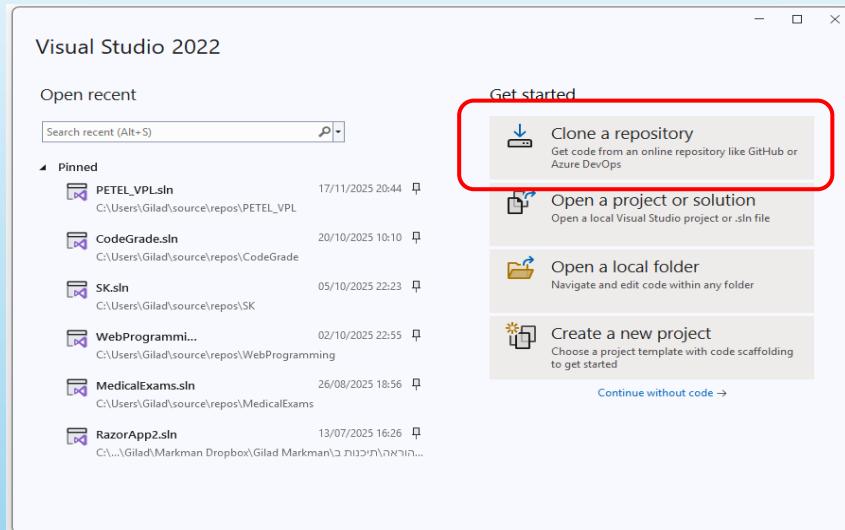
התקנה של הממשק

- הממשק נמצא GitHub פומבי לפי הכתובת הבאה:
https://github.com/MarkmanGilad/PETEL_Tester
- ניתן להוריד את הממשק בשתי דרכים חלופיות:
 - לבצע cloning לפי הכתובת לעיל (נדרשת התקנה של Git במחשב).
 - הורדת הפרויקט כקובץ zip, שמירת הקובץ על המחשב האישי, פתיחת הקובץ, וריצתו.



- נדגים את שתי האפשרויות.

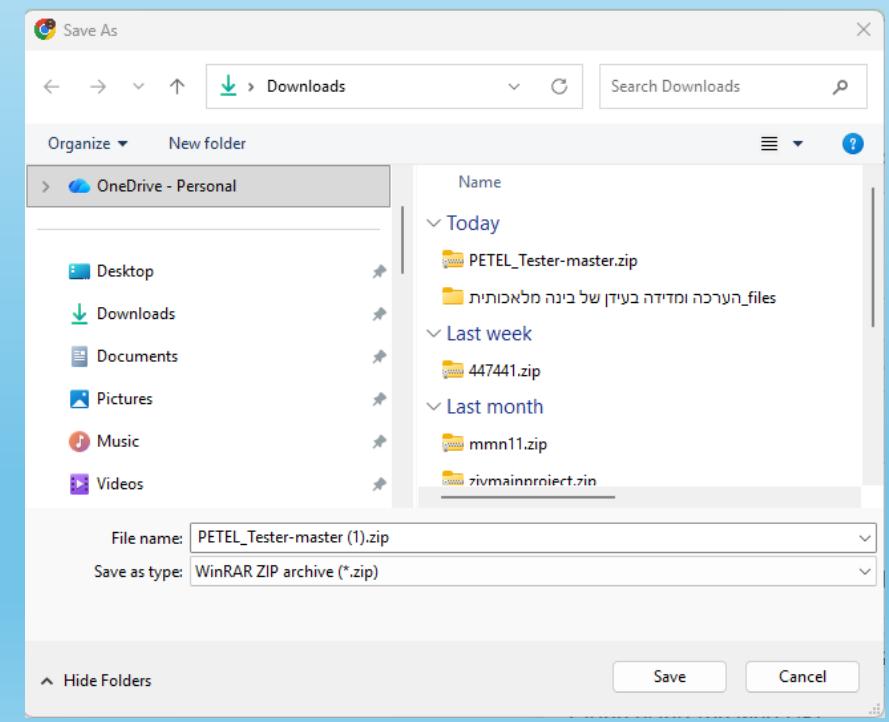
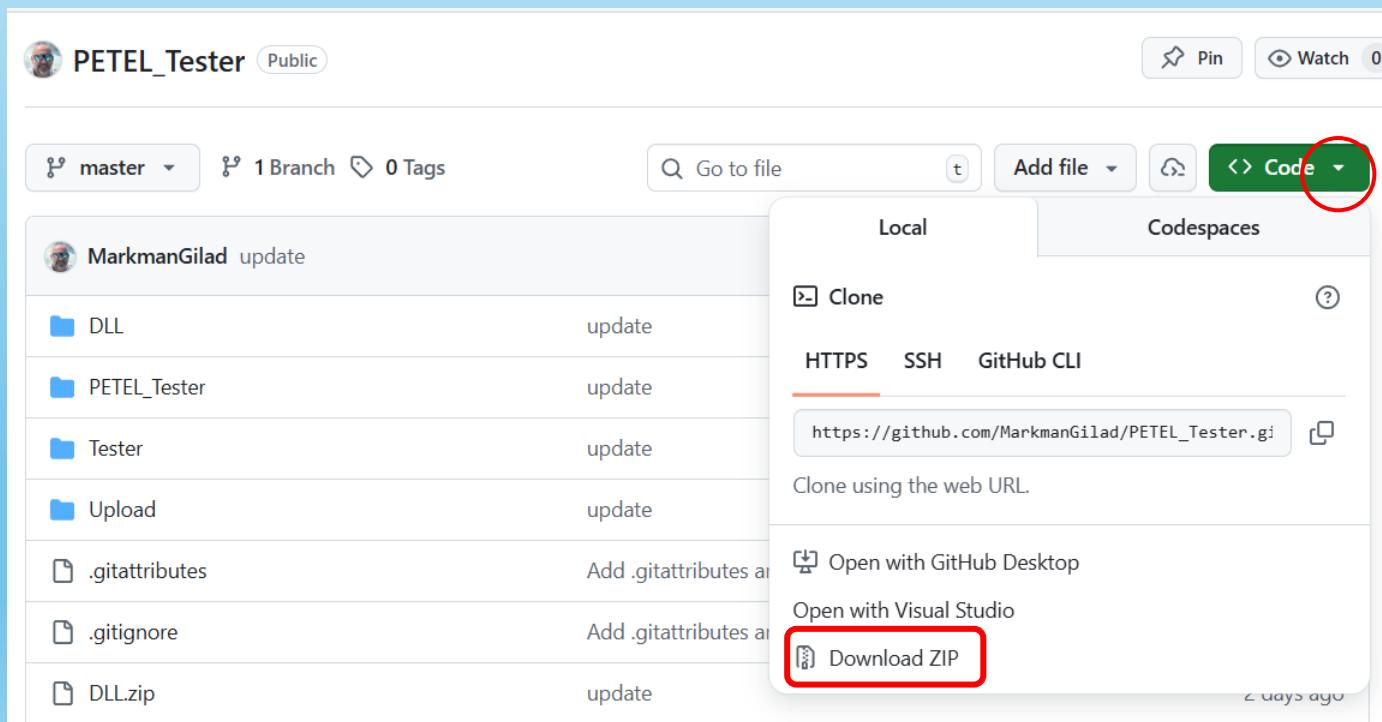
Cloning



- ב מסך הפתיחה של ויזואל סטודיו 2022 יש לבחור **clone a repository**.
- בתיבת המיקום יש להכניס את כתובות **הפרויקט בGIT**:
https://github.com/MarkmanGilad/PETEL_Tester
- ניתן להשאיר את הנתיב לשימרת **הפרויקט במחשב שלנו ללא שינוי או** לבחר **מקום אחר**.
- ללחוץ על כפתור **clone**.

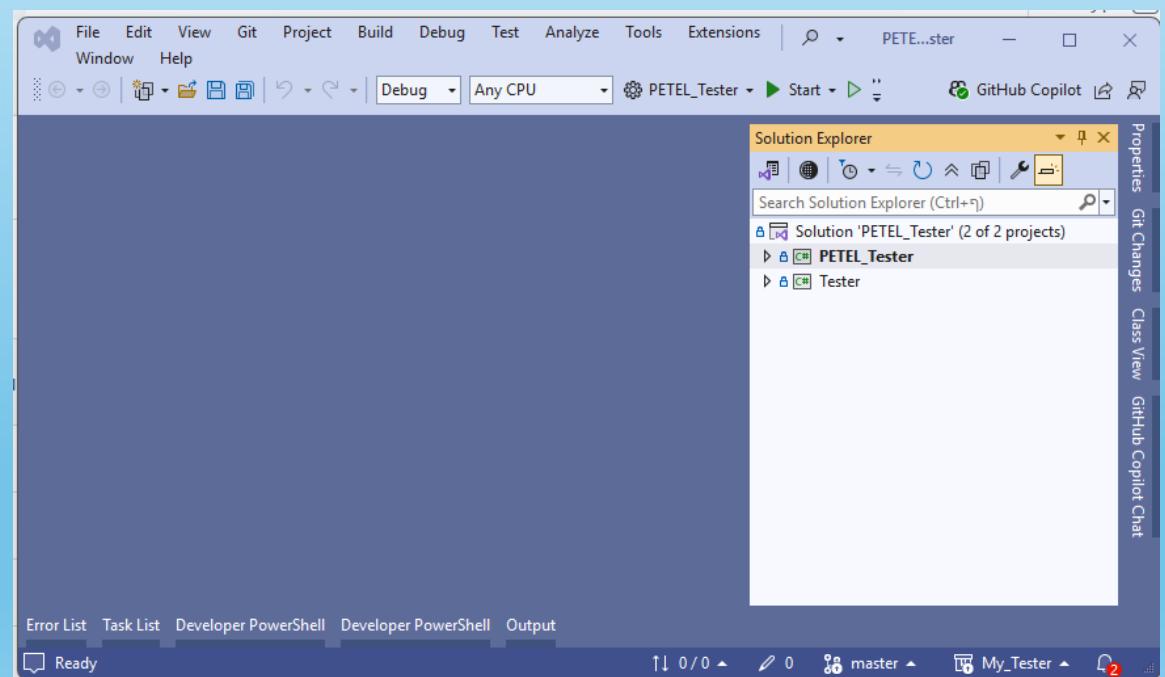
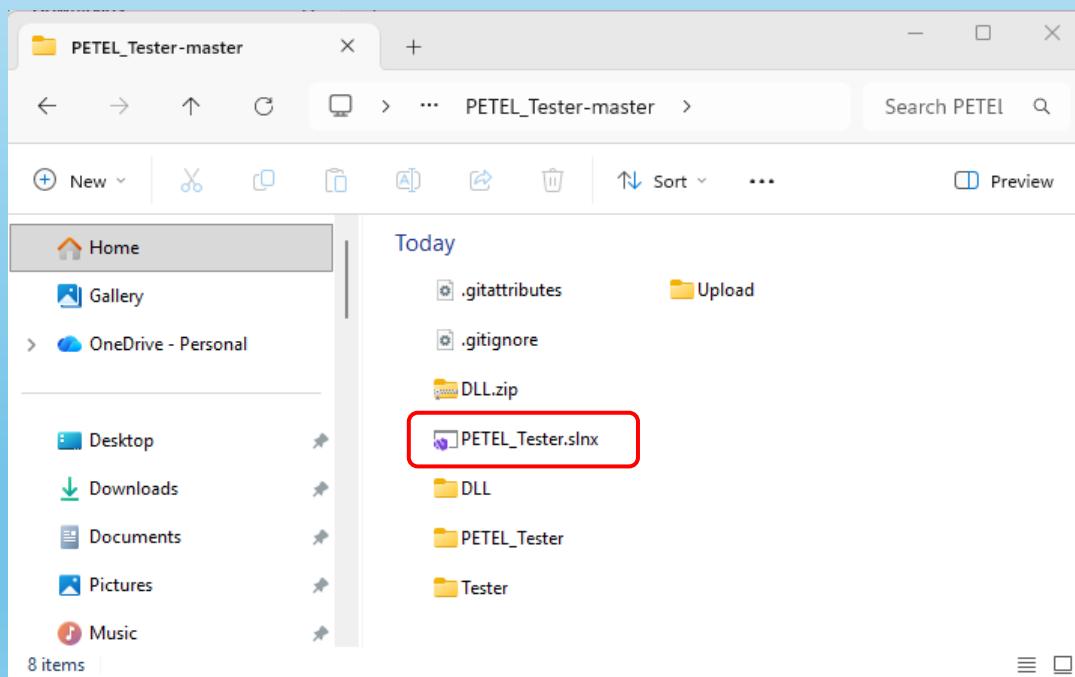
הורדת הפרויקט למחשב (חולופי)

- באתר גיטהוב יש https://github.com/MarkmanGilad/PETEL_Tester. Download Zip , ולבחרו Code , לחוץ על החץ ליד Code , ולבחרו .Download Zip
- לשמור את הקובץ במחשב שלנו.



המשך הורדת הפרויקט

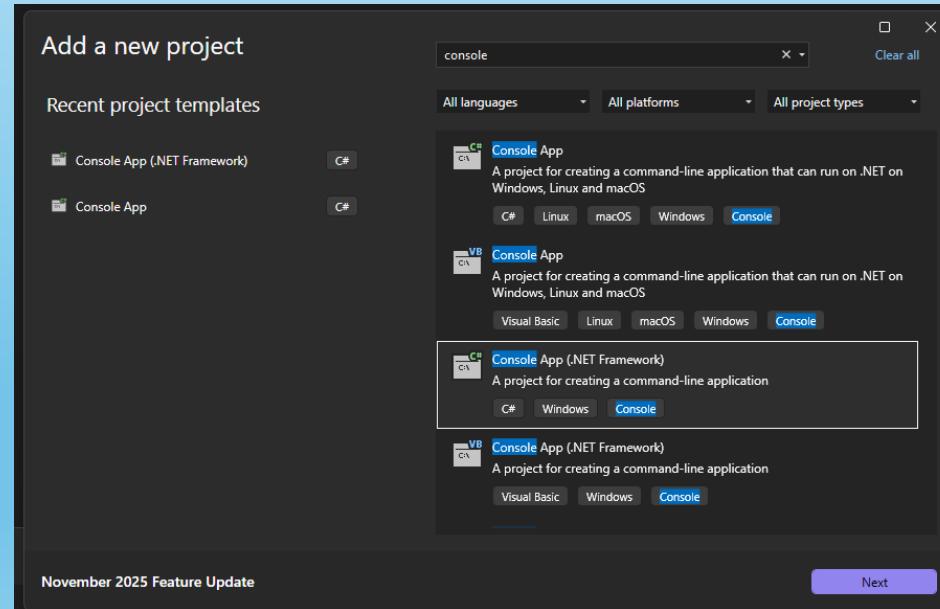
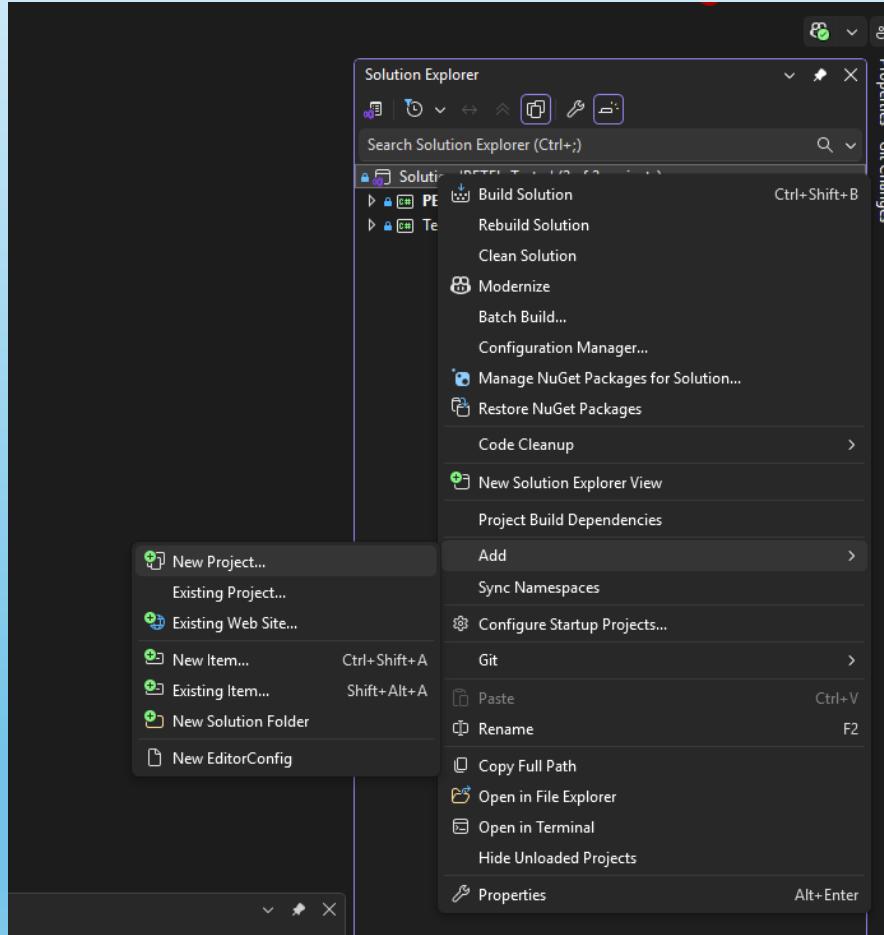
- לאחר הורדת קובץ Zip למחשב יש לפתח (Extract).
- להכנס למספרה PETEL_Tester-master ולהזיכת כפולה על תفعיל את הפרויקט.



הכנת סביבת העבודה למשימה חדשה

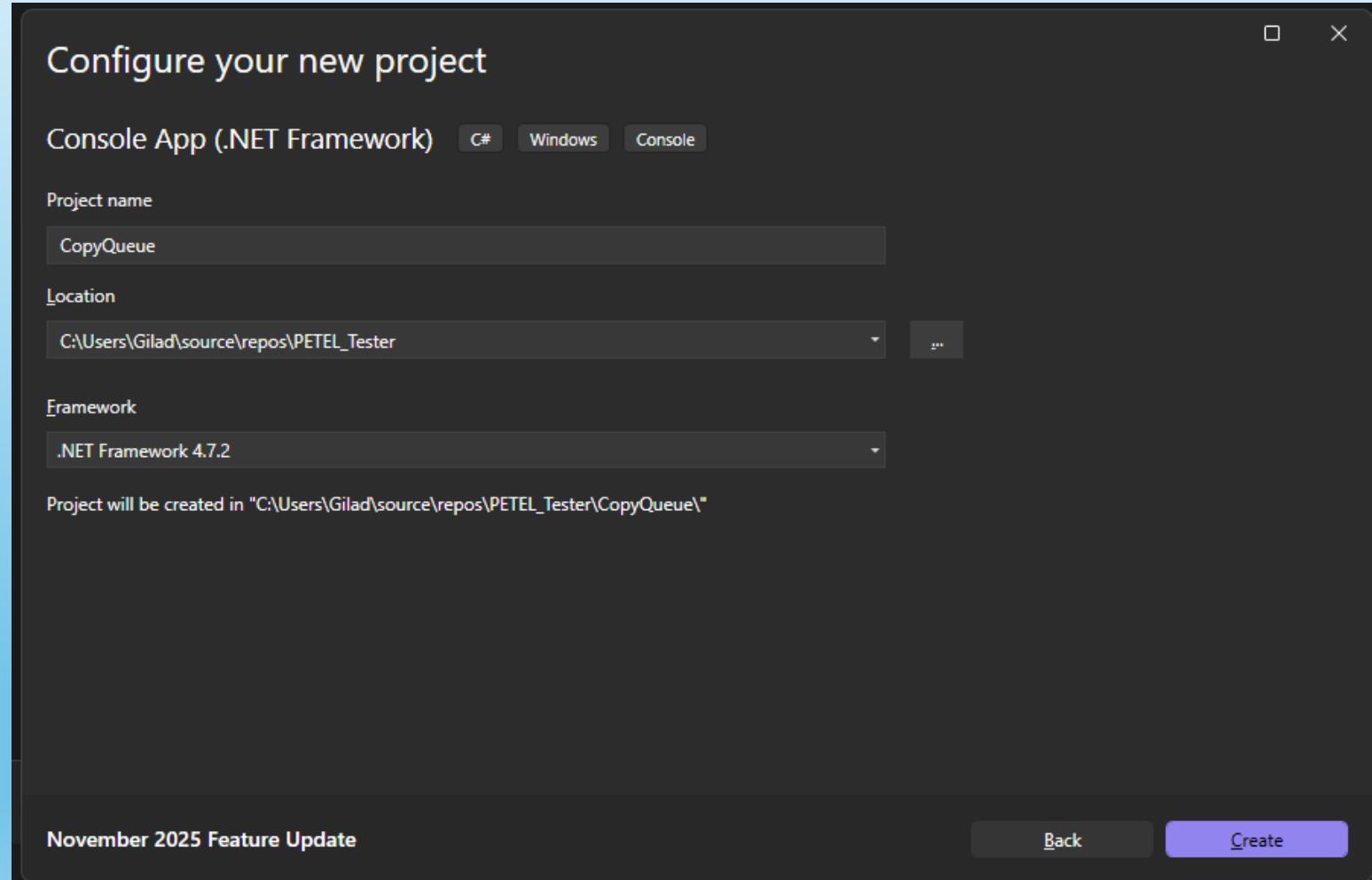


יצירת פרויקט חדש



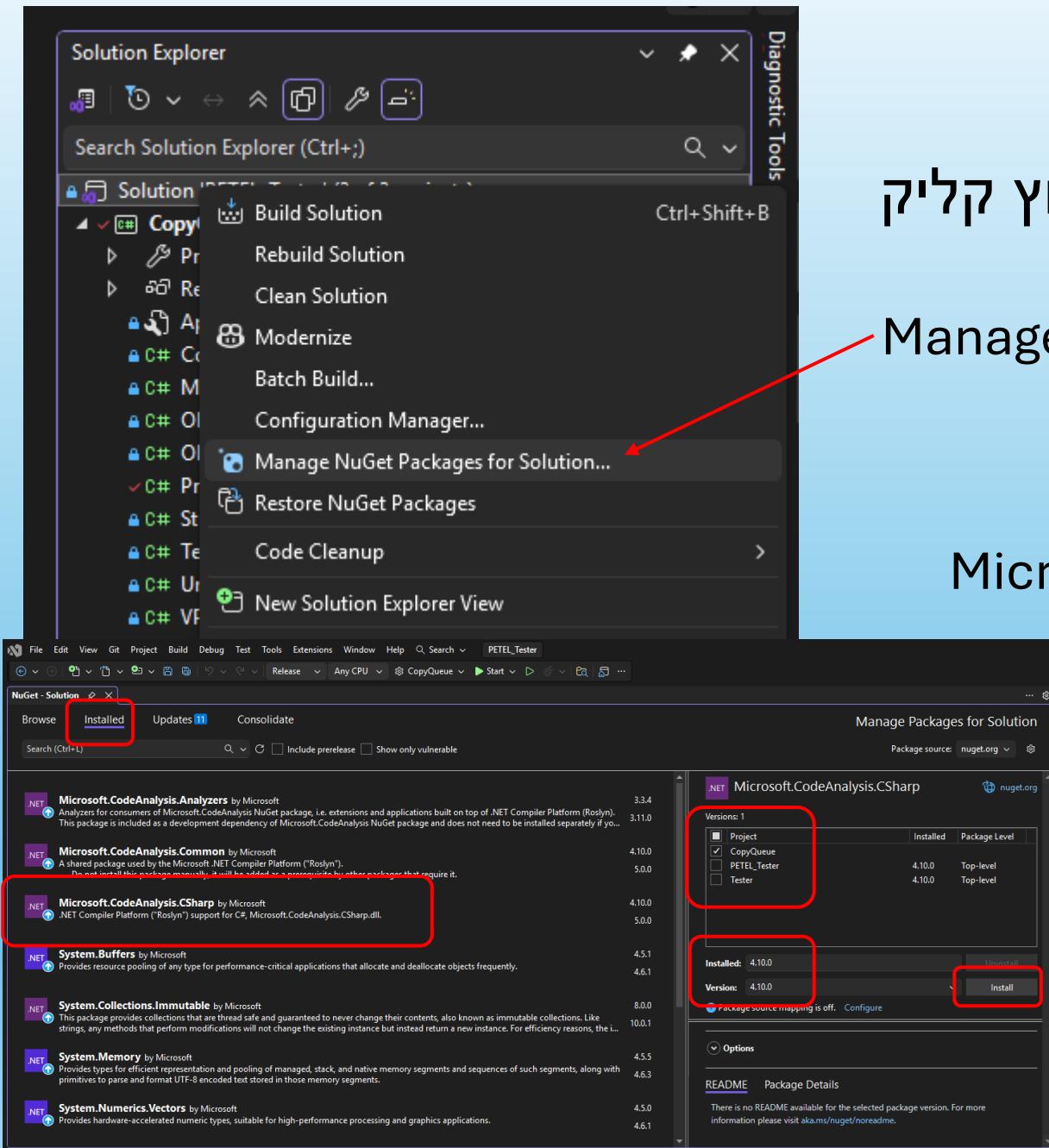
- לחיצה ימנית על ה Solution
Add -> New Project ... •
- לבחור פרויקט מסווג:
Console App (.Net Framework) C#
- ללחוץ **Next**

יצירת פרויקט חדש - המשך



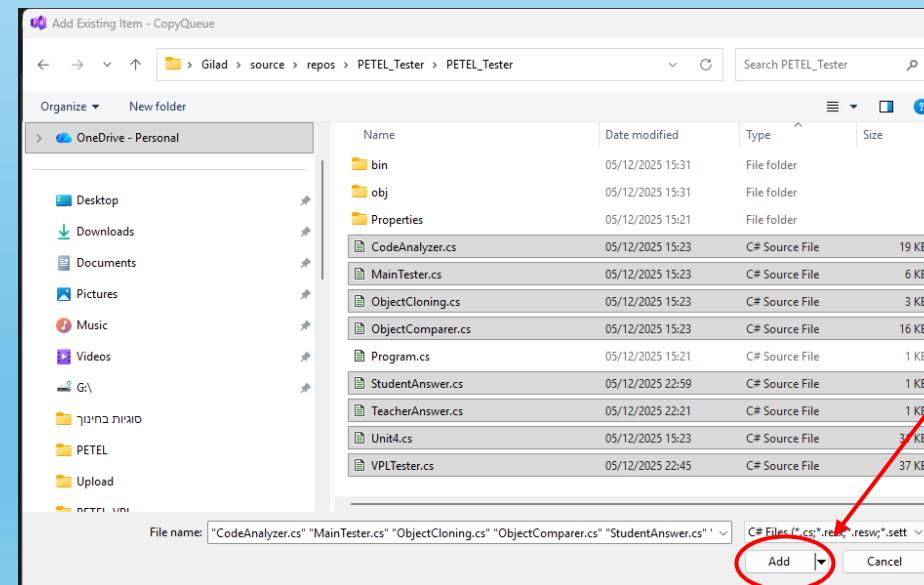
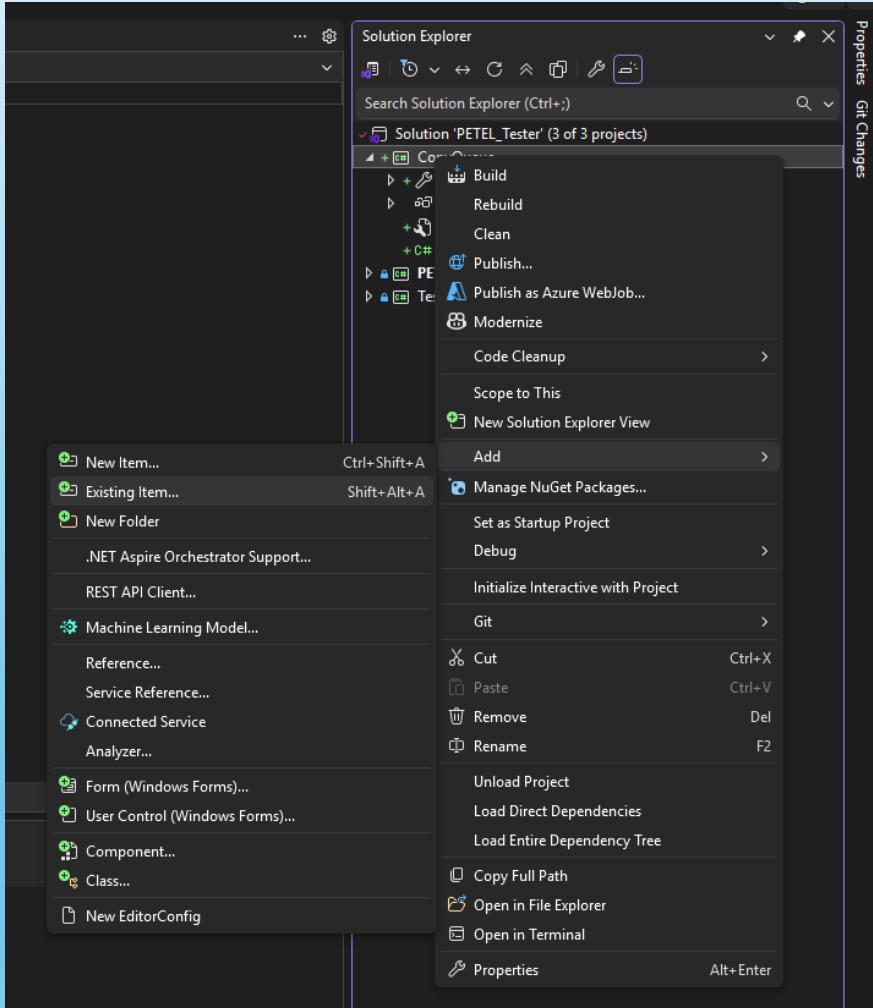
- לבחור שם לפרויקט.
- ללחוץ **Create**.

התקנת ספריות נחוצות



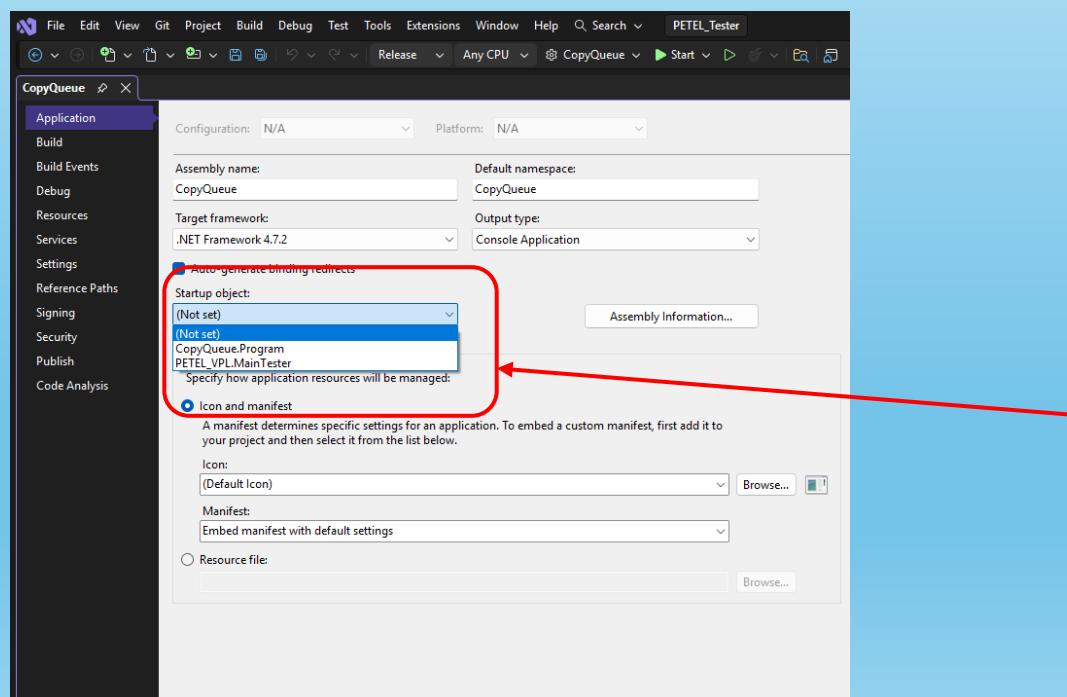
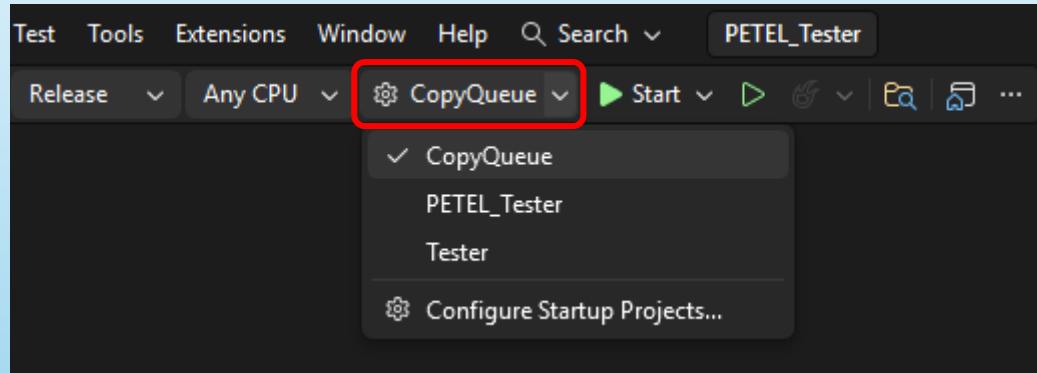
- על מנת להתקין ספריות נחוצות נלחץ קлик ימni על ה Solution וນבחר:
 - Manage NuGet Packages for Solution...
 - ...
- בחלון שיפתח:
 - נבחר `Installed`.
 - נבחר `Microsoft.CodeAnalysis.CSharp`.
 - נסמן ✓ בשם הפרויקט שלנו.
 - נודא שהגרסה היא `4.10.0`.
 - ***** אסור לעדכן גירסה `4.10.1`.
- נלחץ `Install`.
- נאשר `I Accept`

העתקה קבצים לפרויקט החדש



- לחיצה ימנית על הפרויקט ולבחרו:
 - Add -> Existing Item ...
- להגיע לספריה .PETEL_Tester/PETEL_Tester
- לסמן ולהוסיף את כל הקבצים למעט Program.cs
- ללחוץ Add

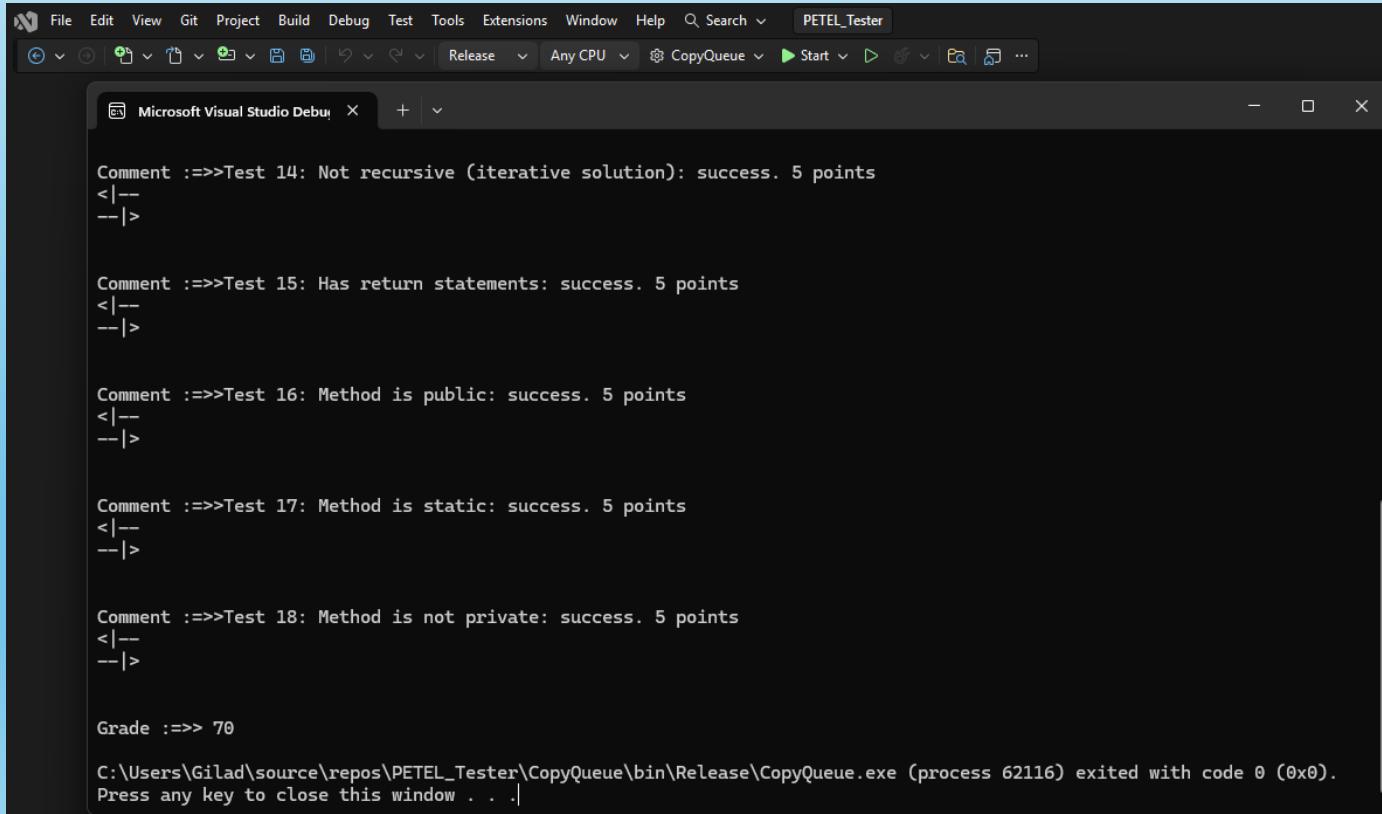
קביעת פרויקט ונקודת כניסה



- נבחר את הפרויקט שאנו מרכיבים.
- הפרויקט כולל שתי פועלות Main:
 - במחלקה Program.cs
 - במחלקה MainTetser.cs
- קליק ימני על שם הפרויקט ולהזיכה על Properties.
- בחירת Main_Tetser כנקודת הכניסה שלנו.
- נשמר את ההגדרות s-ctrl

הפרויקט מוכן לפיתוח

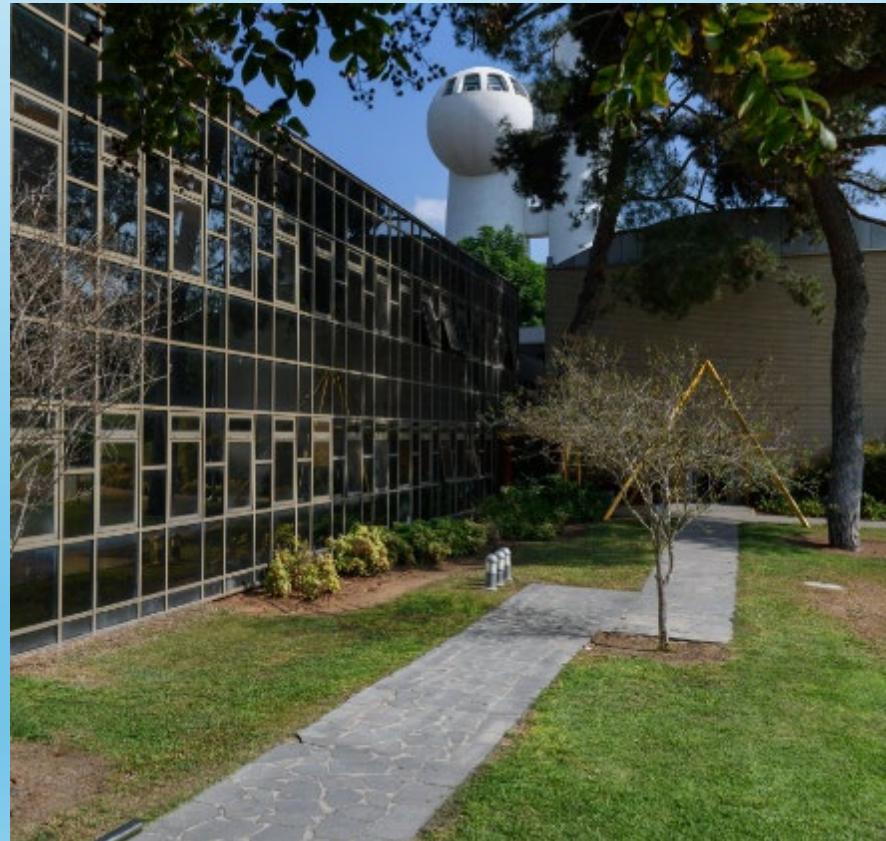
- הפרויקט כולל את כל הקבצים הנדרשים להכנת משימה חדשה.
- ניתן להריץ את הפרויקט ctrl-F5 וראות את תוצאות הרצה של תרגיל לדוגמה: `.countRemoveItem()`.



The screenshot shows the Microsoft Visual Studio Debug window. The title bar says "Microsoft Visual Studio Debug". The menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Tools, Extensions, Window, Help, and Search. The toolbar has icons for file operations like Open, Save, and Build. The status bar at the bottom shows the path "C:\Users\Gilad\source\repos\PTEL_Tester\CopyQueue\bin\Release\CopyQueue.exe (process 62116) exited with code 0 (0x0)." and the message "Press any key to close this window . . .". The main window displays several test comments:

```
Comment :=>Test 14: Not recursive (iterative solution): success. 5 points
<|-->
Comment :=>Test 15: Has return statements: success. 5 points
<|-->
Comment :=>Test 16: Method is public: success. 5 points
<|-->
Comment :=>Test 17: Method is static: success. 5 points
<|-->
Comment :=>Test 18: Method is not private: success. 5 points
<|-->
Grade :=> 70
```

Tester בנית

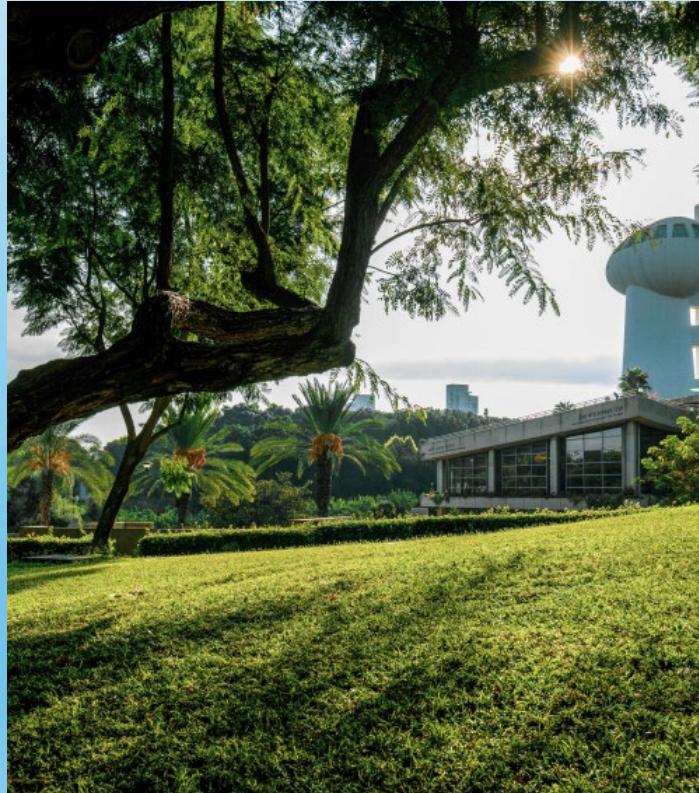


כללי

- **בנייה Tester מחייבת עדכון שלושה קבצים:**
 - קובץ הבדיקות - MainTeter.cs
 - קובץ הפתרון המשמש בסיס להשוואה לקוד התלמיד - TeacherAnswer.cs
 - דוגמה לפתרון תלמיד לבדיקת גילוי שגיאות שונות - StudentAnswer.cs
- אם מבקשים לבדוק את תוצאות הרצת הפתרונות של המורה או של התלמיד, ניתן לזמן אותם מהקובץ Program.cs ולהגידו לו אותו כנקודות כניסה.
- אם רוצים לבדוק כיצד הטスター בודק את פתרון התרגילים, מגידים את MainTester נקודת כניסה וMRIIZIM את הקוד.
- **מחלקה Unit4** כוללת את כל המחלקות הנדרשות ליחידה מבנה נתונים ובנוסף כוללת Helper ליצירת והדפסת מבני נתונים (ראו בהמשך).

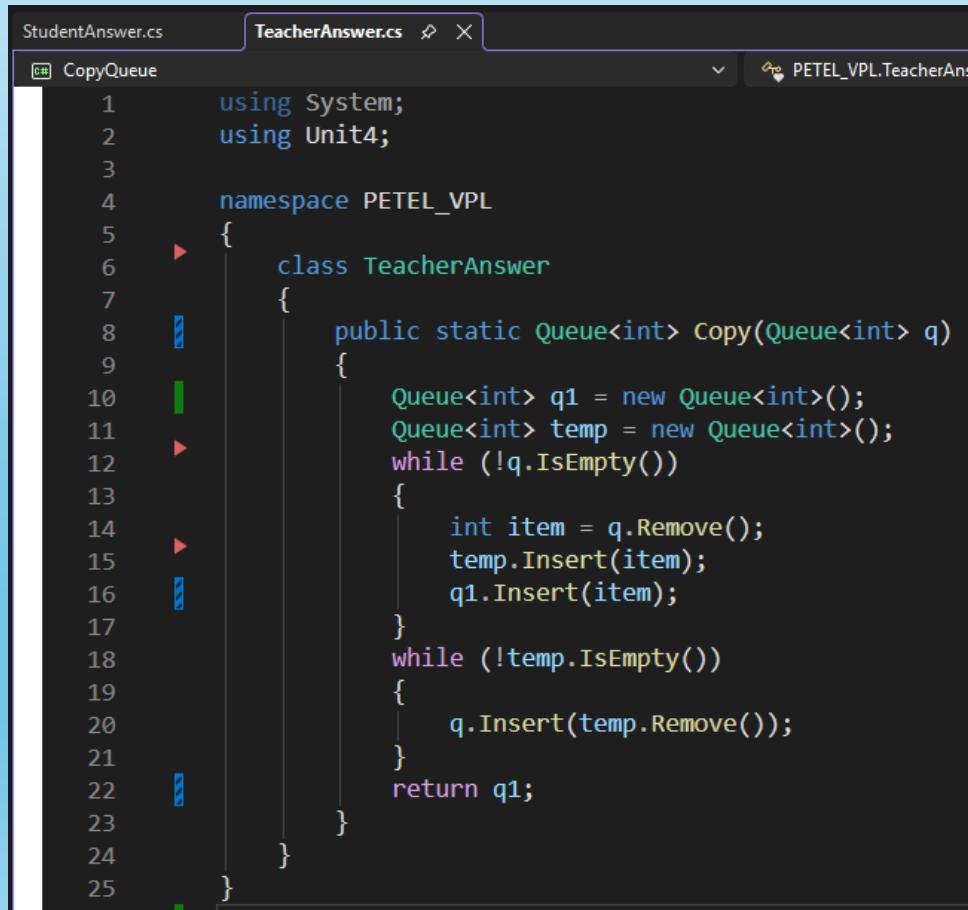
בדיקות התשובה של המורה

Unit4Helper

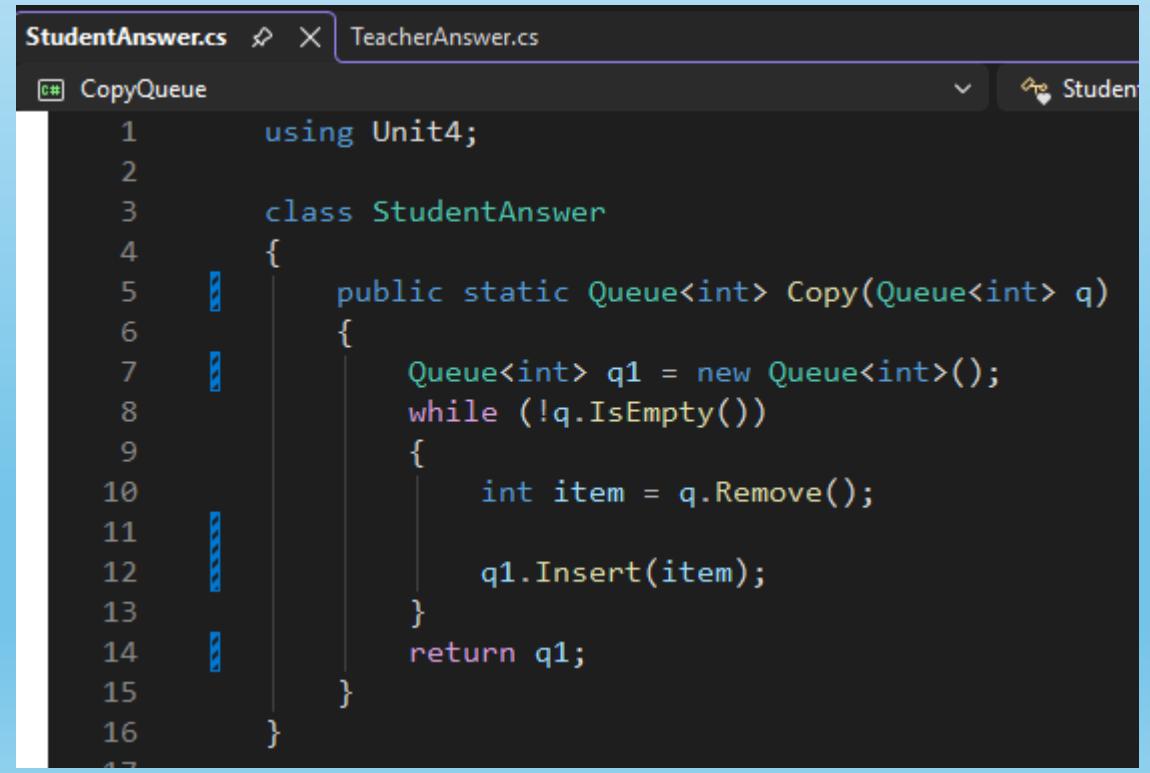


פתרון המורה והתלמיד

- נבנה את פתרון המורה ואת פתרון התלמיד לתרגיל של העתקת תור.
- פתרון התלמיד כולל שגיאה בכר שלא שומר על התור המקורי.



```
StudentAnswer.cs TeacherAnswer.cs
CopyQueue
1 using System;
2 using Unit4;
3
4 namespace PETEL_VPL
5 {
6     class TeacherAnswer
7     {
8         public static Queue<int> Copy(Queue<int> q)
9         {
10            Queue<int> q1 = new Queue<int>();
11            Queue<int> temp = new Queue<int>();
12            while (!q.IsEmpty())
13            {
14                int item = q.Remove();
15                temp.Insert(item);
16                q1.Insert(item);
17            }
18            while (!temp.IsEmpty())
19            {
20                q.Insert(temp.Remove());
21            }
22            return q1;
23        }
24    }
25 }
```



```
StudentAnswer.cs TeacherAnswer.cs
CopyQueue
1 using Unit4;
2
3 class StudentAnswer
4 {
5     public static Queue<int> Copy(Queue<int> q)
6     {
7         Queue<int> q1 = new Queue<int>();
8         while (!q.IsEmpty())
9         {
10            int item = q.Remove();
11            q1.Insert(item);
12        }
13        while (!temp.IsEmpty())
14        {
15            q1.Insert(temp.Remove());
16        }
17    }
18 }
```

הרצה הקוד לבדיקה – Unit4Helper

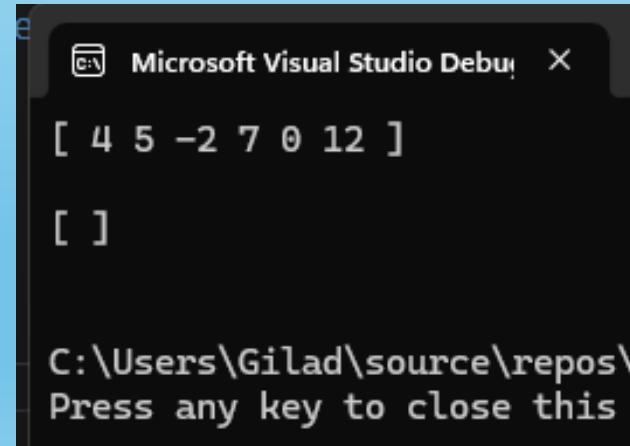
- נגידר את `Program.cs` כנקודת הכניסה באמצעות המאפיינים שלuproject.

```
using System;
using Unit4;

namespace CopyQueue
{
    internal class Program
    {
        static void Main(string[] args)
        {
            RunStudentCode();
        }

        public static void RunStudentCode()
        {
            Queue<int> q1 = Unit4Helper.BuildQueue(
                new int[] { 4, 5, -2, 7, 0, 12 });
            Queue<int> q2 = StudentAnswer.Copy(q1);
            Console.WriteLine(q2);
            Console.WriteLine(q1);
        }
    }
}
```

- נעדכן את `using` בהתאם לתמונה.
- ניצור תור לדוגמה.
- נקרא לפעולה של התלמיד.
- נדפס תוצאות



פועלות ב Unit4Helper

```
public static void NodeList()
{
    Node<int> lst = Unit4Helper.BuildNodeList(new int[] { 4, -2, 7, 0, -1, 0, 0 });
    Unit4Helper.PrintList(lst);
    int[] arr = Unit4Helper.NodeListToArray(lst);
    string str = Unit4Helper.NodeListToString(lst);
    Console.WriteLine(str);
}
```

- רשימת חוליות

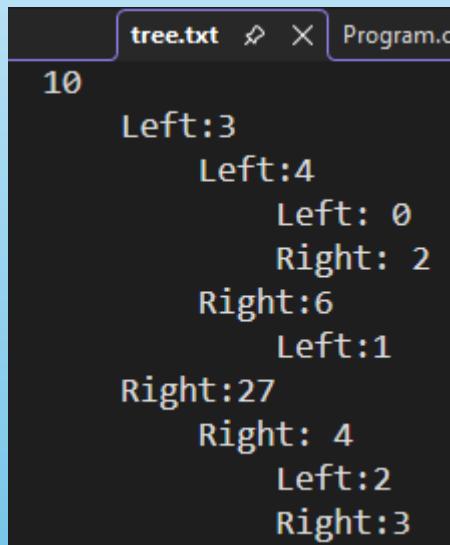
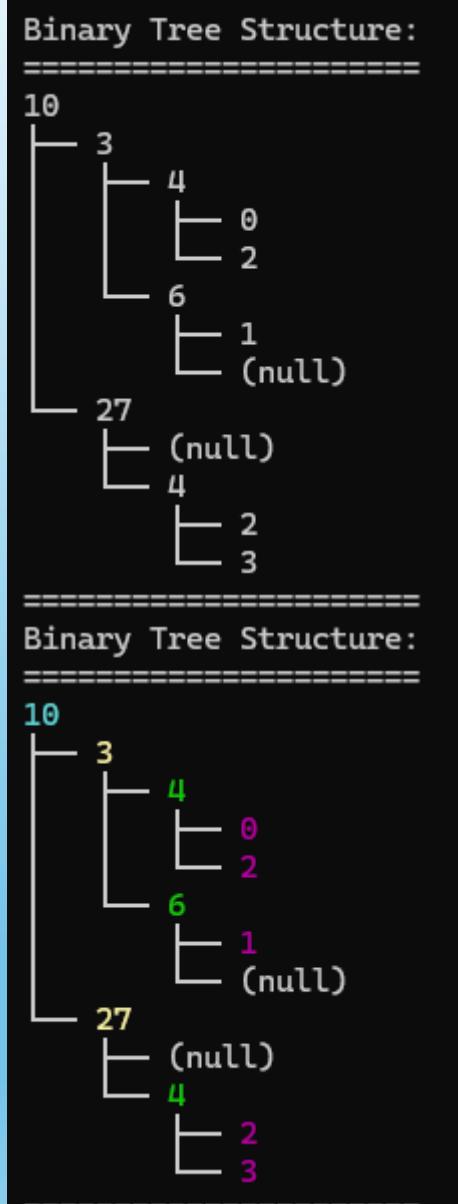
```
public static void QueueMethods()
{
    Queue<int> q = Unit4Helper.BuildQueue(new int[] { 4, -2, 7, 0, -1, 0, 0 });
    Console.WriteLine(q);
    int[] arr = Unit4Helper.QueueToArray(q);
}
```

- תור

```
public static void StackMethods()
{
    Stack<int> s = Unit4Helper.BuildStack(new int[] { 4, -2, 7, 0, -1, 0, 0 });
    Console.WriteLine(s);
    int[] arr = Unit4Helper.StackToArray(s);
}
```

- מחסנית

פעולות עברור עצים



- הוספנו פעולה המאפשרת לבנות עצים מקובץ טקסט.
- אנחנו בונים קובץ טקסט בצורה הבאה. יש לעשות שימוש בטאב לירידה בrama.
- פעולה קוראת את העץ מהקובץ ויוצרת אובייקט של עץ בינהי.
- הוספנו הדפסה של העצים.

```
public static void BinTreeMethods()  
{  
    string path = Unit4Helper.GetTreeFilePath("tree.txt");  
    BinNode<int> tree = Unit4Helper.BuildBinaryTree<int>(path);  
    Unit4Helper.PrintBinaryTree(tree);  
    Unit4Helper.PrintBinaryTreeColored(tree);  
    string str = Unit4Helper.BinaryTreeToString(tree);  
    Console.WriteLine(str);  
}
```

בניהית Case Test



MainTester

- לאחר שבנו את תשובה המורה ותשובה התלמיד נוכל לעבור לבניית הבדיקה עצמה.
- יש להגדיר את MainTester כקודת הכניסה של הפרויקט (במקום Program).
- המחלקה בנויה משלוש פעולות סטטיות:

```
class MainTester
{
    public static void Main(string[] args)...
    private static void CaseTester(VPLTester tester)...
    private static void CodeTester(VPLTester tester)...
}
```

- Main – הפעולה הראשית
- CaseTester – בדיקת הרצת הקוד.
- CodeTester - בדיקה תחביר התשובה של התלמיד.

Main

```
public static void Main(string[] args)
{
    // Initialize the tester
    var tester = new VPLTester(
        studentFile: "StudentAnswer.cs",
        studentNamespace: "",
        studentClassName: "StudentAnswer",
        studentMethodName: "Copy",
        teacherNamespace: "PETEL_VPL",
        teacherClassName: "TeacherAnswer",
        teacherMethodName: "Copy",
        showDetails: true
    );

    // Run all test suites
    CaseTester(tester);
    CodeTester(tester);

    // Display results (VPL parses this output)
    Console.WriteLine("\n" + tester.FormatResponse());
    Console.WriteLine($"Grade :=> {tester.GetGrade()}");
}
```

- **בשלב ראשון עלינו להגדיר אובייקט של הטעстр:**
 - VPL – שם הקובץ ב L
 - StudentNamespace – אם לא הוגדר יש להשאיר ריק.
 - StudentMethodName – שם הפעולה הנבדקת.
 - teacherNamespace - בהתאם לפתרון המורה (מומלץ לא לשנות).
 - teacherMethodNake – שם הפעולה הנבדקת בקוד המורה.
 - showDetails – האם לתת תשובה מפורטת לתלמיד.
- **פרט לכך, אין צורך לעשות שינויים.**

CaseTester

- דוגמה לבדיקה של הפעולה העתקת קוד. הבדיקה נכשלה כיון שהתלמיד שינה את התור המקורי.

```
private static void CaseTester(VPLTester tester)
{
    Queue<int> q1 = Unit4Helper.BuildQueue(new int[] { 3, 5, -9, 3, 5, 5, 2, 1, 2 });
    tester.TestMethod(
        testName: "Test 1: check the correct return",
        points: 10,
        parameters: new object[] { q1 }
    );
}
```

```
Comment :=>>Test 1: check the correct return | params: p0=[ 3 5 -9 3 5 5 2 1 2 ]: failure. 0 points
<|--  
Return value check:  
Expected: Queue<[3, 5, -9, 3, 5, 5, 2, 1, 2]>  
Actual:   Queue<[3, 5, -9, 3, 5, 5, 2, 1, 2]>  
Explanation: Returned value matches the expected result.  
  
Input parameter state after call does not match the requirement:  
p0: expected=Queue<[3, 5, -9, 3, 5, 5, 2, 1, 2]> | actual=Queue<[]>
--|>  
  
Grade :=>> 0
```

השלבים לבניית בדיקה

- בניית הפרמטרים (הדוגמאות), כגון רשיימה, תור וכו'.
- קריאה לפעולות הבדיקה `TestMethod`:
 - `testName` – שם הבדיקה ניתן לכלול פירוט לתלמיד.
 - `points` – מספר הנקודות שהבדיקה שווה.
- `Parameters` - רשיימה של אובייקטים הכולל את הפרמטרים המועברים לפעולה.
- ניתן לבנות מספר בדיקות בזו אחר זו.

```
private static void CaseTester(VPLTester tester)
{
    Queue<int> q1 = Unit4Helper.BuildQueue(new int[] { 3, 5, -9, 3, 5, 5, 2, 1, 2 });
    tester.TestMethod(
        testName: "Test 1: check the correct return",
        points: 10,
        parameters: new object[] { q1 }
    );
}
```

הפרמטרים של TestMethod

פרמטר	סוג	הסבר
testName	String	* שם הבדיקה והסביר קצר לתלמיד
Points	Int	* מספר הנקודות שיקבל התלמיד אם עבר את הבדיקה
Parameters	Object []	מערך אובייקטים הכלול את הפרמטרים שיש להעביר לפעולה.
consoleInput	String , string [], List<string>	הקלט במקורה והפעולה דורשת read. הקלט יכול להיות כל אחד מהסוגים המצוינים. במקרה של לולאת קלט ניתן לתת מערך של מחרוזות או רשימה של מחרוזות.
captureConsoleOutput	Bool	אם להשוות הפלט לקונסול במקרה והפעולה כוללת write ?
compareParams	Bool (default: true)	אם הפעולה שינתה את הפרמטרים (משווה את הפרמטרים לאחר הפעולה) ?
exceptionComments	Dictionary<Type , string>	אופציונלי: אפשרות לשנות את הודעה השגיאה במקרה של Exception אצל התלמיד.
compareReturn	Bool (default: true)	אופציונלי: האם לבדוק את הערך המוחזר מהפעולה?
* פרמטר חובה. השאר אופציונלי		

לוגמאות לבדיקות

```
private static void CaseTester(VPLTester tester)
{
    tester.TestMethod(
        testName: "Test 1: 3 numbers. capture Console Output",
        points: 10,
        parameters: new object[] { 3 },
        compareParams: false,
        consoleInput: new string[] { "5", "3", "7" },
        captureConsoleOutput: true
    );

    tester.TestMethod(
        testName: "Test 2: 3 numbers. Only Compare Return",
        points: 10,
        parameters: new object[] { 3 },
        compareParams: false,
        consoleInput: new string[] { "5", "3", "4" },
        captureConsoleOutput: false
    );

    tester.TestMethod(
        testName: "Test 3: 1 numbers",
        points: 10,
        parameters: new object[] { 1 },
        compareParams: false,
        consoleInput: "3",
        captureConsoleOutput: true,
        compareReturn: false
    );
}
```

- תרגיל: כתוב פעולה מקבלת מספר שלם ונה. הפעולה תקלוט מחרירים של מוצרים כמספר ה נתן. הפעולה תדפס את סכום המוצרים ותחזיר את הממוצע שלהם.

```
Comment :=>Test 1: 3 numbers. capture Console Output | params: p0=3 | input: "5\n3\n7\n": success. 10 points
<|--  
output: "Enter price\nEnter price\nEnter price\nThe Sum is: 15\n"  
return: 5  
--|>

Comment :=>Test 2: 3 numbers. Only Compare Return | params: p0=3 | input: "5\n3\n4\n": success. 10 points
<|--  
return: 4  
--|>

Comment :=>Test 3: 1 numbers | params: p0=1 | input: "3\n": success. 10 points
<|--  
output: "Enter price\nThe Sum is: 3\n"  
--|>

Grade :=> 30
```

דוגמאות לבדיקה

```
private static void CaseTester(VPLTester tester)
{
    Queue<int> q1 = Unit4Helper.BuildQueue(new int[] { 3, 5, -9, 3, 5, 5, 2, 1, 2 });
    tester.TestMethod(
        testName: "Test 1: check the correct return. Don't check the original Queue",
        points: 10,
        parameters: new object[] { q1 },
        compareParams: false
    );

    Queue<int> q2 = Unit4Helper.BuildQueue(new int[] { 3, 5, -9, 3, 5, 5, 2, 1, 2 });
    tester.TestMethod(
        testName: "Test 2: check only the original Queue if it changed",
        points: 10,
        parameters: new object[] { q2 },
        compareParams: true,
        compareReturn: false
    );

    q2 = Unit4Helper.BuildQueue(new int[] { 3, 5, -9, 3, 5, 5, 2, 1, 2 });
    tester.TestMethod(
        testName: "Test 2: check both return and original",
        points: 10,
        parameters: new object[] { q2 },
        compareParams: true,
        compareReturn: true
    );
}
```

- כתבו פעולה המעתיקת תור מבלי לפגוע בתור המקורי.
- בדיקה ראשונה – בודק רק את התור המוחזר ולא בודק אם התור המקורי השתנה.
- בדיקה שנייה – בודק רק אם התור המקורי השתנה.
- בדיקה שלישית – בודק את שניהם

הוספת הודעה שגיאת מותאמת אישית

- באפשרותנו להוסיף להודעות השגיאה של המערכת הודעות שגיאת מותאמת אישית.
- לדוגמה הודעה השגיאה:

```
Comment :=>>Test 1: check the correct return. Don't check the original Queue | params: p0=[ 3 5 -9 3 5 5 2 1 2 ]: failure. 0 points
<|--  
Error during test execution: Object reference not set to an instance of an object.  
--|>
```

- לאחר השינוי הוספה הודעה הבאה:

```
Comment :=>>Test 1: check the correct return. Don't check the original Queue | params: p0=[ 3 5 -9 3 5 5 2 1 2 ]: failure. 0 points
<|--  
Error during test execution: Object reference not set to an instance of an object.  
  
Teacher note: You tried to remove from empty queue  
--|>
```

הוספת הודעת שגיאה מותאמת אישית

- עלינו להגדיר מילון בו המפתח הוא סוג השגיאה והערך הודעת השגיאה.
- בפרמטרים של הבדיקה נוסיף את המילון ב exceptionComments .

```
private static void CaseTester(VPLTester tester)
{
    // Optional: Custom exception message
    var commonExceptionComments = new C.Dictionary<Type, string>
    {
        { typeof(NullReferenceException), "You tried to remove from empty queue" },
    };

    Queue<int> q1 = Unit4Helper.BuildQueue(new int[] { 3, 5, -9, 3, 5, 5, 2, 1, 2 });
    tester.TestMethod(
        testName: "Test 1: check the correct return. Don't check the original Queue",
        points: 10,
        parameters: new object[] { q1 },
        compareParams: false,
        exceptionComments: commonExceptionComments
    );
}
```

בניהת Code Test



בדיקות תחביר הקוד

- במערכת הבדיקות ניתן לבצע בנוסף בדיקה תחבירית של הקוד שכתב התלמיד כגון: האם הפעולה רקורסיבית, האם הפעולה סטטית, האם הפרמטרים של הפעולה תקינים, האם הערך המוחזר תקין ועוד.
- דוגמה – בדיקה האם התלמיד השתמש בלולאות מוכננות:

```
private static void CodeTester(VPLTester tester)
{
    // Initialize code analyzer (handles errors internally)
    tester.InitializeCodeAnalyzer();

    // Test 11: Verify no nested loops (ensures O(n) not O(n2))
    tester.TestCodeStructure(
        testName: "Test 11: No nested loops (O(n) complexity)",
        points: 10,
        checkType: CodestructureCheck.HasNestedLoops,
        shouldPass: false, // We want HasNestedLoops to return FALSE
        failureMessage: "Method must not have nested loops to maintain O(n) complexity"
    );
}
```

בנייה בדיקת תחביר

- בתחילת הפעולה CodeAnalyzer علينا לאתחל את .CodeAnalyzer

```
private static void CodeTester(VPLTester tester)
{
    // Initialize code analyzer (handles errors internally)
    tester.InitializeCodeAnalyzer();

    // Test 11: Verify no nested loops (ensures O(n) not O(n2))
    tester.TestCodeStructure(
        testName: "Test 11: No nested loops (O(n) complexity)",
        points: 10,
        checkType: CodeStructureCheck.HasNestedLoops,
        shouldPass: false, // We want HasNestedLoops to return
        failureMessage: "Method must not have nested loops to ma
    );
}
```

- נוסיף את שם הבדיקה והסביר.
- מספר הנקודות.
- סוג הבדיקה לפי רשימה קבועה מראש (ראו במחשך).
- פרמטרים נוספים אופציונליים: כגון הודעת שגיאה.

פרמטרים של TestCodeStructure

פרמטר	סוג	פירוט
testName	String	שם הבדיקה והסביר קצר שלה – הtekסט יופיע לתלמיד.
points	Int	מספר הנקודות שיקבל התלמיד אם עבר את הבדיקה.
checkType	CodeStructureCheck	סוג הבדיקה לפי רשימה קבועה מראש (ראו בהמשך).
shouldPass	Bool (default=true)	אופציונלי: האם תוצאות הבדיקה המוצופה היא חיובית או שלילית.
expectedCount	Int ? (default=null)	אופציונלי: בבדיקות כמותיות המספר המוצופה (לדוגמא: מספר הלולאות).
failureMessage	String (default= null)	אופציונלי: הודעה שגיאה לתלמיד אם הבדיקה נכשלה.

סוגי הבדיקות – המהלך - CodeStructureCheck

שם הבדיקה	ערך מוחזר	תיאור
IsRecursive	bool	האם קיימת קריאה רקורסיבית
CountForLoop	Int	מספר לולאות for בהם השתמש התלמיד (לאו דווקא מכוננות).
CountWhileLoop	Int	מספר לולאות while בהם השתמש התלמיד (לאו דווקא מכוננות).
CountForEachLoop	Int	מספר לולאות forEach בהם השתמש התלמיד (לאו דווקא מכוננות).
CountAnyLoop	Int	מספר לולאות מכל סוג בהם השתמש התלמיד (לאו דווקא מכוננות).
CountIfStatements	Int	מספר התנאים בהם השתמש התלמיד
CountRecursiveCalls	Int	מספר הקריאה הרקורסיביות בפעולה.
CountReturnStatements	Int	מספר פקודות ה return בפעולה.
CountNewNodes	Int	מספר הפקודות <new Node<T> new בפעולה
CountNewQueue	Int	מספר פקודות <new Queue<T> new בפעולה
CountNewStack	Int	מספר פקודות <new Stack<T> new בפעולה
CountNewBinNode	Int	מספר פקודות <new BinNode<T> new בפעולה

סוגי הבדיקות – המשך

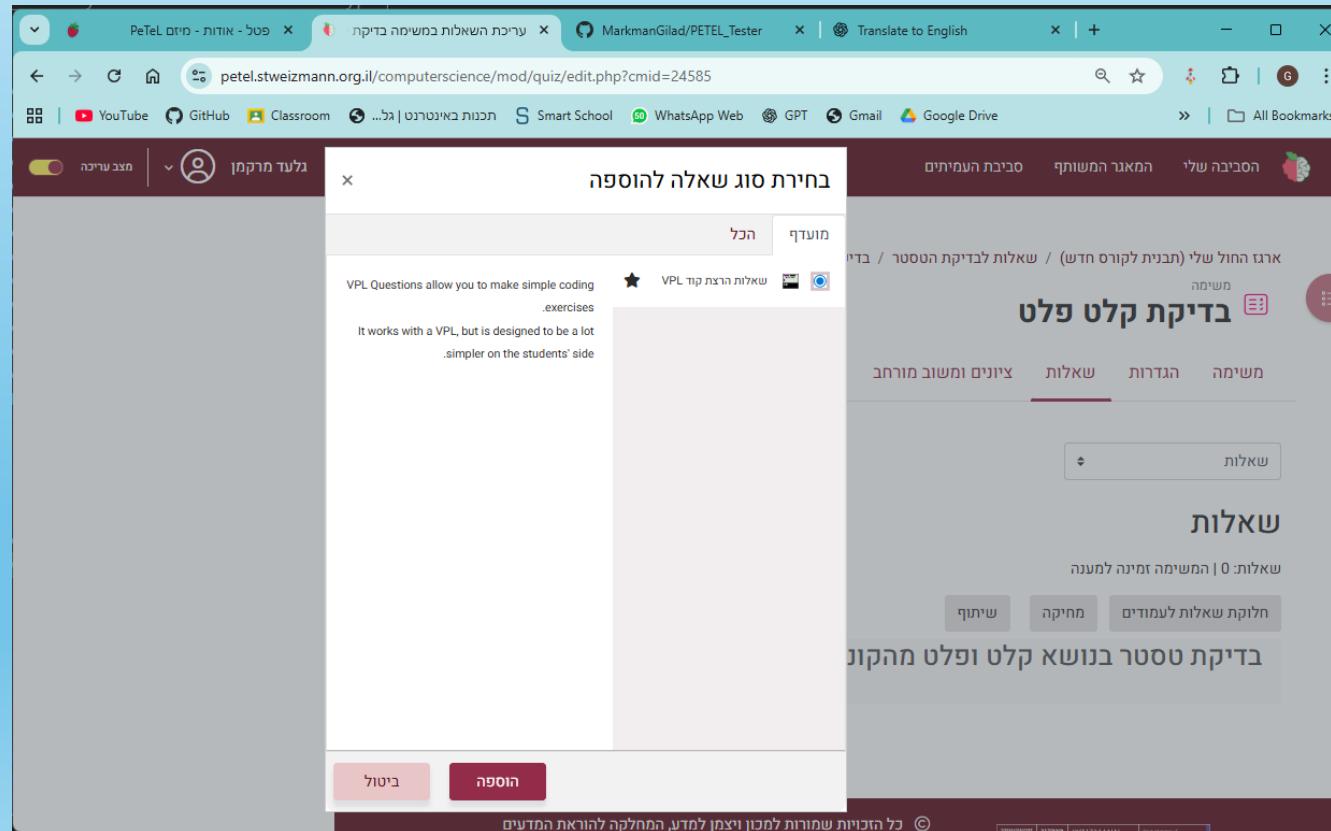
שם הבדיקה	ערך מוחזר	תיאור
CountSetNext	Int	מספר פקודות SetNext של המחלקה <code><T>Node</code>
CountGetNext	Int	מספר פקודות GetNext של המחלקה <code><T>Node</code>
HasNestedLoops	Bool	אם התלמיד השתמש בולולאות מכוננות מכל סוג (לצורך סיבוכיות).
IsStatic	Bool	בדיקה חתימת הפעולה
IsPublic	Bool	בדיקה חתימת הפעולה
IsPrivate	Bool	בדיקה חתימת הפעולה
IsProtected	Bool	בדיקה חתימת הפעולה
IsInternal	Bool	בדיקה חתימת הפעולה
CheckParams	Bool	בדיקה חתימת הפעולה – השוואת הפרמטרים לחתימת פעולות המורה.
CheckReturnType	Bool	בדיקה חתימת הפעולה – השוואת הערך המוחזרת לחתימת פעולות המורה.

העלאת בדיקה ל PETEL



העלאת בדיקה למערכת Moodle PeTel

- לאחר שהרכינו את הבדיקה בסביבת העבודה וניסינו תשובה שונה של התלמידים הגיע הזמן להעלות למערכת PETEL.
- ניצור שאלת הרצת קוד VPL.



בנייהה של שאלת

תבנית תשובה

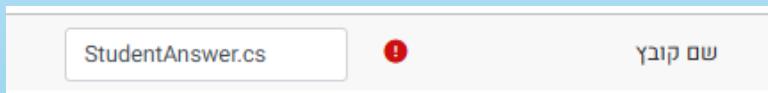
דוגמת קוד נכונה של המורה

?

דוגמת קוד
נכונה של
המורה

```
1 using System;
2
3 class TeacherAnswer
4 {
5     public static double InputOutput(int num)
6     {
7         double sum = 0;
8         for (int i = 0; i < num; i++)
9         {
10             Console.WriteLine("Enter price");
11             double price = double.Parse(Console.ReadLine());
12             sum += price;
13         }
14         Console.WriteLine("The Sum is: " + sum);
15         return sum / num;
16     }
17 }
18 }
```

- לבנות שאלת כרגע
 - שם קובץ: מומלץ StudentAnswer.cs
אחרת יש לשנות את הקובץ ב MainTester.

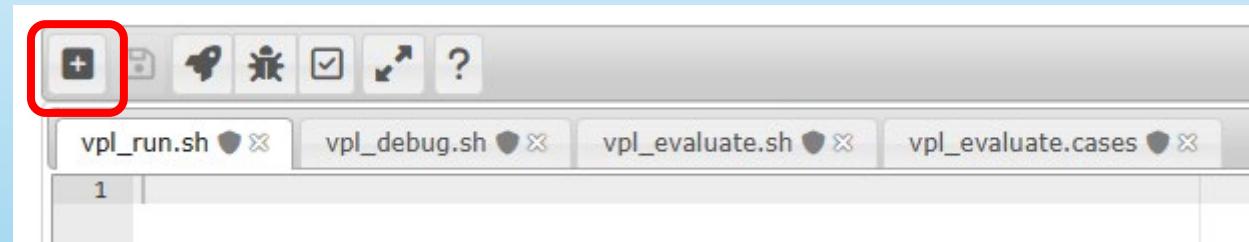


- בתבנית התשובה מומלץ להעתיק את תשובה התלמיד מסביבת העבודה ולמחוק את הפרטים אותם יתבקש התלמיד לכתוב.

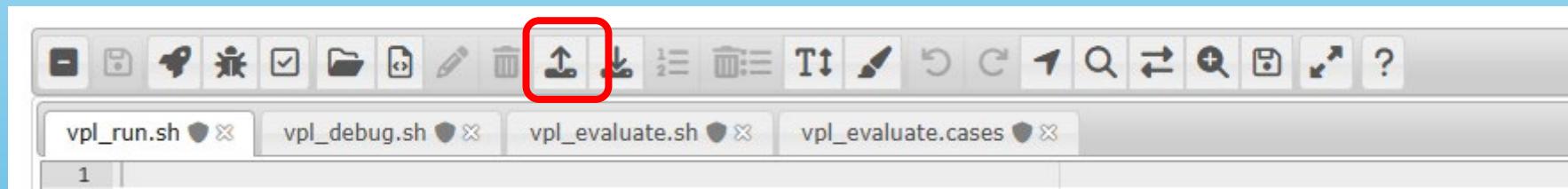
- חשוב – לא לשכוח using.
 - מומלץ - לא להוסיף nameSpace (אחרת יש לעדכן בטעטר)

העלאת קבצי הבדיקה

- בחלון קבצי הרצה ובדיקה קוד יש ללחוץ על +



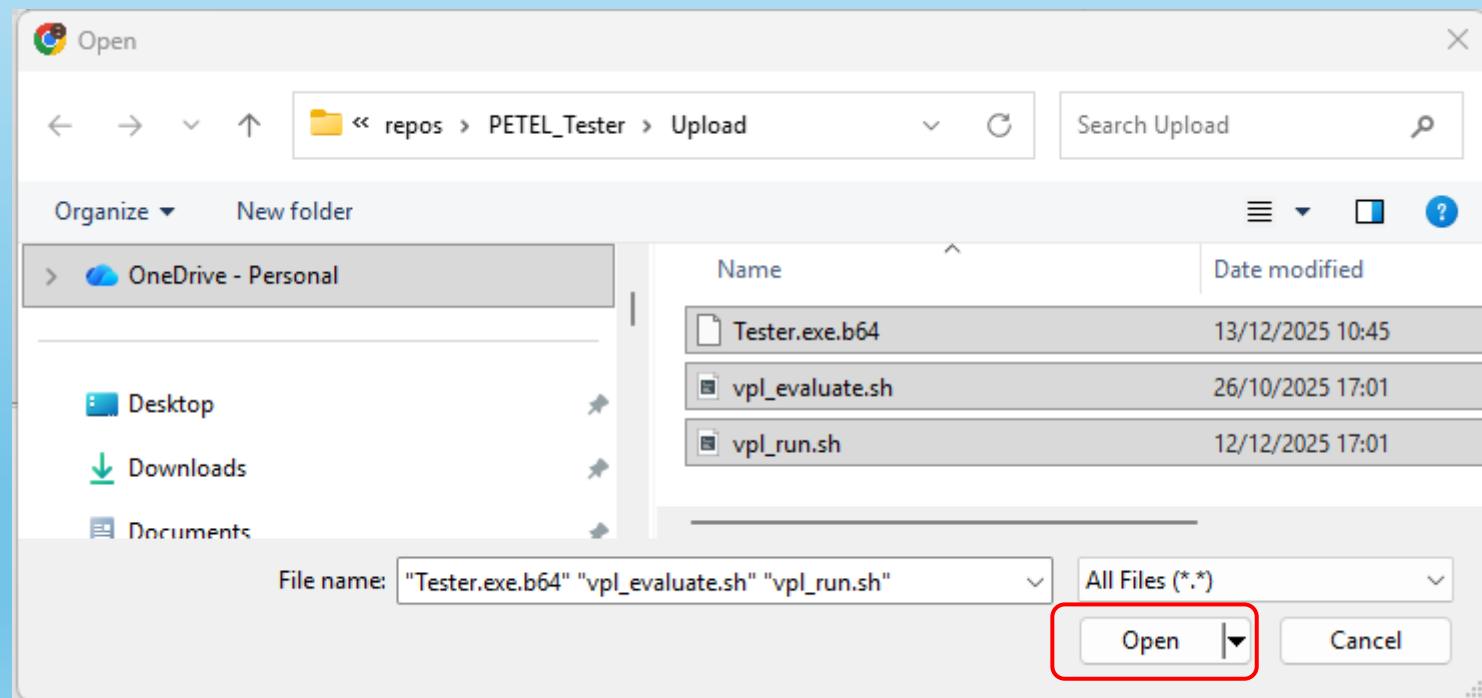
- ולאחר מכן, ללחוץ על העלה (חץ למעלה).



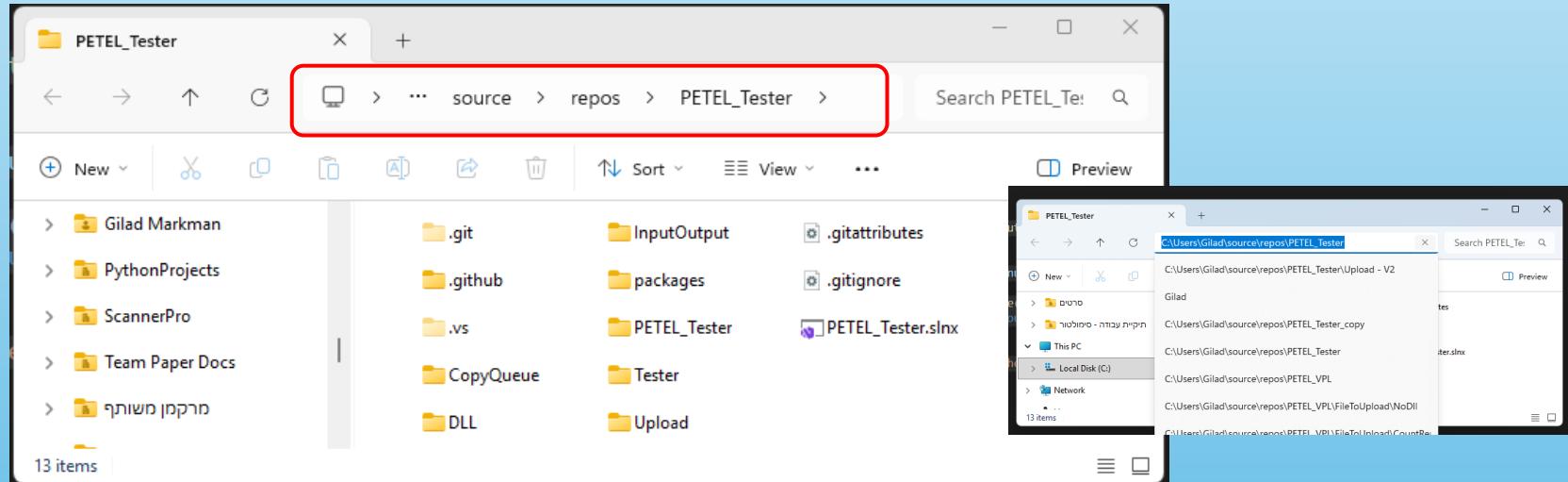
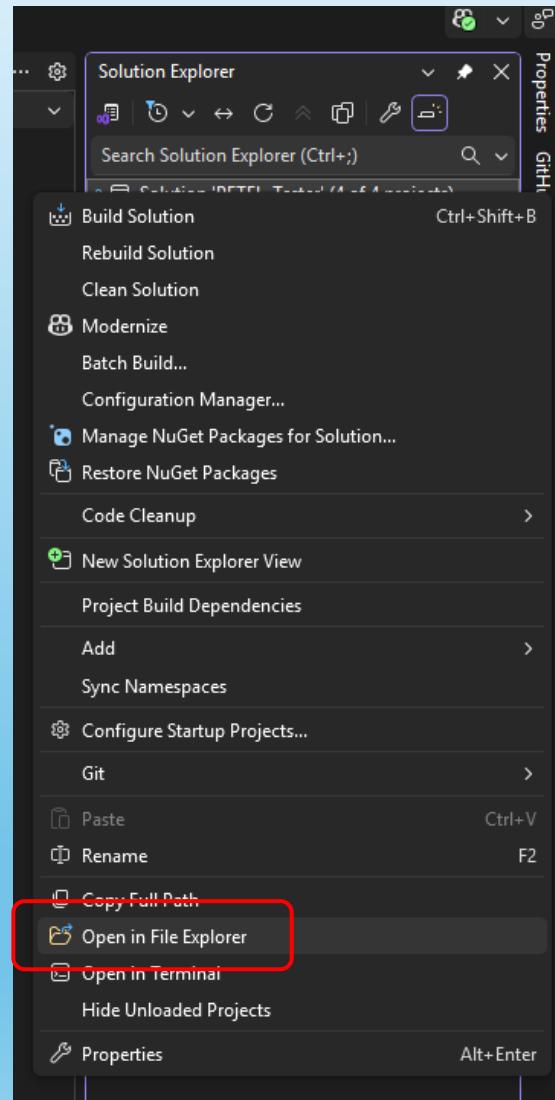
- לאחר הלחיצה יפתח חלון במחשב שלכם.

העלאת קבצי הבדיקה הכלליים

- יש לאתר את הספרייה Upload בתחום הפרויקט שלכם, לסמן את שלושת הקבצים בספריה, ולהעלות אותם (עזרה באיתור הספריה בשקופית הבאה).
- קבצים אלו קבועים בכל המשימות.



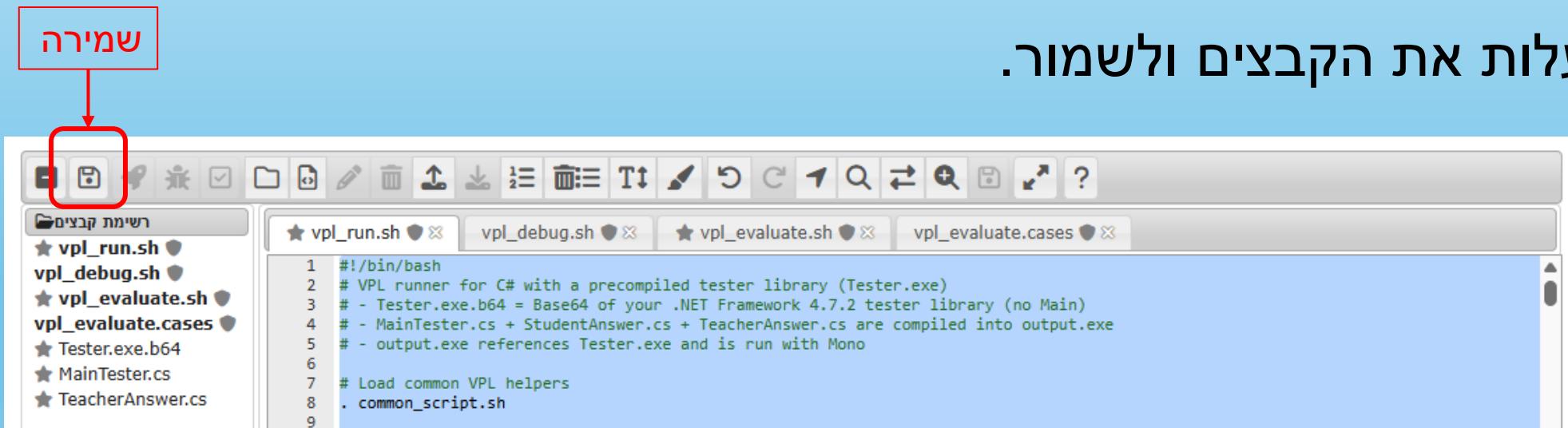
איתור ספריית העבודה במחשב שלנו



- ניתן להעתיק את הנתיב של הפרויקט והתmesh בו בחילון העלאה של ה PETEL.
- על מנת לאתר את ספריית העבודה במחשב שלנו נלחץ קлик ימני על ה solution שלנו.
- בחירה ב Open in File Explorer .

העלאת הקבצים של המשימה

- בשלב הבא נעה את הקבצים של המשימה הספציפית.
- ללחוץ שוב על החץ לעלה, לאתר את הספריה שלuproject, לסמן רק את שני הקבצים:
 - MainTester.cs •
 - TeacherAnswer.cs •
- לעלות את הקבצים ולשמור.

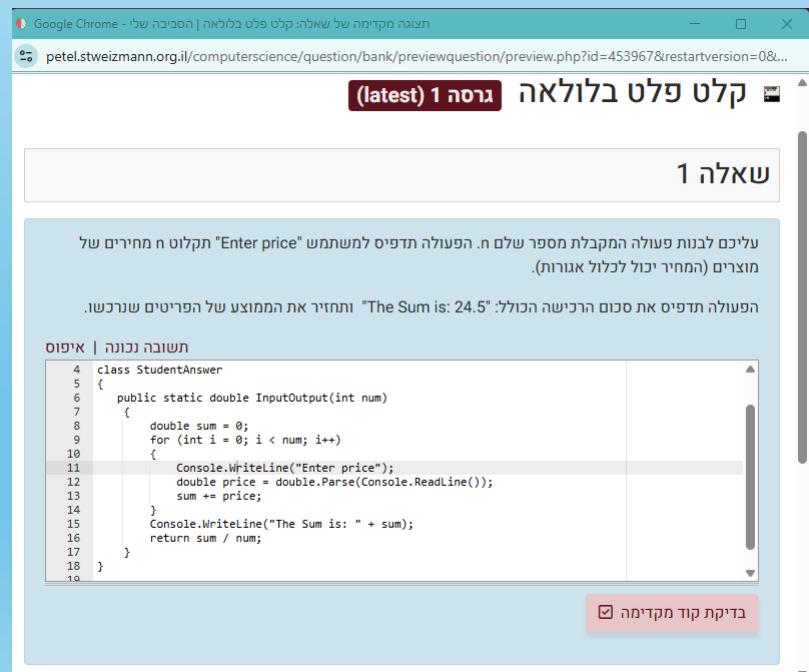


בדיקה

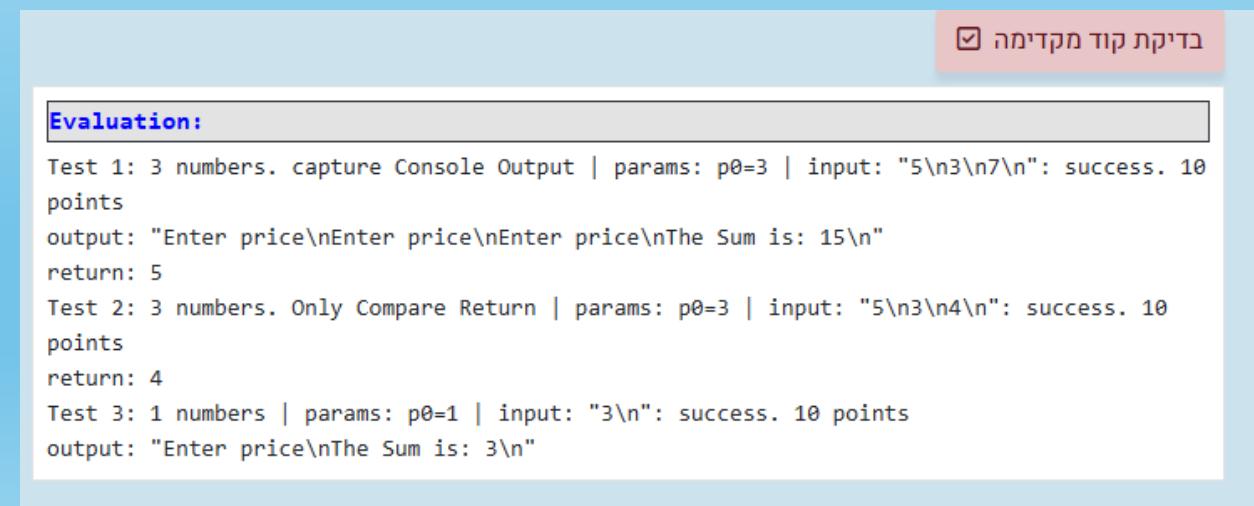
- בחלק התיכון של הטופס נלחץ על שמירה והמשר עריכה ולאחר מכן על **תצוגה מקדימה**.



- נעתיק את תשובה התלמיד מסביבת העבודה ל PETEL ונלחץ "בדיקה קוד מקדימה".



- נודא שקיבלנו תוצאה צפויה.





מכון
ויצמן
למדעים

WEIZMANN
INSTITUTE
OF SCIENCE



עבודה בעימה

