**LINQ FUNKCIJE**

LINQ služi nam za upit podataka. Postoje različite vrste podataka s kojima možemo koristiti LINQ. To su: zbirke, nizovi, XML datoteke i baze podataka. Zapravo, LINQ nam može poslužiti za rad s gotovo svim vrstama podataka.

1. **LINQ metode i svojstva**

Promatrat ćemo sljedeće metode i svojstva:

SUM

MIN

MAX

AVERAGE

CONTAINS

ELEMENT AT

FIRST

LAST

DISTINCT

COUNT

TAKE

INTERSECT

Projekt smo započeli izradom forme koja se sastoji od jednog gumba i listBox-a.

Dvostrukim klikom na gumb otvorio nam se sadržaj koda. U dio koda koji se odnosi na gumb dodali smo niz cijelih brojeva.

**int[] numberList = new int[5] {1, 2, 3, 4, 5};**

Zatim smo u listBox dodali sljedeće informacije:

**listBox1.Items.Add("ARRAY ITEMS: 1, 2, 3, 4, 5"); listBox1.Items.Add("========================");**

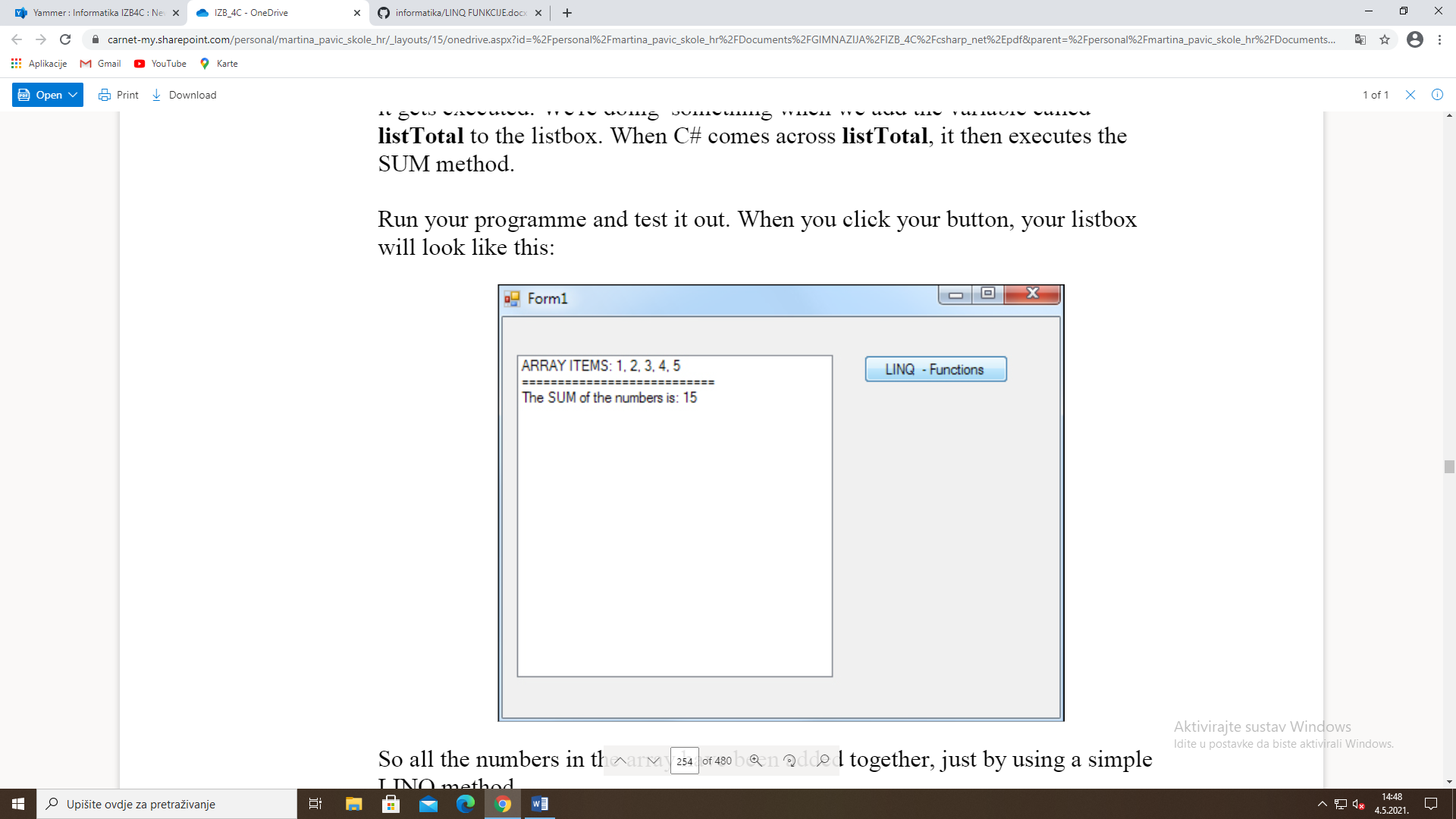
1. **SUM metoda**

LINQ za zbrajanje koristi metodu SUM. Da bi smo objasnili princip rada metode, stvorili smo varijablu tipa int i dali joj naziv listTotal.

**int listTotal = numberList.Sum();**

Varijabla je tipa int, što znači da će i metoda sum vratiti vrijednost u obliku int varijable. Također smo u ovom primjeru upotrijebili i svojstvo Add.

**listBox1.Items.Add("The SUM of the numbers is: " + listTotal);**



1. **Minimum i maximum**

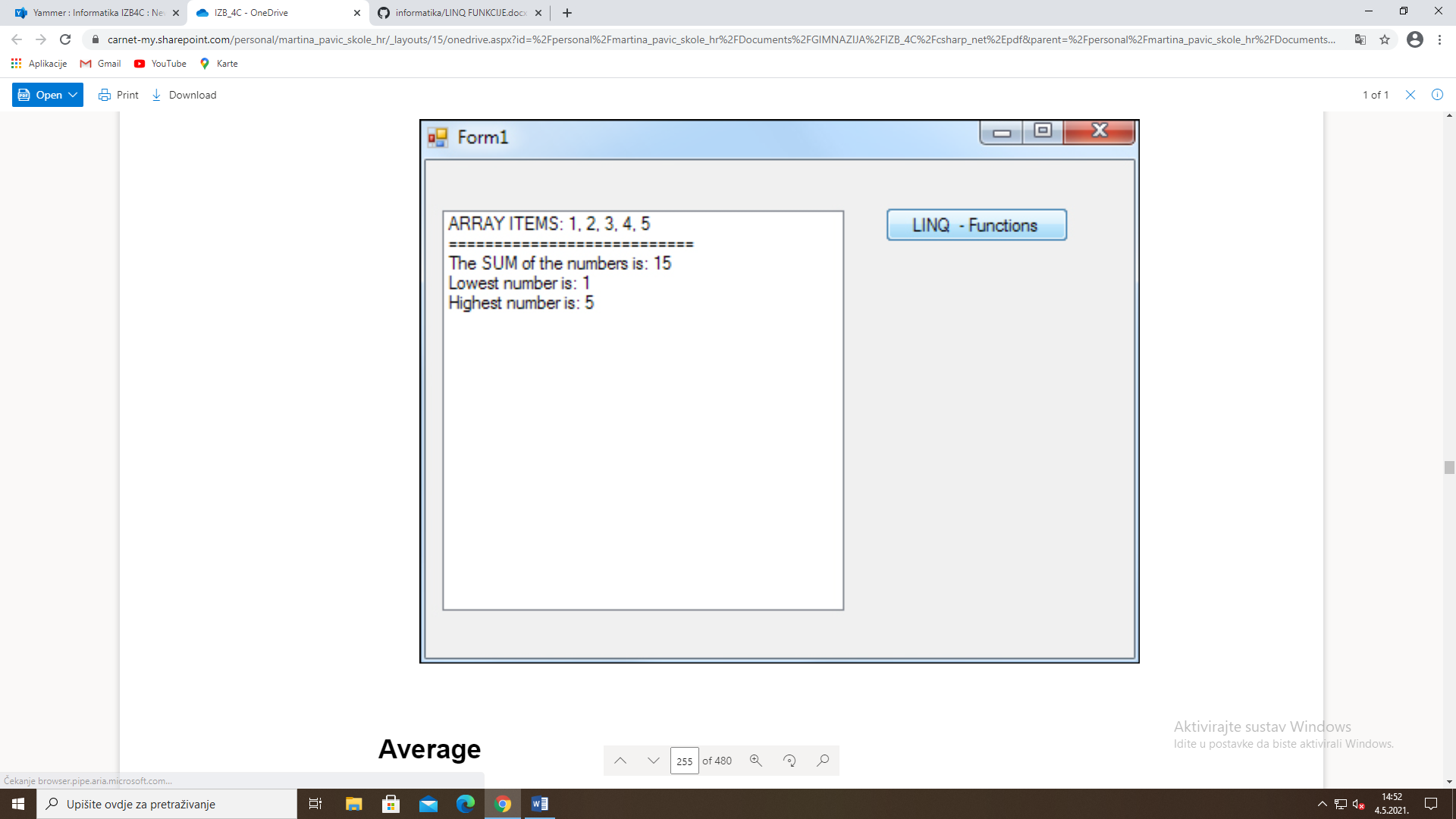
Ove nam dvije metode omogućuju pronalazak najmanjeg, odnosno najvećeg elementa niza.

**int lowestNumber = numberList.Min();**

**listBox1.Items.Add("Lowest number is: " + lowestNumber);**

**int highestNumber = numberList.Max();**

**listBox1.Items.Add("Highest number is: " + highestNumber);**



1. **Average**

Metoda Average omogućuje nam izračun prosjeka elemenata nekoga niza. Upotreba ove metoda vrlo je jednostavna:

**double averageValue = numberList.Average();**

**listBox1.Items.Add("Average of all values is: " + averageValue);**

1. **Contains**

Ova nam metoda omogućuje provjeru postojanja određenog elementa u nizu. Upotreba i ove metode vrlo je jednostavna. Ono što je važno jest deklaracija varijable tipa bool. Ova varijabla ima dvije mogućnosti: biti pozitivna (1) ili biti negativna (0).

**bool doesContain = numberList.Contains(3);**

**listBox1.Items.Add("Contains the number 3: " + doesContain);**

1. **Element At**

Ova nam metoda omogućuje da saznamo koji se element nalazi na određenoj poziciji. Deklarirali smo varijablu tipa int i ona će nam otkriti element niza čija je pozicija broj jedan:

**int elementValue = numberList.ElementAt(1);**

**listBox1.Items.Add("The element at array position 1 is: " + elementValue);**

1. **First & Last**

Ove nam dvije metode omogućuju pronalaženje prvog, odnosno posljednjeg elementa niza.

**int firstElement = numberList.First();**

**listBox1.Items.Add("First Array Value = " + firstElement);**

**int lastElement = numberList.Last();**

**listBox1.Items.Add("Last Array Value = " + lastElement);**

1. **Distinct**

Ova nam metoda omogućuje da elemente niza koji su duplicirani, uklonimo. Dakle, metoda zapravo u nizu ostavlja samo po jedan primjerak određenoga elementa.

**listBox1.Items.Add("==================================");**

**listBox1.Items.Add("ARRAY ITEMS: 1, 1, 2, 2, 3, 4, 5, 5"); listBox1.Items.Add("REMOVE DUPLICATES FROM ARRAY"); listBox1.Items.Add("==================================");**

**int[] aryNums = new int[8] { 1, 1, 2, 2, 3, 4, 5, 5 };**

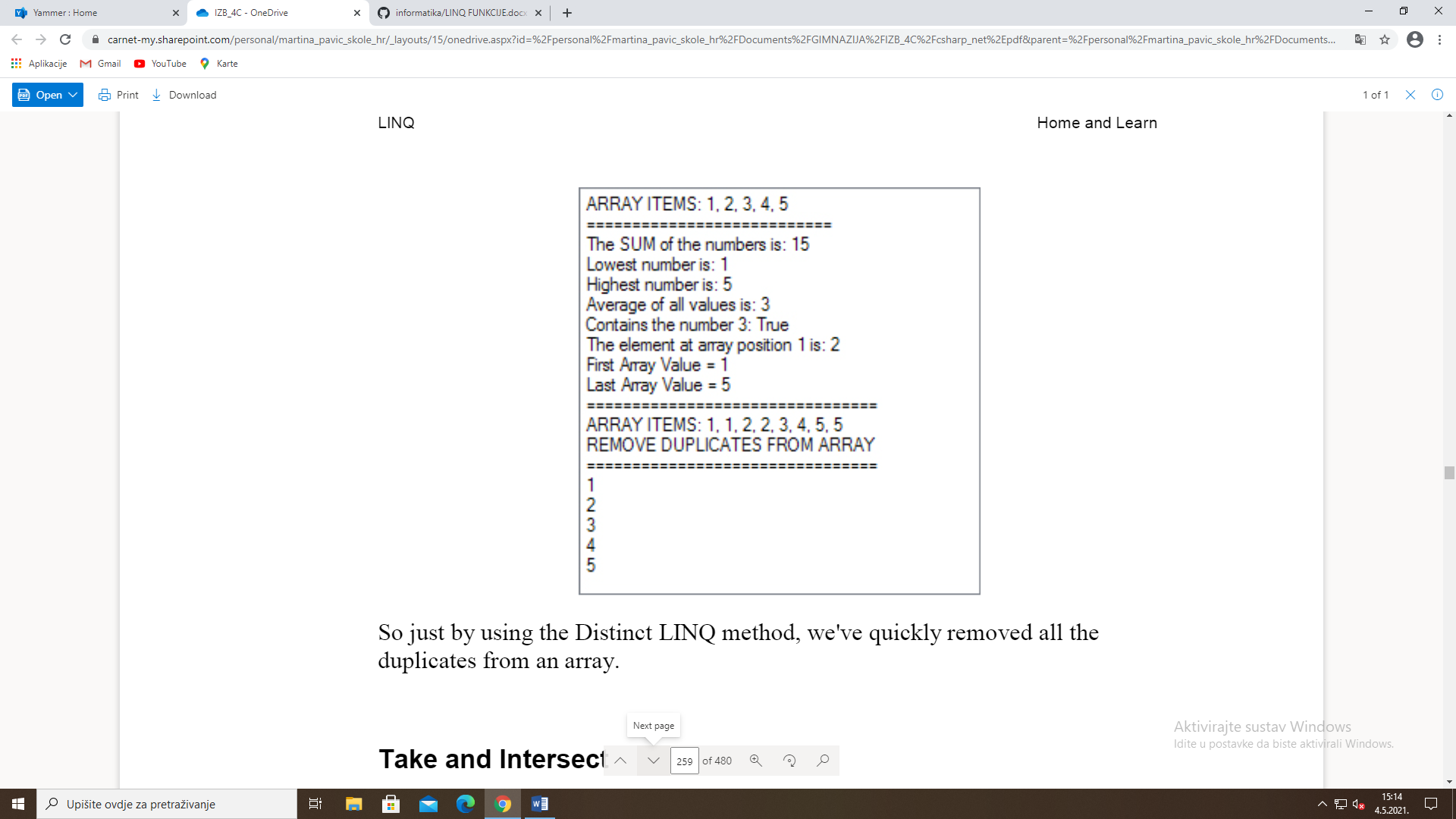
**var distinctNums = aryNums.Distinct();**

**foreach (var num in distinctNums)**

**{**

**listBox1.Items.Add(num.ToString());**

**}**



1. **Take & Intersect**

Metoda *Take* omogućuje nam uzimanje određenog broja elemenata iz jednoga niza, dok metoda *Intersect* provjera koliko se elemenata prvog niza nalazi u drugom (novom) nizu. Njihov je poziv jako jednostavan.

**var winners = lotNums.Take(6);**

**var myNumbers = chosen.Intersect(winners);**