Übung in Prog 2, Thema: Wiederholung aus Prog 1.

1 Nachimplementierung der Funktion Math.Clamp aus der .Net Core Library

Name der Funktion soll CustomClamp lauten.

Die Signature der Funktion:

```
public static double CustomClamp(double value, double min, double max)
```

Verhalten:

Das Verhalten der Funktion kann unter folgender Dokumentation nachgelesen werden.

Testprogramm soll folgendes sein:

```
Console.WriteLine($"Math.CustomClamp(2.0, 0.0, 4.0) = {CustomClamp(2.0, 0.0, 4.0)}");
Console.WriteLine($"Math.CustomClamp(-2.0, 0.0, 4.0) = {CustomClamp(-2.0, 0.0,
4.0)}");
Console.WriteLine($"Math.CustomClamp(900.0, 1.0, 800.0) = {CustomClamp(900.0, 1.0,
800.0)}");
```

Ausgabe der Funktion sollte folgende sein:

Math.Clamp(2.0, 0.0, 4.0) = 2
Math.Clamp(-2.0, 0.0, 4.0) = 0
Math.Clamp(900.0, 1.0, 800.0) = 800

2 Array Functions

2.1 GetAverage

Implementiere die Funktion namens GetAverage.

Die Signature der Funktion:

```
public static double GetAverage(double[] values)
```

Verhalten:

Es soll der Durchschnittswert aus den Element des arrays als Parameter der Funktion zurück gegeben werden.

Testprogramm soll folgendes sein:

```
double[] numbers = new double[] { 2.0, 2.0, 8.0};
Console.WriteLine($"GetAverage(numbers) = {GetAverage(numbers)}");
numbers = new double[] { 5, 2.5 };
Console.WriteLine($"GetAverage(numbers) = {GetAverage(numbers)}");
```

Ausgabe der Funktion sollte folgende sein:

GetAverage(numbers) = 4 GetAverage(numbers) = 3.75

2.2 IsUnique

Implementiere die Funktion namens IsUnique.

Die Signature der Funktion:

```
public static bool IsUnique(string[] values, string valueToBeUnique)
```

Verhalten: Die Funktion gibt true zurück wenn der Parameter valueToBeUnique genau einmal im Array existiert. Die Funktion gibt false zurück wenn der Parameter valueToBeUnique nicht oder mehrmals im Array vorkommt.

Testprogramm soll folgendes sein:

```
string[] names = new string[] {"Max", "Flo", "Alf", "Flo"};
Console.WriteLine($"IsUnique(names, \"Max\") = {IsUnique(names, "Max")}");
Console.WriteLine($"IsUnique(names, \"Flo\") = {IsUnique(names, "Flo")}");
Console.WriteLine($"IsUnique(names, \"Anna\") = {IsUnique(names, "Anna")}");
```

Ausgabe der Funktion sollte folgende sein:

IsUnique(names, Max) = true
IsUnique(names, Flo) = false
IsUnique(names, Anno) = false

2.3 GetUniqueValues

Implementiere die Funktion namens GetUniqueValues.

Die Signature der Funktion:

```
public static string[] GetUniqueValues(string[] values)
```

Verhalten:

Gibt ein Array zurück die alle Werte zurück, welche nur genau einmal im Array vorkommen. Dabei soll das Array eine Länge exakt für alle einzigartigen Element haben.

Testprogramm soll folgendes sein:

```
string[] names = new string[] {"Max", "Flo", "Alf", "Flo", "Alfred"};
string[] allUniques = GetUniqueValues(names);

Console.WriteLine("List of all unique value:");

foreach(string name in allUniques)
{
    Console.WriteLine(name);
}
```

Ausgabe der Funktion sollte folgende sein:

List of all unique value:
Max
Alf Alfred

2.4 GetValuesWithCount

Implementiere die Funktion namens GetUniqueValuesWithCount.

Die Signature der Funktion:

```
public static ValueWithCount[] GetValuesWithCount(string[] values)
```

Verhalten:

Gibt ein Array zurück welche die Länge aller einzigartigen Elemente hat. Jedes Element ist vom UniqueValue. Eine UniqueValue hat mindestens folgende Properties haben:

```
public string Value { get; private set; } // Wert
public int Count { get; private set; } // Wie oft der Wert vorkommt
```

Testprogramm soll folgendes sein:

```
string[] names = new string[] {"Max", "Flo", "Alf", "Flo", "Alfred", "Alfred",
"Alfred"};
ValueWithCount[] valuesWithCount = GetValuesWithCount(names);

Console.WriteLine("List of all values with count:");

foreach(ValueWithCount valueAndCount in valuesWithCount)
{
    Console.WriteLine($"Value: {valueAndCount.Value}, Count: {valueAndCount.Count}");
}
```

Ausgabe der Funktion sollte folgende sein:

List of all values with count:
Value: Max, Count: 1
Value: Flo, Count: 2
Value: Alf, Count: 1
Value: Alfred, Count: 3

3 Klasse ChatUser und static

Implementiere eine Klasse namens ChatUser.

Die Klasse soll folgende Felder/Methoden mindestens haben.

```
public string Name {get; private set;}

public void TypeMessage(string message)

public static string GetAllMessages()
```

Verhalten:

Wenn eine Instance erstellt wird, soll ein string parameter übergeben werden. Beim instantiieren soll das Property Name gleich den Parameter gesetzt werden.
Beim Aufrufen der Methode GetAllMessages sollen alle bisherigen eingebend Nachrichten von Nutzern als string zurück gegeben.
Ein Nachricht soll durch die Methode TypeMessage eingeben werden. Eine Nachricht soll folgendermassen aufgebaut sein. Name des Nutzer: Eingebende Nachricht. Dabei kommt jeder Nachricht auch in eine neue Zeile.

Ein Beispiel:

```
ChatUser chatUser = new ChatUser("HappyDude42");
user.TypeMessage("Hi everybody !");
user.TypeMessage("How are you doing ?");
Console.WriteLine(ChatUser.GetAllMessages());
```

Ausgabe ist:

HappyDude42: Hi everybody HappyDude42: How are you doing ?

Testprogramm soll folgendes sein:

```
ChatUser happy = new ChatUser("HappyDude42");
ChatUser meier = new ChatUser("Meier");
happy.TypeMessage("Hi everybody !");
meier.TypeMessage("Hihi");
happy.TypeMessage("How are you doing ?");
Console.WriteLine(ChatUser.GetAllMessages());
```

Ausgabe der Funktion sollte folgende sein:

HappyDude42: Hi everybody
Meier: Hihi
HappyDude42: How are you doing ?