Exercise: List

Fill in the blanks so that the list is printed

```
List<String> list = new List<String>();

list.add("Andy");
list.add("Bart");
list.add("Carl");
list.add("Doug");
list.add("Elmo");

foreach (var ____ in ____) {
    Console.WriteLine(____);
}
```

Answer:

```
foreach(var name in list)
{
  Console.Writeline(name);
}
```

SDU

Exercise: LinkedList- What is the output of 3 loops?

```
public static void Main(string args)
{
    LinkedList<int> list = new LinkedList<int>();
    list.AddFirst(2);
    list.AddFirst(8);
    list.AddFirst(5);
    list.AddFirst(1);

    foreach (var number in list)
    {
        Console.Write(number + " ");
    }

    Console.WriteLine();

    foreach (var number in list.Reverse())
    {
        Console.Write(number + " ");
    }

    Console.WriteLine();

    foreach (var number in list)
    {
        Console.Write(number + " ");
    }
}
```

Answer: b
a
b

a) 2 8 5 1

b) 1 5 8 2

c) 1 2 5 8

d) 2 1 8 5

SDU

Example: Using the Comparable Interface

```
public class SortRectangles
{
    public static void Main(string[] args)
    {
        ComparableRectangle[] rectangles =
        {
            new ComparableRectangle(3.4, 5.4),
            new ComparableRectangle(13.24, 55.4),
            new ComparableRectangle(7.4, 35.4),
            new ComparableRectangle(1.4, 25.4)
        };

        Array.Sort(rectangles);

        foreach (var rectangle in rectangles)
        {
            Console.WriteLine(rectangle);
        }
    }
}
```

```
Width: 3.4 Height: 5.4 Area: 18.36
Width: 1.4 Height: 25.4 Area: 35.559999999999995
Width: 7.4 Height: 35.4 Area: 261.96
Width: 13.24 Height: 55.4 Area: 733.496
```

How could we use the sort method to sort an array of rectangle objects?

Answer: The Array.Sort() method in C# is used to sort the elements of an array. However, for it to work correctly with ComparableRectangle objects, the ComparableRectangle class **must implement the IComparable interface**.

SDU

Example Cont'd with sorting

```
public class SortStringByLength {
  public static void Main(string[] args) {
    string[] cities = {"Atlanta", "Savannah", "New York", "Dallas"};
    Array.Sort(cities, new MyComparer());
    foreach (var city in cities) {
      Console.WriteLine(city + " ");
    }
  }
  public static class MyComparer: IComparer<String> {

    public int Compare(String s1, String s2) {
    return s1.length() - s2.length();
    }
  }
}
```

What is special about the sort method?

Answer: Overloaded method to sort the strings in the string array through IComparer interface using a new criteria e.g. length of the string

What will be the output, if I replace
Array.Sort(cities, new MyComparer());
with
Array.Sort(cities);

```
Dallas Atlanta Savannah New York
```

Answer: Strings will be sorted alphabetically using IComparable interface

SortedSet

- The SortedSet class guarantees that the elements in the set are sorted.
- Elements can also be ordered
  - using the IComparable interface, if the elements implements the comparable interface.
  - By specifying a comparer as an argument to the SortedSet constructor.

```
public class MySetWithCompr {

    public static void Main(String[] a){

SortedSet<string> ss = new SortedSet <string>();

        ss.Add("RED");
        ss.Add("ORANGE");
        ss.Add("BLUE");
        ss.Add("GREEN");
        Console.WriteLine(String.Join(" ", ss));
    }
}
```

Output???

Answer:
Blue, Green,Red, Orange

SDU

SortedSet: IComparer as an argument to the SortedSet constructor.

```
public class MySetWithCompr {
    public static void Main(String[] a){

        SortedSet<string> ss = new SortedSet<string>(new MyComp());

        ss.Add("RED");
        ss.Add("ORANGE");
        ss.Add("BLUE");
        ss.Add("GREEN");
        Console.WriteLine(String.Join(" ", ss));
    }
}
class MyComp: IComparer<String>{

    public int Compare(String str1, String str2) {
        return str1.length()-str2.length();
    }
}
```

Output???

Answer:
Red, Blue, Green Orange

SDU

Mini Quiz: What is the output????

```
public class HashSetExample
{
    public static void Main(string[] args)
    {
        HashSet<string> hashSet = new HashSet<string>();

        hashSet.Add("Geeks");
        hashSet.Add("For");
        hashSet.Add("Geeks");
        hashSet.Add("GeeksforGeeks");

        Console.WriteLine(string.Join(", ", hashSet));
    }
}
```

a Or b ???

a) [Geeks, For, Geeks, GeeksforGeeks]

b) [GeeksforGeeks, Geeks, For]

Answer: **b**

SDU

SortedDictionary Traversal

```
public class GFG
{
    public static void Main(string args)
    {
        SortedDictionary<string, string> foodTable = new SortedDictionary<string,string>();
        foodTable.Add("A", "Angular");
        foodTable.Add("P", "Python");
        foodTable.Add("J", "Java");

        foreach (var set in foodTable)
        {
            Console.WriteLine(set.Key + " = " + set.Value);
        }
    }
}
```

Answers:
A = Angular
J = Java
P = Python

SDU

Mini Quiz: What is the output????

```
class Main
{
    public static void Main(string args)
    {
        SortedDictionary<string, int> numbers = new SortedDictionary<string, int>();

        numbers.Add("One", 1);
        numbers.Add("Two", 2);
        numbers.Add("Three", 3);

        Console.WriteLine("Sorted Dictionary: " + String.Join(", ", numbers));
        Console.WriteLine("Keys: " + String.Join(", ", numbers.Keys));
        Console.WriteLine("Values: " + String.Join(", ", numbers.Values));
    }
}
```

Answer:
Sorted Dictionary: [One, 1], [Three, 3], [Two, 2]
Keys: One, Three, Two
Values: 1, 3, 2

SDU

Mini Quiz: What is the output????

```
public class DictionaryExample
{
    public static void Main(string args)
    {
        IDictionary<string, int> dict = new SortedDictionary<string, int>();

        dict.Add("John", 23);
        dict.Add("Monty", 27);
        dict.Add("Richard", 21);
        dict.Add("Devid", 19);

        Console.WriteLine("Before adding duplicate keys:");
        Console.WriteLine(string.Join(", ", dict));

        dict.Add("Monty", 25);

        Console.WriteLine("\nAfter adding duplicate keys:");
        Console.WriteLine(string.Join(", ", dict));
    }
}
```

Answer:
Before adding duplicate keys:
[Devid, 25], [John, 22], [Monty, 23], [Richard, 24]

Following error will occur:
Unhandled exception. System.ArgumentException: An item with the same key has already been added. Key: [Monty, 25]

What if I replace **dict[“Monty”]=25;**

Answer: Monty value will be replaced by 25

SDU

NUnit Attributes: Example

```
[OneTimeSetUp]
public void Init()
{
    Console.WriteLine("OneTimeSetUp executed");
}

[SetUp]
public void Setup()
{
    Console.WriteLine("SetUp executed");
}

[Test]
public void TestEqual()
{
    int result = ageComparer.Compare(p1, p2);
    Assert.That(0, Is.EqualTo(result));
}

[Test]
public void TestGreaterThan()
{
    int result = ageComparer.Compare(p1, p2);
    Assert.That(result, Is.GreaterThanOrEqualTo(1));
}

[TearDown]
public void Teardown()
{
    Console.WriteLine("TearDown executed");
}

[OneTimeTearDown]
public void Cleanup()
{
    Console.WriteLine("OneTimeTearDown executed");
}
```

Answer:

OneTimeSetUp executed
SetUp executed
====== Equal TEST EXECUTED =======
TearDown executed
OneTimeTearDown executed

What will be the execution flow, If Ignore is added to **GreaterThan** method????

SDU