**Sorting**



These exercises help to explore more about sorting.

Exercise 1 – 5 cover key aspects of this topic and will be a very helpful practice for the assessed exercise.

Exercise 6 - 8 are extra tasks, more advanced, to complement your knowledge and often given at job interviews.

**Exercise 1**

Create a Console Application in C#. Implement and test the Insertion Sort. Add the code of the insertion sort (see lecture slides) in the class Program.

You can use the following code (in main) to test Insertion Sort on an array of random numbers

// create an array of integers (randomly chosen) to test the sorting

int Min = 0;

int Max = 200;

int[] test2 = new int[10];

Random randNum = new Random();

for (int i = 0; i < test2.Length; i++)

{

test2[i] = randNum.Next(Min, Max);

}

InsertionSort(test2);

Console.WriteLine("Insertion Sort - AFTER SORTING");

for (int i = 0; i < test2.Length; i++)

{

Console.WriteLine(test2[i]);

}

Console.ReadKey();

Add a class **Book** to your project with members string ISBN; string Title; string Author.

Implement an Insertion Sort method to sort an array of Books. We need to opportunely modify the code of the Insertion Sort that sort an array of int.

How can we sort an array of books by ISBN ? How can we sort it by Title ? How can we change the sorting order ? Test that the sorting of an array of books work correctly.

If you do not remember (please revise the materials of weeks 1 and 2), you can create an array of books in this way in your Main()

string[] titles = {"Writing Solid Code",

"Objects First","Programming Gems",

"Head First Java","The C Programming Language",

"Mythical Man Month","The Art of Programming",

"Coding Complete","Design Patterns",

"ZZ"};

string[] authors = { "Maguire", "Kolling", "Bentley", "Sierra", "Richie", "Brooks", "Knuth", "McConnal", "Gamma", "Weiss" };

string[] isbns = { "948343", "849328493", "38948932", "394834342", "983492389",

"84928334", "4839455", "21331322", "348923948", "43893284",

"9483294", "9823943" };

Book[] library = new Book[10];

// create an array of books

for (int i = 0; i < library.Length; i++)

{

library[i] = new Book(isbns[i], titles[i], authors[i]);

}

**Exercise 2**

In the same project of Exercise 1 create a static method (in the class Program) called *IsInorder* that takes an array of integers and returns (true / false) whether the array is sorted in an ascending order. Use this method to check whether an array of int is correctly sorted

**Exercise 3**

In the same project of Exercise 1, complete (in the class Program) the generic static method (notice the generic syntax)

static bool IsInOrder<T>(T[] a) where T : IComparable

{

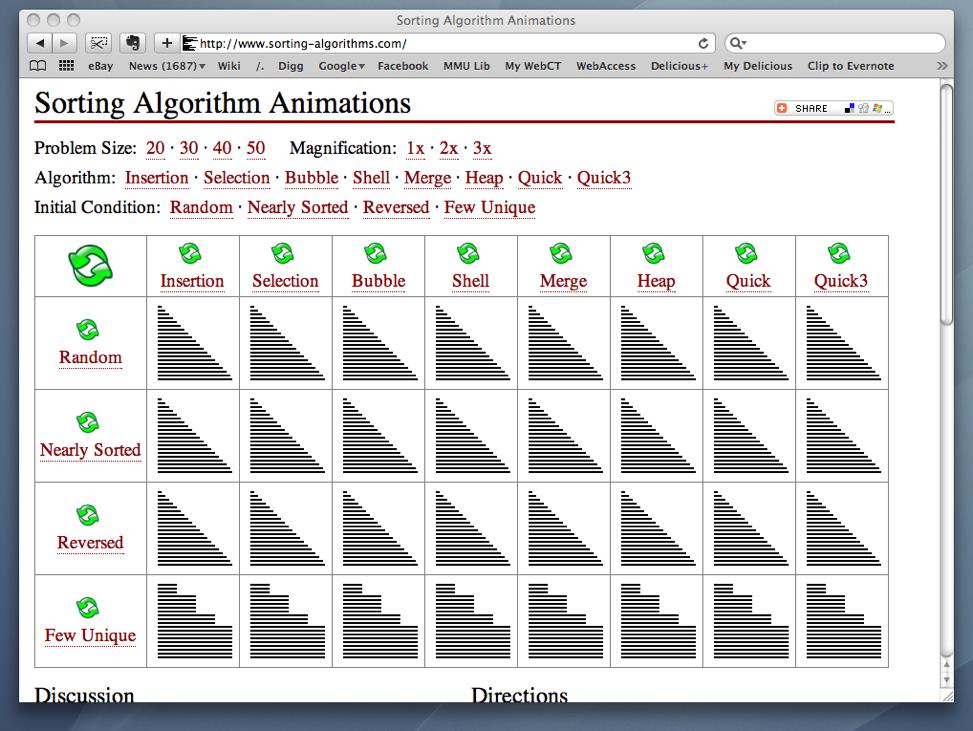
}

which returns true if the array **a** is sorted in an ascending order and returns false if the array **a** is unsorted. This is the generic version of the method done in Exercise 2. Use this method to check whether an array of books is correctly sorted. Can you also use this method to check if an array of int is correctly sorted ?

**Exercise 4**

Follow the link: *https://www.toptal.com/developers/sorting-algorithms*

to the sort comparison page. Examine the running of the different sorting algorithms (try a large input and different conditions). Is there one best sort algorithm under all input conditions?



**Exercise 5**

Implement *BubbleSort* and *Selection Sort* for integers (see lecture slides). As done for the Insertion Sort above, the code can be added in the class Program of the project done in Exercise 1.

Fill two arrays with the same random values (see how we did in Exercise 1). Confirm both algorithms work by sorting the arrays and using your IsInOrder function implemented above. Which algorithm was the fastest? Try very large arrays.

Measure the execution times using the built-in property Environment.TickCount or the built-in Stopwatch class. See <https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.stopwatch?view=netframework-4.8>

**Exercise 6 – Peaks and Valleys (often given at job interviews)**

In an array of integers “peaks” are the elements greater or equal than the adjacent elements. “Valleys” are the elements that are less or equal to the adjacent integers.

Implement a method that takes as input an array of integers and sort the array into an alternating sequence of peaks and valleys.

*From G. McDowell, Cracking the Coding Interview*

*Example*

*Input = [5, 3, 1, 2, 3]*

*Output = [5, 1, 3, 2, 3]*

**Exercise 7 [Advanced] (often given at job interviews)**

Write a method that takes as input an array of strings and sort the array in a such a way that that all the anagrams are next to each other.

*From G. McDowell, Cracking the Coding Interview*

**Exercise 8 [Advanced]**

Donald Shell modified the standard Insertion sort to create what is now called *Shell Sort*. Implement shell sort for integers and objects (such as the Books class used in previous exercises).