1: Customers’ product scores (with Tutor)

An array contains the scores related to reviews of products sold online. Each score is a *double* value in the range of 1.0 to 5.0. The scores are stored in the array in *descending* order (from highest to lowest) and are known when the array is declared (array initialisation).
Write a program that uses the element of the above array and stores the same information in *ascending* order inside an *additional* array.

Solution

The elements of the *scores* array are copied into the *scoreReverse* array as follows: the last element of *scores* is placed as the first of *scoresReverse*, the penultimate as the second, etc.

```
using System;

namespace ReverseScores
{
   class Program
   {
      static void Main(string[] args)
      {
         // array is initialised with some values
         double[] scores = { 4.4, 3.3, 3.1, 3.0, 2.45, 2.3, 1.1, 1.01 };

         // second array that will store the information in reverse order
         double[] scoresReverse = new double[scores.Length];

         // loop through the element of the scores array
         for (int i = 0, j = scoresReverse.Length - 1; i < scores.Length; i++, j--)
         {
            // starting from last position on the destination array: scoresReverse.Le
ngth - 1
            // starting from first position (0) on source array
            scoresReverse[j] = scores[i];
         }

         // printing the original array
         for (int i = 0; i < scores.Length; i++)
         {
            Console.Write(scores[i] + " ");
         }

         Console.WriteLine();

         // printing the reversed array
         for (int i = 0; i < scores.Length; i++)
         {
            Console.Write(scoresReverse[i] + " ");
         }
      }
   }
}
```

2: Average of Scores (independent work)

Write a program that calculates the average of all the scores (similar to the previous exercise) stored inside an unordered array of *integers*. The scores are not known in advance and must be provided as input from the keyboard (at runtime). The program should check that the input is provided in the correct format, i.e., each score is within the range 1-5. Note that whilst the array contains integer values, the result of the average calculation can be a non-integer value.

Solution

The possible following solution uses a *foreach* loop to iterate over the elements of the array and calculate their sum. The sum is finally divided by the number of elements of the array to obtain the average score. This solution uses a *do while* statement to check that the provided input is correct. Please find more about the *do while* at:
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/statements/iteration-statements#the-do-statement

```
using System;

namespace Average
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Insert the number of scores:");
         int size = Convert.ToInt32(Console.ReadLine());
         int[] scores = new int[size];


         for (int i = 0; i < scores.Length; i++)
         {
            int value;

            // using a do while as the block needs to be executed at least 1 time
            // the loop is executed while the inserted value is outside of the expect
ed range
            // the value is read again at each iteration

            do
            {
               Console.WriteLine($"Insert score {i + 1}: ");
               value = Convert.ToInt32(Console.ReadLine());
            } while (value < 1 || value > 5);

            // the value is correct, can be stored inside the array
            scores[i] = value;
         }
```

```
            // going through all the array elements and calculating their sum
            int sum = 0;
            foreach (int v in scores)
            {
                Console.Write(v + " ");
                sum += v;
            }

            Console.WriteLine();

            // divide the sum by the number of elements to calculate the average
            // casting is used to obtain a double as result of the division
            double average = (double) sum / scores.Length;
            Console.WriteLine("The average of the scores is: " + average);
        }
    }
}
```

3: Best performer of a Marathon (independent work)

Write a program that finds the best performer of a marathon (i.e., smaller time in minutes). The program receives as input the number of contestants. The timings are expressed as *double* values generated randomly (>=20.0, <100.0). To generate the random double values in the requested range use the instruction *Random.NextDouble()* as suggested in the hint.

Solution

This is the problem of finding the minimum value of the array. Please see comments in the source code as explanation of the solution.

```
using System;

namespace Min
{
    class Program
    {
        static void Main(string[] args)
        {
            Random r = new Random();

            // read the number of contestants as input
            Console.WriteLine("Insert the number of contestants: ");
            int contestants = Convert.ToInt32(Console.ReadLine());


            // declare the array
            double[] timings = new double[contestants];

            // generate the timings randomly
            for (int i = 0; i < timings.Length; i++)
            {
                // should generate numbers in the requested range
                // from 20.0 (included) to 100.0 (excluded)
                timings[i] = 80 * r.NextDouble() + 20.0;
                Console.WriteLine(timings[i]);
            }

            // find the best contestant: find the min of the array

            // assuming the min is in position 0
            int firstIndex = 0;
            double first = timings[0];

            // loop through the array elements (starting from position 1)
            for (int i = 1; i < timings.Length; i++)
            {
                // the array value is smaller than the current min: updating
                if (timings[i] < first)
                {
                    // the min is updated with the current array value
                    first = timings[i];
                    firstIndex = i;
                }
            }

            Console.WriteLine("Best timing (min) is: " + first + " + " located at index
" + firstIndex]);
        }
    }
}
```

Extra problem (independent work)

Modify the previous program (1) to reorder the scores without using an additional array.

Solution

```
using System;

namespace Reverse2
{
    class Program
    {
        static void Main(string[] args)
        {
            // array is initiliased with some random values
            double[] scores = { 4.4, 3.3, 3.1, 3.0, 2.45, 2.3, 1.1,
1.01, 0.5 };

            // used as variable to support the swapping of the elements
            double buffer;

            // two indexes are used within the loop
            // the loops terminates when the indexes are both in the
middle of the array (i=j)
            for (int i = 0, j = scores.Length - 1; i < j; i++, j--)
            {
                // values is saved in the buffer to be used later
                buffer = scores[i];

                // swap the elements
                scores[i] = scores[j];

                // use the previously saved value
                scores[j] = buffer;
            }

            for (int i = 0; i < scores.Length; i++)
            {
                Console.WriteLine(scores[i]);
            }
        }
    }
}
```