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 1.0 to 5.0. The scores are stored in the array in *descending* order (from highest to lowest) and they 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.

Hint

```
class Program
{
   static void Main(string[] args)
   {
      // array (scores) is initialised with some values

      // second array (scoresReverse)
      // will store the information in reverse order

      // loop through the element of the two arrays
      for (   )
      {
         // read starting from the first element of scores
         // copy starting from the last position of scoresReverse
      }

      // print the array scores

      // print the array scoresReverse
   }
}
```

2: Average of Scores (independent work)

Write a program that calculates the average of all the scores (similar to the previous exercise) that are stored inside an unordered array of *integers*. The scores are not known in advance and need to 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.
The hint 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

Hint

```
class Program
{
   static void Main(string[] args)
   {
      Console.WriteLine("Insert the number of scores:");
      int size = Convert.ToInt32(Console.ReadLine());
      // declare an array scores with size elements
```

```
        for (    ) // iterate over the scores array
        {
            int value;

            do
            {

              // read input from the keyboard

             } while (     ); // loop while value is incorrect

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

        // go through all the array elements and calculate their sum

        // divide the result by the number of elements

    }
}
```

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.

Hint

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

        // read the number of contestants as input

        // declare an array timings
        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 in 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 (    ) // complete the for definition
        {
            // if the array value is smaller than the current min: updating
            if (    ) // complete with right condition
            {
                // the min is updated with the current array value
            }

        }

        Console.WriteLine("Best timing (min) is: " + timings[firstIndex]);
    }
}
```

Extra problems

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

b) Modify the previous program (3) to also find the second-best performer.