Candidate Challenge

# Hello World

Hello, candidate! We are happy you reached out to us.

We created this challenge to get a grasp on your know-how. For us this document represents a reference point on how to classify your skills. The tasks in this challenge rank from easy to difficult and we therefore don’t expect you to solve everything without difficulties. Just try to solve whatever you can and as much as possible.

If you have any questions or encounter a problem, don’t hesitate to contact us!

# Instructions

For a programming task, your solution must contain (i) an explanation of your solution to the problem, (ii) the C# code, in a form that we can run it, (iii) instructions how to run it. Also put the source code into your solution document. For all programming tasks, it is not allowed to use any external libraries (“import/using’s”) if not stated otherwise.

To start working follow these steps:

1. Import the project into your favorite C# IDE (we recommend [Visual Studio](https://visualstudio.microsoft.com/vs/)) using .NET Framework v4.7 (or higher)
2. At this point, you will find the classes in the project *coolOrange\_CandidateChallenge* and the Test in the *coolOrange\_CandidateChallengeTest* node.
3. Implement the tasks by refining the ones you find in the classes or by creating new ones if stated.
4. Use [GitHub](http://github.com/) as your code base. Once you are finished send us the link to the public repository and we can review it.

# Tasks

## Basic Operations

Implement in C# the class Array, which offers basic operations over one-dimensional and two-dimensional arrays. *All* the static methods *must* be implemented. The signature of the methods in the class are the following:

1. public static int FindMaxValue (int[] array, int position1, int position2)

returns the maximum value found in the array between two positions.

1. public static int FindMinValuePosition(int[] array, int position1, int position2)

returns the position of the minimum value in the array between two positions.

1. public static void Swap(int[] array, int position1, int position2)

swaps the elements of the two passed positions in the array.

1. public static void ShiftLeftByOne(int[] array, int position1, int position2)

shifts all the elements between the passed positions to the left by one position and leaves position2 unchanged.

1. public static int[] CreateRandomArray(int size, int minValue, int maxValue)

creates and returns an array with the passed size and random values between min and max (use the Random Class of C#).

1. public static int[,] CreateRandomMatrix(int rows, int columns, int minValue, int maxValue)

creates and returns a two-dimensional array with rows and columns of random elements with values between min and max (use the Random Class of C#).

1. public static int[,] CopyArray(int[] array)

returns a two-dimensional array with the original and the copy of the array.

1. public static int FindInSortedArray(int[] array, int number)

returns a random position of the passed number in the sorted array. The function returns −1 if the number is not present in the array.   
The method assumes that the array is already sorted.

*Bonus*: Exploit the fact that the array is sorted to find an *efficient* algorithm.  
(Hint: Binary search algorithm)

For this assignment, we have designed UnitTests that your code should satisfy.

## Recursion

A palindrome is a phrase that reads the same forward and backward (examples: ‘racecar’, radar’, ‘noon’, or ‘rats live on no evil star’). By extension we call every string a palindrome that reads the same from left to right and from right to left.

1. Develop a *recursive* algorithm that takes a string as input and decides whether the string is a palindrome. Implement your algorithm in the PalindromeChecker class.

For this assignment, we have designed UnitTests that your code should satisfy.

## Object Orientated Programming (OOP)

The exercise consists of two parts: One part (1.) is the implementation of classes, interfaces, and functions… the other part (2.) is defining and implementing the according UnitTests. If you like challenges or have already experience with UnitTests, start with defining and implementing them first. (Test-driven development)

1. Define a class Task modeling simple everyday’s tasks, such as “Doing Homework”, “Eating Lunch”, “Programming”.

* The class should implement your interfaces IPriority, IComplexity, and the IComparable from the System namespace.
* Every task has three variables: Name, Priority, and Complexity
* The constructor sets up a Task object with the given Name, no Complexity, and Priority equals to MED\_PRIORITY
* The class should implement all the abstract methods defined in interfaces IPriority and IComplexity
* Implement the CompareTo() method declared in the Comparable interface, that compares this Task with another. The comparison should be based on the tasks’ priorities.
* Define the interface IPriority, that defines
  + three levels of priority (MIN\_PRIORITY, MED\_PRIORITY, MAX\_PRIORITY)
  + abstract methods SetPriority() that sets the object's priority level, and GetPriority() that returns the object's priority level
* Define the interface IComplexity that defines the abstract methods SetComplexity() that sets the object's complexity level and GetComplexity() that returns the object's complexity level
* Finally, define a class TaskDriver with a main method that creates three instances of Task
  + prints out as shown in the output window below.
    - the three tasks
    - the task which has the highest priority

Output:

TO-DO

---------

Doing Homework priority: 10 complexity: 8

Eating Lunch priority: 1 complexity: 2

Programming priority: 5 complexity: 5

Doing Homework is one of the most important tasks

1. Create UnitTests in the *coolOrange\_CandidateChallengeTest* projectso you are sure all methods in the class Task are working as intended.
   * Use [NUnit](https://github.com/nunit/docs/wiki/NUnit-Documentation) as unit testing framework. It is already referenced in the project.
   * Define a class TaskTests.
   * Define and implement according tests for the class Task you implemented in the previous exercise.
   * As a reference you can check the UnitTests from the PalindromeCheckerTests class.