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/usings”) if not stated otherwise.

To start working follow these steps:

1. Download the [Zip file](https://github.com/lustricker/coolOrange_CandidateChallenge/archive/master.zip) to some directory and unzip it. (That zip file contains a “project”.)
2. Import the project into your favorite C# IDE (we recommend [Visual Studio](https://visualstudio.microsoft.com/vs/))
3. At this point, you will find the classes in the project *coolOrange\_CandidateChallenge* and the Test in the *coolOrange\_CandidateChallengeTest* node.
4. Implement the tasks by refining the ones you find in the classes or by creating new ones if stated.
5. Make sure that they pass the Unit tests in the test directory.
6. Use [GitHub](http://github.com/) as your code base. Once you are finish send us the link to the public repository and we can review it.

# Tasks

## Basic Operations

Implement in C# the class ArrayUtility, which offers basic operations over onedimensional and two-dimensional arrays. *All* methods *must* be implemented as class methods (i.e., static methods). The signature of the methods in the ArrayUtility class are the following:

1. public static int findMax(int[] A, int i, int j)

returns the maximum value occurring in the array A between position i and j.

1. public static int findMinPos(int[] A, int i, int j)

returns the position of the minimum value in the array A between position i and j.

1. public static void swap(int[] A, int i, int j)

swaps the elements in position i and j in the array A.

1. public static void shiftLeft(int[] A, int i, int j)

shifts to the left all the elements of the array A, from position j down to position i (i.e., moves the element in position k to position k − 1 for all i < k ≤ j, and leaves the position j unchanged).

1. public static int[] createRandomArray(int size, int min, int max)

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

1. public static int[][] createRandomMatrix(int rows, int cols, int min, int max)

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

1. public static int[] copyArray(int[] A)

returns a two-dimensional array that is a copy of A.

1. public static int findInSortedArrary(int[] A, int q)

returns a (not the!) position of the number q in the sorted array A (returns −1 if q is not present in A). The method assumes that the array A is sorted, it need not be correct if A is not sorted.

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

## Recursion

1. 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.   
   Develop a *recursive* algorithm that takes as input a string 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)

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. In case you are an advanced developer, write the UnitTests before implementing the methods. (Test Driven Development).
   * 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.