

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 - Community](#)) using `.NET Framework v4.8.x`
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](#) to host your code by creating a new public repository. Once you are finished send us the link to the repository so we can review it.

Tasks

Basic Operations

Implement the static methods in the `Array` class, that offers basic operations for one-dimensional and two-dimensional arrays. 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.
2. `public static int FindMinValuePosition(int[] array, int position1, int position2)`
returns the position of the minimum value in the array between two positions.
3. `public static void Swap(int[] array, int position1, int position2)`
swaps the elements of the two passed positions in the array.

4. `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.
5. `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 from the `System` namespace).
6. `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 from the `System` namespace).
7. `public static int[][] CopyArray(int[] array)`
returns a two-dimensional array with the original and the copy of the array.
8. `public static int FindInSortedArray(int[] array, int number)`
returns a position of the given 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:

1. Implementation of classes, interfaces, and methods.
2. Defining and implementing the respective 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 task, such as "Doing Homework", "Eating Lunch" and "Programming".

- Define an enum `Priority`, that defines
 - o three levels of priority (`MIN_PRIORITY=1`, `MED_PRIORITY=5`, `MAX_PRIORITY=10`)
- Define an interface `IPriority`, that defines
 - o the method `SetPriority()` – that sets the object's priority level, and `GetPriority()` that returns the object's priority level
- Define the interface `IComplexity` that defines the methods `SetComplexity()` that sets the object's complexity level and `GetComplexity()` that returns the object's complexity level
- The class `Task` should implement your `IComplexity` and `IPriority` interfaces and the `Comparable` interface from the `System` namespace.
- Every task has a `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 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.
- Finally, define a class `TaskDriver` with a main method that creates a `List<Task>` with three instances of the `Task` class with the following values:
 - o Name: Doing Homework, Priority: 10, Complexity: 8
 - o Name: Eating Lunch, Priority: 1, Complexity: 2
 - o Name: Programming, Priority: 5, Complexity: 5
- Print the list of all `Tasks` with their values and additionally the most important task:

```
TO-DO
-----
Doing Homework priority: 10 complexity: 8
Programming priority: 5 complexity: 5
Eating Lunch priority: 1 complexity: 2
Doing Homework is one of the most important tasks
```

2. Create `UnitTests` in the `coolOrange_CandidateChallengeTest` project so you are sure all methods in the class `Task` are working as intended.
 - Use [NUnit](#) as unit testing framework. It is already referenced in the project.
 - Define a class `TaskTests`.
 - Define and implement `UnitTests` for the class `Task`.
 - As a reference you can check the `UnitTests` from the `PalindromeCheckerTests` class.