CoolOrange Candidate Challenge

# 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”) if not stated otherwise.

To start working follow these steps:

1. Download the zip file “.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
3. At this point, you will find the classes in the project *coolOrange\_CandidateChallenge* and the Test in the *coolOrange\_CandidateChallengeTest* node.
4. Implement your methods by refining the ones you find in the classes.
5. Make sure that they pass the Unit tests in the test directory.

# Exercises

## 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.

## Maximal Length of Ascents in Arrays

1. Consider an array A[1..n] of integers. A subarray of A is a contiguous segment of A. We denote the subarray from position k to position l as A[k..l]. The subarray A[k..l] is an ascent if A[j] ≤ A[j + 1] for all j where k ≤ j < l. In other words, an ascent is a nondecreasing segment of A. We want to compute the maximal length of an ascent in A. For instance, for the array A = [3, 1, 4, 2, 4, 4, 5, 3], the maximal length of an ascent would be 4, because the subarray A[4..7] = [2, 4, 4, 5] is the longest ascent in that array. We are interested in an efficient algorithm.

Implement your *iterative* algorithm maxAscent(A) that takes an array A of integers as input and returns the *maximal* length of an ascent in A in the maxAscentsInArrays class.

## Unit Tests

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