IT230 Homework 1

Score: ______/10 v1.1

The purpose of this assignment is to refresh your memory of arrays and to start thinking of efficient code.

Specifications

Suppose you're coaching kids-league baseball, and you want to keep track of which players are present at the practice field. What you need is an attendance-monitoring program for your laptop—a program that maintains a database of the players who have shown up for practice. You can use a simple data structure to hold this data. There are several actions you would like to be able to perform:

- Insert a player into the data structure when the player arrives at the field.
- Check to see whether a particular player is present, by searching for the player's number in the structure.
- Delete a player from the data structure when that player goes home.

These three operations—insertion, searching, and deletion—are fundamental ones in most of the data storage structures we'll study in this class. In addition, there is a fourth operation, read, which is not directly covered in this assignment.

In the Source.cs file, you will see 3 methods that have not been implemented and an array called `players` that is not initialized. Do not initialize the array out of a method.

The `InsertIntoArray()` method takes an integer as a parameter. It will add the number to the `players` array. The `players` array should be increased in size if an insertion is happening while the `players` array is at capacity. For example, if the `players` array looks like [1, 4] and `InsertIntoArray()` receives a 6, then the `players` array should look like [1, 4, 6].

`SearchArray()` takes an integer as a parameter. Return true if that number exists in the `players` array, otherwise, return false.

`DeleteFromArray()` takes an integer as a parameter. If the number exists in the `players` array, delete it from the array. If the number is not in the `players` array, do nothing. Remember, elements in an array should be contiguous; do not allow holes in your array. Additionally, resize your the `player` array to never have an empty space at the end.

(Normally, having empty space at the end of your array is acceptable [to a degree], but for the sake of this lab, and the fact that we are dealing with a primitive type, shrink the `player` array when deleting every time.)

Constraints

- Do not use LINQ.
- Do not use properties.
- Excluding the `players` array, limit your utilization of static or class variables to a maximum of 5 bytes (40 bits).
- Do not edit the Program.cs file.
- Do not create additional files.
- Do not change the data type or visibility of the `players` array.
- Holes must not exist in the `players` array after `InsertIntoArray()`, `SearchArray()`, or `DeleteFromArray()` return.

Submission

The submission for this assignment is the commit ID you want to be graded. You will submit the commit ID via the Canvas assignment. If you need help finding the commit ID, ask for help.

10 points will be assigned by correctness and your observable understanding as reflected in your code.

Remember to write efficient code. Optimize your code wherever possible.

Tips

- You may edit the inputs.txt file for debugging purposes. The input.txt file in your repository will not be used for grading.
- It is not recommended to resize your array after every insertion/deletion.