|  |  |  |
| --- | --- | --- |
|  |  | **ISM 6225**  **Distributed Information systems** |
| Prof Clinton Daniel TA: Nagababu Veganti | | |

Assignment 1 – Programming Introduction

Primary objective: Develop familiarity with essential programming constructs

Secondary objective: develop comfort with using the IDE and GitHub

*Estimated time for newbie developers: 30 hours*

## Introduction

Full-stack application development is an essential skill needed to succeed and even survive in business analytics and/ or information systems roles, especially as AI takes over many rudimentary tasks formerly performed by analysts. This assignment introduces the essential programming constructs such as variables, selection, loops, methods, and arrays used to build such applications. Specifically, this assignment avoids the use of API methods and object-oriented programming. Those tasks are left for later assignments. This assignment also does not check for efficiency in program implementation. That is something you will develop over a lifetime in the profession. Rather, the focus is on simple programming exercises for students to learn basic industry best practices. One design goal for this assignment was to focus tightly on introductory programming structures, with a low probability that students would find ready-to-use solutions online.

This is an individual assignment, to allow every student to develop the necessary skills to become a productive contributor to project teams in this class and beyond.

## Activity

Use only **C#** as a programming Language to define methods to do the operations specified in the method signatures and hints below. The methods are listed in the recommended sequence of development. A starter Program.cs file is included in the appendix.

## Submission

Push the code to GitHub and submit the URL to Canvas. Also, get the output from a sample run that shows the use of all required methods and upload/push a screenshot to GitHub. This serves as a quick check. Submit your self-reflection as a comment on the assignment.

## Grading scheme

Each method carries 1 point. You will be graded on the following aspects for each question:

Logic (including appropriate organization of logic into methods) : 0.4

Handling all reasonable corner cases : 0.3

Descriptive comments explaining the logic to reviewer : 0.2

Self-reflection (time taken, learning, and recommendations) : 0.1

# **GitHub Link**

## Use the Program.CS File from this [GitHub Repository](https://github.com/NagababuVeganti/ISM_6225_Spring_2022_Assignment_1.git).

## Method specifications

**QUESTION 1:**

Given a string *s*, remove the vowels 'a', 'e', 'i', 'o', and 'u' from it, and return the new string.

Example 1:

Input: s = "MumaCollegeofBusiness"

Output: "MmCllgfBsnss"

Example 2:

Input: s = "aeiou"

Output: ""

**Constraints:**

* 0 <= s.length <= 10000
* s consists of **uppercase and lowercase** letters.

public string RemoveVowels(string s);

**QUESTION 2:**

Given two string arrays bulls\_string1 and bulls\_string2 , return true if the two arrays represent the same string, and false otherwise.

A string is represented by an array if the array elements concatenated in order forms the string.

Example 1:

Input: bulls\_string1  = ["Marshall", "Student",”Center”], bulls\_string2  = ["MarshallStudent ", "Center"]

Output: true

Explanation:

word1 represents string "marshall" + "student" + “center” -> "MarshallStudentCenter "

word2 represents string "MarshallStudent" + "Center" -> "MarshallStudentCenter"

The strings are the same, so return true.

Example 2:

Input: bulls\_string1  = ["Zimmerman", "School", ”of Advertising”, ”and Mass Communications”], bulls\_string2  = ["Muma", “College”,"of”, “Business"]

Output: false

Example 3:

Input: bulls\_string1  = ["University", "of", "SouthFlorida"], bulls\_string2  = ["UniversityofSouthFlorida"]

Output: true

public bool ArrayStringsAreEqual(string[] bulls\_string1 ,string[] bulls\_string2 );

**QUESTION 3:**

You are given an integer array *bull\_bucks*. The unique elements of an array are the elements that appear exactly once in the array.

Return the sum of all the unique elements of *nums*.

**Example 1:**

Input: *bull\_bucks* = [1,2,3,2]

Output: 4

Explanation: The unique elements are [1,3], and the sum is 4.

**Example 2:**

Input: *bull\_bucks* = [1,1,1,1,1]

Output: 0

Explanation: There are no unique elements, and the sum is 0.

**Example 3:**

Input: *bull\_bucks* = [1,2,3,4,5]

Output: 15

Explanation: The unique elements are [1,2,3,4,5], and the sum is 15.

Constraints:

• 1 <= *bull\_bucks*.length <= 100

• 1 <= *bull\_bucks* [i] <= 100

public int SumOfUnique(int[] *bull\_bucks*)

Hint: Use the fact that 1<= *bull\_bucks*[i]<=100 which is not very large number to store.

**QUESTION 4:**

Given a square matrix *bulls\_grid*, return the sum of the matrix diagonals.

Only include the sum of all the elements on the primary diagonal and all the elements on the secondary diagonal that are not part of the primary diagonal.

**Example 1:**

Diagram

Description automatically generated with medium confidence

Input: *bulls\_grid* = [[1,2,3],[4,5,6], [7,8,9]]

Output: 25

Explanation: Diagonals sum: 1 + 5 + 9 + 3 + 7 = 25

Notice that element mat[1][1] = 5 is counted only once.

**Example 2:**

Input: *bulls\_grid* = [[1,1,1,1], [1,1,1,1],[1,1,1,1], [1,1,1,1]]

Output: 8

**Example 3:**

Input: *bulls\_grid* = [[5]]

Output: 5

public int DiagonalSum(int[][] bulls\_grid);

**QUESTION 5:**

Given a string *bulls\_string* and an integer array *indices* of the **same length**.

The string *bulls\_string* will be shuffled such that the character at the ith position moves to *indices[i]* in the shuffled string.

Return *the shuffled string*.

**Example 1:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 5 | 6 | 7 | 0 | 2 | 1 | 3 |
| **c** | **o** | **d** | **e** | **l** | **e** | **e** | **t** |
|  |  |  |  |  |  |  |  |
| **l** | **e** | **e** | **t** | **c** | **o** | **d** | **e** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

**Input:** *bulls\_string*  = "codeleet", indices = [4,5,6,7,0,2,1,3]

**Output:** "leetcode"

**Explanation:** As shown, "codeleet" becomes "leetcode" after shuffling.

**Example 2:**

**Input:** *bulls\_string*  = "abc", indices = [0,1,2]

**Output:** "abc"

**Explanation:** After shuffling, each character remains in its position.

**Example 3:**

**Input:** *bulls\_string*  = "aiohn", indices = [3,1,4,2,0]

**Output:** "nihao"

**Example 4:**

**Input:** *bulls\_string*  = "aaiougrt", indices = [4,0,2,6,7,3,1,5]

**Output:** "arigatou"

**Example 5:**

**Input:** *bulls\_string*  = ”USF”, indices = [1,0,2]

**Output:** “SUF”

**Constraints:**

* *bulls\_string* .length == indices.length == n
* 1 <= n <= 100
* *bulls\_string*  contains only lower-case English letters.
* 0 <= indices[i] < n
* All values of indices are unique (i.e. indices is a permutation of the integers from 0 to n - 1).

public string RestoreString(string *bulls\_string* , int[] indices)

**Question 6:**

Given a 0-indexed string *bulls\_string*  and a character *ch*, reverse the segment of word that starts at index 0 and ends at the index of the first occurrence of *ch* (inclusive). If the character *ch* does not exist in *word*, do nothing.

For example, if *bulls\_string* = "abcdefd" and *ch* = "d", then you should reverse the segment that starts at 0 and ends at 3 (inclusive). The resulting string will be "dcbaefd".

Return the resulting string.

**Example 1:**

Input: *bulls\_string*  = "mumacollegeofbusiness", *ch* = "c"

Output: “camumollegeofbusiness”

Explanation: The first and only occurrence of "c" is at index 4.

Reverse the part of word from 0 to 4 (inclusive), the resulting string is "camumollegeofbusiness".

**Example 2:**

Input: *bulls\_string*  = "zimmermanschoolofadvertising", *ch* = "x"

Output: "zimmermanschoolofadvertising"

Explanation: "x" does not exist in word.

You should not do any reverse operation, the resulting string is "abcd".

**Constraints**:

1 <= *bulls\_string* .length <= 250

*bulls\_string*  consists of lowercase English letters.

*ch* is a lowercase English letter.

**Note**: You are not allowed to use any inbuilt functions.

public string ReversePrefix(string *bulls\_string* , char ch)