ISM6225 Fall24 Assignment 2

Mauricio Bermudez

Invalid Date

# Question 1:

**Me:** I am looking to review a way to check if I am missing a number in my counting array. Here is the code I currently have. Would this be correct?

int[] tempArray = new int[nums.Length];   
for (int i = 0; i < nums.Length; i++)  
{   
 if (nums[i] > 0 && nums[i] <= nums.Length)  
 {  
 tempArray[nums[i] - 1] = 1;  
 }  
}

**Copilot:** Your code snippet is a good start for identifying missing numbers in an array. However, it currently only marks the presence of numbers in the tempArray but does not return the missing numbers. To complete the functionality, you need to iterate through the tempArray and collect the indices where the value is still 0 (indicating the missing numbers). Here’s how you can modify your code to achieve this:

int[] tempArray = new int[nums.Length]; for (int i = 0; i < nums.Length; i++) { if (nums[i] > 0 && nums[i] <= nums.Length) { tempArray[nums[i] - 1] = 1; } }  
List<int> missingNumbers = new List<int>(); for (int i = 0; i < nums.Length; i++) { if (tempArray[i] == 0) { missingNumbers.Add(i + 1); } }  
return missingNumbers;

# Question 2:

**Me:** Hi there, I am trying to take an array of numbers, check if it’s odd or even, assign it to an even array if it is even and an odd array if it’s odd. Then I’d like to combine the 2 arrays, where the even array is first and the odd array is second. Here is my code, I was getting an exception as well. Would you be able to verify if I’m doing this in the right fashion, and that my code is not out of scope?

int[] evenIndex = new int[nums.Length/2];  
int[] oddIndex = new int[nums.Length/2];  
int[] sortedArray = new int[nums.Length];  
for (int i = 0; i < nums.Length; i++)  
{  
 if (nums[i] % 2 == 0)  
 {  
 evenIndex[i] = nums[i];  
 }  
 else  
 {  
 oddIndex[i] = nums[i];  
 }  
}  
sortedArray = evenIndex.Concat(oddIndex).ToArray();  
return sortedArray;

**Copilot:** Your code has a few issues that could lead to exceptions or incorrect behavior:

Array Index Out of Bounds:

You are using evenIndex[i] and oddIndex[i] directly, but i is iterating over the entire nums array. Since evenIndex and oddIndex are initialized with a size of nums.Length / 2, this will cause an IndexOutOfRangeException when i exceeds the size of these arrays. Incorrect Array Initialization:

The size of evenIndex and oddIndex assumes that exactly half of the numbers will be even and half odd, which may not always be true. This assumption can lead to incorrect results or exceptions. Unused Indices for Even and Odd Arrays:

You are not keeping track of the current index for evenIndex and oddIndex, so values are being placed at incorrect positions. Corrected Code Here is a corrected version of your method:

List<int> evenIndex = new List<int>();  
List<int> oddIndex = new List<int>()  
for (int i = 0; i < nums.Length; i++)  
{  
 if (nums[i] % 2 == 0)  
 {  
 evenIndex.Add(nums[i]);  
 }  
 else  
 {  
 oddIndex.Add(nums[i]);  
 }  
}  
return evenIndex.Concat(oddIndex).ToArray();

# Question