*Owen Lindsey*

*Professor Demland, David*

*CST-201 Exercise 3*

*09/21/2024*

**Exercise 1.4 - 1a**

**Delete ith Element from Unsorted Array**

*Algorithm Explanation*

1. *Define the DeleteIthElement method, which takes an array arr[ ] and an index ‘i’ as input. It returns a new array with the ith element removed.*
2. *First, check if the provided index ‘i’ is within the valid range (1 ≤ i ≤ arr.Length). If not, throw an ArgumentOutOfRangeException.*
3. *Create a new array newArray[ ] with a length one less than the original array arr[ ].*
4. *Use Array.Copy to copy elements from arr[ ] to newArray[ ]:*

*Copy elements from index 0 to i-1 of arr[ ] to the same positions in newArray[].*

*Copy elements from index ‘i’ to the end of arr[ ] to positions starting from i-1 in newArray[ ].*

1. *Return the newArray[ ] containing all elements except the ith one.*

**Delete ith Element from Unsorted Array Applying [1, 2, 3, 4, 5] to the list, deleting the 3rd element**

*Algorithm Explanation*

1. *Start with an array arr[ ] = [1, 2, 3, 4, 5] and i = 3.*
2. *Check if i is valid (1 ≤ 3 ≤ 5), which it is.*
3. *Create newArray[ ] of length 4.*
4. *Copy elements:*

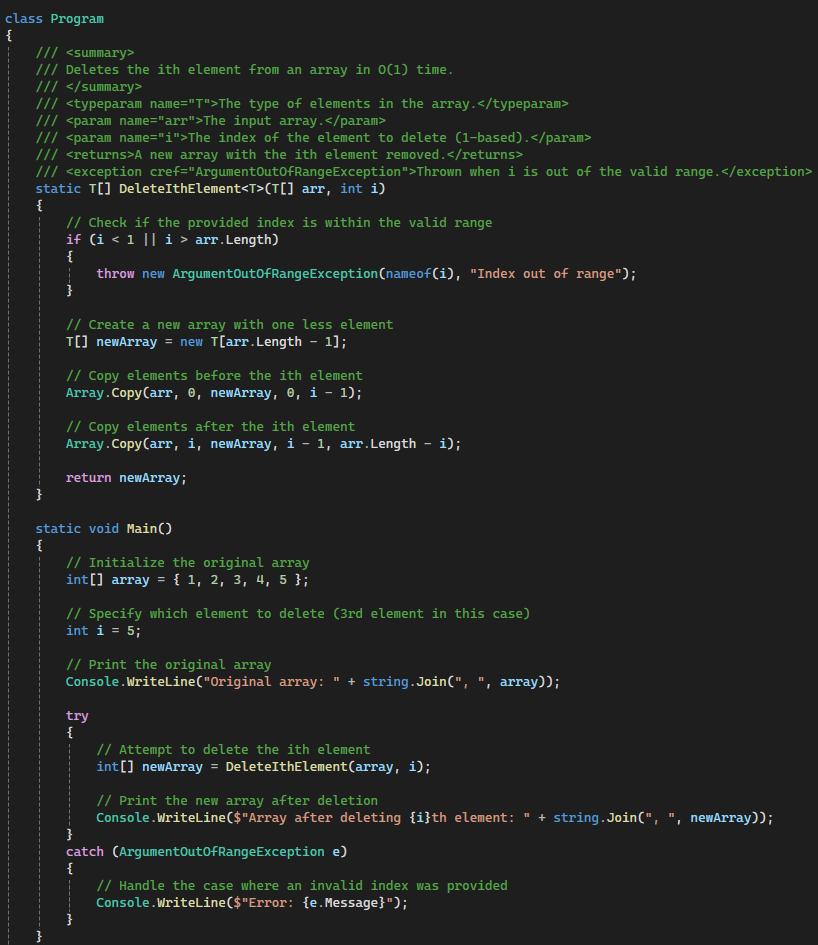
*Copy [1, 2] to newArray[0] and newArray[1].*

*Copy [4, 5] to newArray[2] and newArray[3].*

*5. Return newArray[ ] = [1, 2, 4, 5].*

**Program Screenshots**

*Screenshot 1/1*



**Program Output Screenshots**

*Screenshot 1/1*



**Exercise 1.4 - 1b**

**Delete ith Element from a sorted Array**

*Algorithm Explanation*

1. *Define the DeleteIthElementSorted method, which takes a sorted array arr[ ] and an index ‘i’ as input. It returns a new sorted array with the ith element removed.*
2. *First, check if the provided index ‘i’ is within the valid range (1 ≤ i ≤ arr.Length). If not, throw an ArguementOutOfRangeException.*
3. *Create a new array newArray[ ] with a length one less than the original array arr[ ].*
4. *Use Array.Copy to copy elements from arr[ ] to newArray[]:*

*Copy elements from index 0 to i-1 of arr[ ] to the same position in the newArray[ ].*

*Copy elements from index ‘i’ to the end of arr[ ] to positions starting from i-1 in new Array[ ].*

*5. Return the new Array[ ] containing all elements except the ith one, which remains sorted.*

**Delete ith Element from Sorted Array Applying [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] to the list, deleting the 5th element**

*Algorithm Explanation*

1. *Start with a sorted array arr[] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] and i = 5.*
2. *Check if i is valid (1 ≤ 5 ≤ 10), which it is.*
3. *Create newArray[] of length 9.*
4. *Copy elements:*

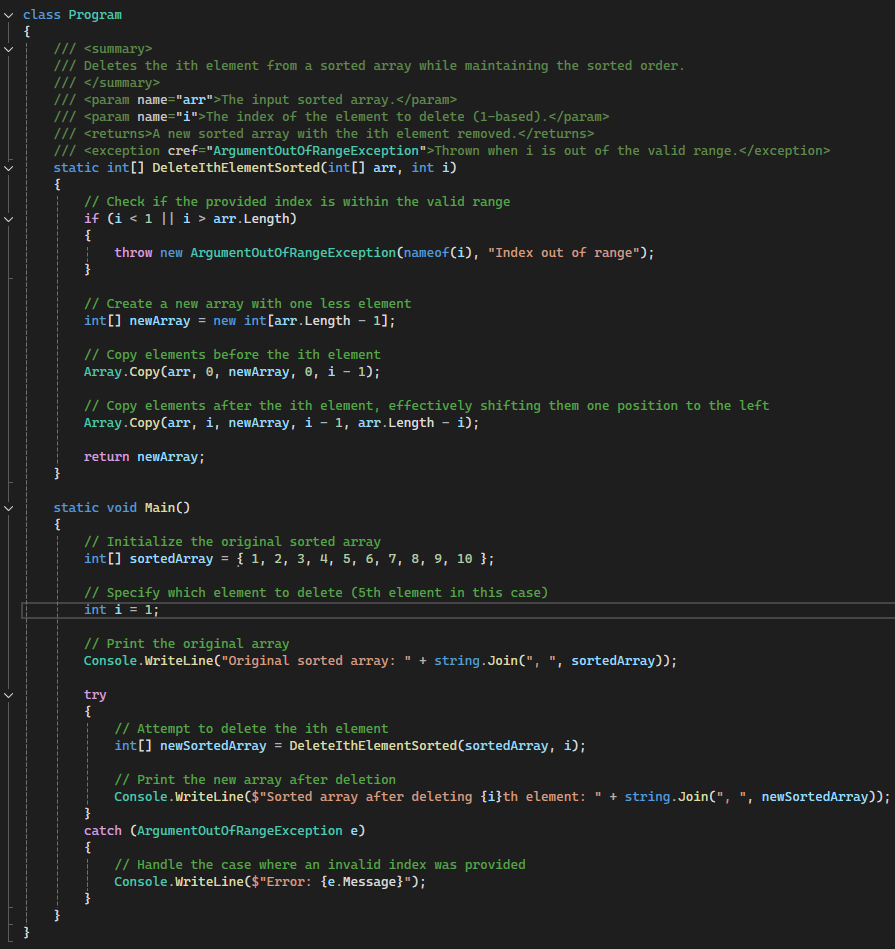
*Copy [1, 2, 3, 4] to newArray[0] through newArray[3].*

*Copy [6, 7, 8, 9, 10] to newArray[4] through newArray[8].*

1. *Return newArray[] = [1, 2, 3, 4, 6, 7, 8, 9, 10].*

**Program Screenshots**

*Screenshot 1/1*



**Program Output Screenshots**

*Screenshot 1/1*



**Exercise 1.4 - 10**

**Anagram Checker**

*Algorithm Explanation*

1. *Define the AreAnagrams method, which takes two strings word1 and word2 as input and returns a boolean indicating whether they are anagrams.*
2. *Covert both words to lowercase to ensure case-insensitive comparison.*
3. *Check if the lengths of the two words are equal. If not, return false as they cannot be anagrams.*
4. *Create a Dictionary<char, int> called charCount to store the count of each character in the first word.*
5. *Iterate through each character in word1:*

*If the character is already in charCount, increment its count.*

*If it's not in charCount, add it with a count of 1.*

1. *Iterate through each character in word2:*

*If the character is not in charCount, return false as its not in word1.*

*Decrement the count for this character in charCount.*

*If the count becomes negative, return false as word2 has more of this character than word1.*

*7. If both words return true, the words are anagrams.*

**Exercise 1.4 - 10**

**Anagram Checker Applying listen and silent**

*Algorithm Explanation*

1. *Start with word1 = “listen” and word2 = “silent” .*
2. *Convert to lowercase:*   
     
   *word1 = “listen” word2 =”silent”*
3. *Check lengths: Both are 6 characters, so continue.*
4. *Create charCount dictionary.*
5. *Count characters in “listen”:*   
   *‘l’ in charCount increment to 1*

*‘i’ in charCount increment to 1*  
*‘s’ in charCount increment to 1*

*‘t’ in charCount increment to 1*

*‘e’ in charCount increment to 1*

*‘n’ in charCount increment to 1*

1. *Check characters in “silent”:*   
   *‘s’ in charCount decrement to 0*

*‘i’ in charCount decrement to 0*  
*‘l’ in charCount decrement to 0*

*‘e’ in charCount decrement to 0*

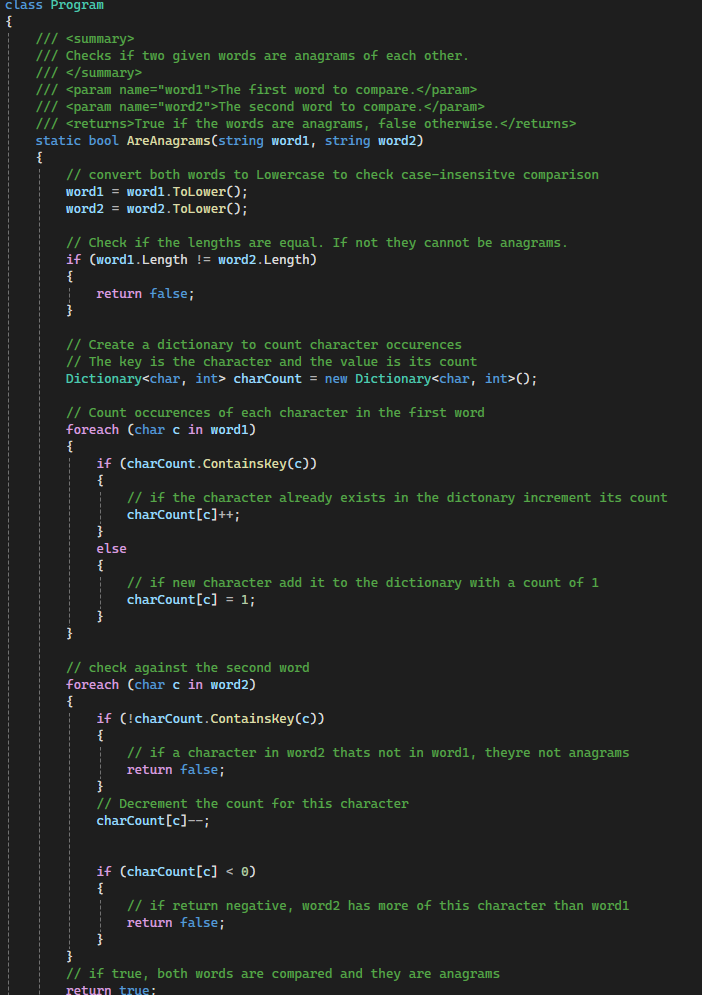
*‘n’ in charCount decrement to 0*

*‘t’ in charCount decrement to 0*

1. *All charcters checked without returning false, so return true.*

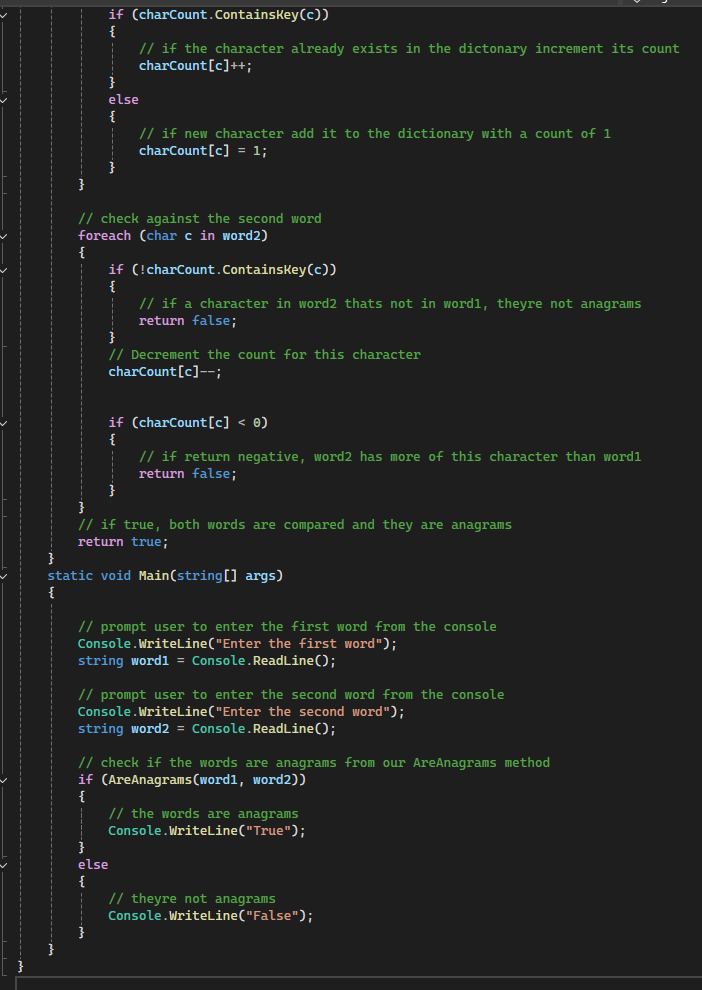
**Program Screenshots**

*Screenshot 1 / 2*



**Program Screenshots**

*Screenshot 2 / 2*



**Program Output Screenshots**

*Screenshot 1/1*

