**Question:1

Write a C# program that takes an array of strings as input, concatenates all the elements into a single string, extracts numeric characters from the concatenated string, and calculates the maximum, minimum, and difference between the extracted numeric values. If no numeric characters are found, return `0` for maximum, minimum, and difference.

Boilerplate Code:

```
using System;
using System.Collections.Generic;
using System.Linq;

class Program
{
    static void Main()
    {
        // User input for the array
        Console.WriteLine("Enter the elements of the array separated by spaces:");
        string[] inputArray = Console.ReadLine().Split(' ');

        // Step 1: Concatenate all elements of the array into a single string
        string concatenatedString = ""; // Complete this part
        Console.WriteLine($"Concatenated String: {concatenatedString}");

        // Step 2: Extract all numeric characters from the concatenated string
        List<int> extractedNumbers = new List<int>();
        foreach (char c in concatenatedString)
        {
            // Extract numeric characters and add them to the list
            // Complete this part
        }

        // Display extracted numbers
        Console.WriteLine("Extracted Numbers: [" + string.Join(", ", extractedNumbers) + "]");

        // Check if any numbers were extracted
        if (extractedNumbers.Count > 0)
        {
            // Step 3: Find the maximum and minimum numbers
            int maximumNumber = 0; // Complete this part
            int minimumNumber = 0; // Complete this part
            Console.WriteLine($"Maximum Number: {maximumNumber}");
            Console.WriteLine($"Minimum Number: {minimumNumber}");
            Console.WriteLine($"Difference: {maximumNumber - minimumNumber}");
        }
        else
        {
            // Handle the case where no numbers are found
            Console.WriteLine("Maximum Number: 0");
            Console.WriteLine("Minimum Number: 0");
            Console.WriteLine("Difference: 0");
        }
    }
}
```

Question :2

Write a C# program to perform the following operations on a given string:

1. **Check Substring:** Verify if a given substring exists in the main string.
2. **Replace Characters:** Replace all occurrences of a specified character in the string with another character.
3. **Swap Case:** Convert uppercase characters to lowercase and lowercase characters to uppercase in the string.
4. **Remove Whitespace:** Remove all whitespace characters from the string.
5. **Count Letters:** Count the frequency of each letter (ignoring case) in the string.

Input Details:

1. **Input 1:** The main string.
2. **Input 2:** The substring to check.

3. **Input 3:** The character to be replaced.
4. **Input 4:** The replacement character.

Boilerplate Code:

```
using System;
using System.Collections.Generic;
using System.Linq;

class Program
{
    static void Main()
    {
        // Input 1: Main String
        string mainString = GetInput("Enter the main string:");

        // Input 2: Substring
        string substring = GetInput("Enter the substring to check:");

        // Input 3: Character to be replaced
        char charToReplace = GetInput("Enter the character to replace:")[0];

        // Input 4: Character to replace with
        char replacementChar = GetInput("Enter the replacement character:")[0];

        bool substringExists = CheckSubstringExists(mainString, substring);
        string replacedString = ReplaceCharacter(mainString, charToReplace, replacementChar);
        string caseSwapped = SwapCase(mainString);
        string noSpaces = RemoveWhitespace(mainString);
        Dictionary<char, int> letterCount = CountLetters(mainString);

        Console.WriteLine($"Substring Exists: {(substringExists ? "Yes" : "No")}");
        Console.WriteLine($"Replaced: {replacedString}");
        Console.WriteLine($"Case Swapped: {caseSwapped}");
        Console.WriteLine($"No Spaces: {noSpaces}");
        Console.WriteLine($"Letter Count: {string.Join(", ", letterCount.Select(kvp => $"{kvp.Key}: {kvp.Value}"))}");
    }

    static string GetInput(string prompt)
    {
        Console.WriteLine(prompt);
        return Console.ReadLine();
    }

    static bool CheckSubstringExists(string main, string sub)
    {
        // Complete this function
        return false; // Replace with actual logic
    }

    static string ReplaceCharacter(string input, char oldChar, char newChar)
    {
        // Complete this function
        return ""; // Replace with actual logic
    }

    static string SwapCase(string input)
    {
        // Complete this function
        return ""; // Replace with actual logic
    }

    static string RemoveWhitespace(string input)
    {
        // Complete this function
        return ""; // Replace with actual logic
    }

    static Dictionary<char, int> CountLetters(string input)
    {
        // Complete this function
        return new Dictionary<char, int>(); // Replace with actual logic
    }
```

```
}
```

Requirements:

1. Implement the **CheckSubstringExists** function to return whether the substring exists in the main string.
2. Implement the **ReplaceCharacter** function to replace all occurrences of a specified character with another character.
3. Implement the **SwapCase** function to toggle the case of characters in the string.
4. Implement the **RemoveWhitespace** function to remove all whitespace characters from the string.
5. Implement the **CountLetters** function to count the frequency of each letter (case-insensitive).

Question 3:

Implement the missing logic for the given functions.

Boilerplate Code

```
using System;

public class ArrayOperations
{
    // Function to calculate the median of an array
    public static double CalculateMedian(int[] arr)
    {
        // Implement logic here
        return 0.0;
    }

    // Function to find the second largest element in an array
    public static int FindSecondLargest(int[] arr)
    {
        // Implement logic here
        return 0;
    }

    // Function to check if an array is a palindrome
    public static bool IsPalindrome(int[] arr)
    {
        // Implement logic here
        return false;
    }

    // Function to rotate the array to the left by a given number of steps
    public static int[] RotateLeft(int[] arr, int steps)
    {
        // Implement logic here
        return arr;
    }

    // Main method for testing
    public static void Main(string[] args)
    {
        int[] testArray = { 5, 1, 4, 2, 5 };

        Console.WriteLine("Median: " + CalculateMedian(testArray));
        Console.WriteLine("Second Largest: " + FindSecondLargest(testArray));
        Console.WriteLine("Is Palindrome: " + IsPalindrome(testArray));

        int[] rotatedArray = RotateLeft(testArray, 2);
        Console.WriteLine("Rotated Array: " + string.Join(", ", rotatedArray));
    }
}
```

Tasks to Implement:

1. **CalculateMedian**: Write logic to sort the array and calculate the median.
2. **FindSecondLargest**: Write logic to find the second largest element in the array.
3. **IsPalindrome**: Write logic to check if the array is the same when reversed.
4. **RotateLeft**: Write logic to rotate the array to the left by a specified number of steps.

Question 4 :

```
using System;

public class AdvancedArrayOperations
{
    // Function to find all unique elements in an array
    public static int[] FindUniqueElements(int[] arr)
    {
        // Implement logic here
        return new int[0];
    }

    // Function to find the intersection of two arrays
    public static int[] FindIntersection(int[] arr1, int[] arr2)
    {
        // Implement logic here
        return new int[0];
    }

    // Function to merge two arrays and remove duplicates
    public static int[] MergeAndRemoveDuplicates(int[] arr1, int[] arr2)
    {
        // Implement logic here
        return new int[0];
    }

    // Function to find the longest increasing subsequence in an array
    public static int[] LongestIncreasingSubsequence(int[] arr)
    {
        // Implement logic here
        return new int[0];
    }

    // Main method for testing
    public static void Main(string[] args)
    {
        int[] array1 = { 1, 2, 2, 3, 4, 5 };
        int[] array2 = { 2, 3, 6, 7, 5 };

        Console.WriteLine("Unique Elements: " + string.Join(", ", FindUniqueElements(array1)));
        Console.WriteLine("Intersection: " + string.Join(", ", FindIntersection(array1, array2)));
        Console.WriteLine("Merged Without Duplicates: " + string.Join(", ", MergeAndRemoveDuplicates(array1, array2)));
        Console.WriteLine("Longest Increasing Subsequence: " + string.Join(", ", LongestIncreasingSubsequence(array1)));
    }
}
```

Tasks to Implement:

1. **FindUniqueElements**: Write logic to extract all unique elements from an array.
2. **FindIntersection**: Write logic to find the common elements between two arrays.
3. **MergeAndRemoveDuplicates**: Write logic to merge two arrays into one, removing duplicate elements.
4. **LongestIncreasingSubsequence**: Write logic to find the longest sequence of increasing elements in the array.

Question 5 :

```
using System;

public class ArrayAdvancedChallenges
{
    // Function to find the majority element in an array (element appearing more than n/2 times)
    public static int? FindMajorityElement(int[] arr)
    {
        // Implement logic here
        return null;
    }

    // Function to find the smallest missing positive integer
    public static int FindSmallestMissingPositive(int[] arr)
    {
        // Implement logic here
        return -1;
```

```
    }

    // Function to find the kth largest element in the array
    public static int FindKthLargest(int[] arr, int k)
    {
        // Implement logic here
        return 0;
    }

    // Function to check if the array contains a duplicate
    public static bool ContainsDuplicate(int[] arr)
    {
        // Implement logic here
        return false;
    }

    // Main method for testing
    public static void Main(string[] args)
    {
        int[] testArray = { 3, 1, 2, 3, 4, 2, 1 };

        Console.WriteLine("Majority Element: " + FindMajorityElement(testArray));
        Console.WriteLine("Smallest Missing Positive: " + FindSmallestMissingPositive(testArray));
        Console.WriteLine("3rd Largest Element: " + FindKthLargest(testArray, 3));
        Console.WriteLine("Contains Duplicate: " + ContainsDuplicate(testArray));
    }
}
```

Tasks to Implement:

1. **FindMajorityElement**: Write logic to identify the majority element, if one exists, in the array.
2. **FindSmallestMissingPositive**: Write logic to find the smallest positive integer that is missing from the array.
3. **FindKthLargest**: Implement a function to find the kth largest element in the array.
4. **ContainsDuplicate**: Write logic to determine if the array contains any duplicate elements.

Question 6 :

```
using System;

public class StringFunctions
{
    // Function to reverse a string
    public static string ReverseString(string input)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to check if a string is a palindrome
    public static bool IsPalindrome(string input)
    {
        // Implement logic here
        return false;
    }

    // Function to count the frequency of each character in a string
    public static void CharacterFrequency(string input)
    {
        // Implement logic here
    }

    // Function to find the first non-repeating character in a string
    public static char? FirstNonRepeatingCharacter(string input)
    {
        // Implement logic here
        return null;
    }

    // Main method for testing
    public static void Main(string[] args)
    {
        string testString = "civic";
```

```
        Console.WriteLine("Reversed String: " + ReverseString(testString));
        Console.WriteLine("Is Palindrome: " + IsPalindrome(testString));

        Console.WriteLine("Character Frequency:");
        CharacterFrequency(testString);

        Console.WriteLine("First Non-Repeating Character: " + FirstNonRepeatingCharacter(testString));
    }
}
```

Tasks to Implement:

1. **ReverseString**: Write logic to reverse the input string.
2. **IsPalindrome**: Write logic to check if the input string is the same forwards and backwards.
3. **CharacterFrequency**: Implement logic to count and display the frequency of each character in the input string.
4. **FirstNonRepeatingCharacter**: Write logic to find the first character in the string that does not repeat.

Question 7 :

```
using System;

public class AdvancedStringFunctions
{
    // Function to find the longest substring without repeating characters
    public static string LongestUniqueSubstring(string input)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to check if two strings are anagrams
    public static bool AreAnagrams(string str1, string str2)
    {
        // Implement logic here
        return false;
    }

    // Function to capitalize the first letter of each word in a string
    public static string CapitalizeWords(string input)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to count the number of vowels and consonants in a string
    public static (int vowels, int consonants) CountVowelsAndConsonants(string input)
    {
        // Implement logic here
        return (0, 0);
    }

    // Main method for testing
    public static void Main(string[] args)
    {
        string testString = "programming";
        string testString2 = "margorp";

        Console.WriteLine("Longest Unique Substring: " + LongestUniqueSubstring(testString));
        Console.WriteLine("Are Anagrams: " + AreAnagrams(testString, testString2));
        Console.WriteLine("Capitalized Words: " + CapitalizeWords("hello world from csharp"));

        var counts = CountVowelsAndConsonants(testString);
        Console.WriteLine("Vowels: " + counts.vowels + ", Consonants: " + counts.consonants);
    }
}
```

Tasks to Implement:

1. **LongestUniqueSubstring**: Write logic to find the longest substring of the input string without repeating characters.

2. **AreAnagrams**: Write logic to check if two input strings are anagrams of each other.
3. **CapitalizeWords**: Implement logic to capitalize the first letter of each word in the input string.
4. **CountVowelsAndConsonants**: Write logic to count the number of vowels and consonants in the input string.

Question 8 :

```
using System;

public class StringModification
{
    // Function to insert a character at every nth position in a string
  public static string InsertAtEveryNthPosition(string input, char toInsert, int n)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to remove every occurrence of a specific character from a string
    public static string RemoveAllOccurrences(string input, char toRemove)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to replace the nth occurrence of a substring with another substring
    public static string ReplaceNthOccurrence(string input, string toReplace, string replacement, int n)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to modify a string by removing all characters after a specific index
    public static string RemoveAfterIndex(string input, int index)
    {
        // Implement logic here
        return string.Empty;
    }

    // Main method for testing
    public static void Main(string[] args)
    {
        string testString = "hello-world-hello-world";

        Console.WriteLine("Insert at Every 3rd Position: " + InsertAtEveryNthPosition(testString, '*', 3));
        Console.WriteLine("Remove All Occurrences of '-': " + RemoveAllOccurrences(testString, '-'));
        Console.WriteLine("Replace 2nd Occurrence of 'world': " + ReplaceNthOccurrence(testString, "world", "C#", 2));
        Console.WriteLine("Remove After Index 10: " + RemoveAfterIndex(testString, 10));
    }
}
```

Tasks to Implement:

1. **InsertAtEveryNthPosition**: Write logic to insert a specified character at every nth position in the input string.
2. **RemoveAllOccurrences**: Write logic to remove all occurrences of a specific character from the input string.
3. **ReplaceNthOccurrence**: Implement logic to replace the nth occurrence of a substring in the input string with another substring.
4. **RemoveAfterIndex**: Write logic to remove all characters from the string after a specified index

Question 9 :

```
using System;

public class AdvancedStringModification
{
    // Function to insert a substring at every position where a specific character is found
    public static string InsertAfterCharacter(string input, char target, string toInsert)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to remove the first n characters from a string
```

```
    public static string RemoveFirstNCharacters(string input, int n)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to replace all vowels in a string with a specified character
    public static string ReplaceVowels(string input, char replacement)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to reverse only the words in a sentence while preserving spaces
    public static string ReverseWords(string input)
    {
        // Implement logic here
        return string.Empty;
    }

    // Main method for testing
    public static void Main(string[] args)
    {
        string testString = "hello world this is C#";

        Console.WriteLine("Insert After Character 'o': " + InsertAfterCharacter(testString, 'o', "-inserted"));
        Console.WriteLine("Remove First 5 Characters: " + RemoveFirstNCharacters(testString, 5));
        Console.WriteLine("Replace Vowels with '*': " + ReplaceVowels(testString, '*'));
        Console.WriteLine("Reverse Words: " + ReverseWords(testString));
    }
}
```

Tasks to Implement:

1. **InsertAfterCharacter**: Write logic to insert a substring after every occurrence of a specified character in the input string.
2. **RemoveFirstNCharacters**: Write logic to remove the first `n` characters from the input string.
3. **ReplaceVowels**: Implement logic to replace all vowels in the input string with a specified character.
4. **ReverseWords**: Write logic to reverse only the words in a sentence, maintaining the order of spaces.

Question 10:

```
using System;

public class ComplexStringModification
{
    // Function to insert a sequence of characters in between each character of the string
    public static string InsertBetweenCharacters(string input, string toInsert)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to remove all duplicate characters from a string
    public static string RemoveDuplicates(string input)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to replace the last occurrence of a substring with another substring
    public static string ReplaceLastOccurrence(string input, string toReplace, string replacement)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to modify a string by keeping only unique words
    public static string KeepUniqueWords(string input)
    {
        // Implement logic here
        return string.Empty;
    }
```

```
    // Main method for testing
    public static void Main(string[] args)
    {
        string testString = "hello world hello universe";

        Console.WriteLine("Insert Between Characters: " + InsertBetweenCharacters(testString, "-"));
        Console.WriteLine("Remove Duplicates: " + RemoveDuplicates(testString));
        Console.WriteLine("Replace Last Occurrence of 'hello': " + ReplaceLastOccurrence(testString, "hello", "hi"));
        Console.WriteLine("Keep Unique Words: " + KeepUniqueWords(testString));
    }
}
```

Tasks to Implement:

1. **InsertBetweenCharacters**: Write logic to insert a given sequence of characters between each character of the input string.
2. **RemoveDuplicates**: Write logic to remove all duplicate characters from the input string while preserving the first occurrence.
3. **ReplaceLastOccurrence**: Implement logic to replace only the last occurrence of a specified substring in the input string.
4. **KeepUniqueWords**: Write logic to keep only the unique words in a sentence, removing all repeated words.

Question 11 :

```
using System;

public class ArrayOperationsAdvanced
{
    // Function to rotate an array to the right by a given number of steps
    public static int[] RotateRight(int[] arr, int steps)
    {
        // Implement logic here
        return arr;
    }

    // Function to find all triplets in an array that sum to a specific value
    public static void FindTripletsWithSum(int[] arr, int targetSum)
    {
        // Implement logic here
    }

    // Function to find the maximum product of three numbers in an array
    public static int MaxProductOfThree(int[] arr)
    {
        // Implement logic here
        return 0;
    }

    // Function to find the element that appears only once in an array where every other element appears twice
    public static int FindUniqueElement(int[] arr)
    {
        // Implement logic here
        return 0;
    }

    // Main method for testing
    public static void Main(string[] args)
    {
        int[] testArray = { 1, 2, 3, 4, 5, 6, 7 };
        int targetSum = 12;

        Console.WriteLine("Rotated Array (Right by 2): " + string.Join(", ", RotateRight(testArray, 2)));

        Console.WriteLine("Triplets with Sum " + targetSum + ":");
        FindTripletsWithSum(testArray, targetSum);

        Console.WriteLine("Max Product of Three: " + MaxProductOfThree(testArray));

        int[] uniqueArray = { 2, 2, 3, 4, 4 };
        Console.WriteLine("Unique Element: " + FindUniqueElement(uniqueArray));
    }
}
```

Tasks to Implement:

1. **RotateRight**: Write logic to rotate the array to the right by a given number of steps.
2. **FindTripletsWithSum**: Implement logic to find all unique triplets in the array that sum up to the given target value.
3. **MaxProductOfThree**: Write logic to calculate the maximum product of any three numbers in the array.
4. **FindUniqueElement**: Write logic to find the element that appears only once in an array where all other elements appear twice.

Question 12

```
using System;
using System.Text;

public class StringBuilderOperations
{
    // Function to reverse a string using StringBuilder
    public static string ReverseString(string input)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to remove all vowels from a string using StringBuilder
    public static string RemoveVowels(string input)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to append a specified character at the start and end of each word in a sentence
    public static string AppendToWords(string input, char toAppend)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to replace all occurrences of a specific word in a string with another word using StringBuilder
    public static string ReplaceWord(string input, string targetWord, string replacementWord)
    {
        // Implement logic here
        return string.Empty;
    }

    // Main method for testing
    public static void Main(string[] args)
    {
        string testString = "StringBuilder is powerful";

        Console.WriteLine("Reversed String: " + ReverseString(testString));
        Console.WriteLine("String Without Vowels: " + RemoveVowels(testString));
        Console.WriteLine("Appended to Words: " + AppendToWords(testString, '*'));
        Console.WriteLine("Replace 'powerful' with 'amazing': " + ReplaceWord(testString, "powerful", "amazing"));
    }
}
```

Tasks to Implement:

1. **ReverseString**: Use `StringBuilder` to reverse the input string.
2. **RemoveVowels**: Implement logic to remove all vowels (a, e, i, o, u) from the input string using `StringBuilder`.
3. **AppendToWords**: Use `StringBuilder` to append a specified character at the start and end of each word in a sentence.
4. **ReplaceWord**: Implement logic to replace all occurrences of a specific word in the input string with another word using `StringBuilder`.

Question 13

```
using System;
using System.Text;

public class StringBuilderOperations
{
    // Function to reverse every word in a sentence using StringBuilder
    public static string ReverseWords(StringBuilder input)
    {
```

```
        // Implement logic here
        return string.Empty;
    }

    // Function to remove all vowels from a string using StringBuilder
    public static string RemoveVowels(StringBuilder input)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to replace every nth character in a string with a specific character using StringBuilder
    public static string ReplaceEveryNthCharacter(StringBuilder input, char replacement, int n)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to generate a palindrome from a string using StringBuilder
    public static string GeneratePalindrome(StringBuilder input)
    {
        // Implement logic here
        return string.Empty;
    }

    // Main method for testing
    public static void Main(string[] args)
    {
        StringBuilder testInput = new StringBuilder("hello world from csharp");

        Console.WriteLine("Reversed Words: " + ReverseWords(testInput));
        Console.WriteLine("Without Vowels: " + RemoveVowels(testInput));
        Console.WriteLine("Replace Every 3rd Character with '*': " + ReplaceEveryNthCharacter(testInput, '*', 3));
        Console.WriteLine("Generated Palindrome: " + GeneratePalindrome(new StringBuilder("race")));
    }
}
```

Tasks to Implement:

1. **ReverseWords**: Write logic to reverse every word in the input sentence while keeping the words in order.
2. **RemoveVowels**: Implement logic to remove all vowels from the input string using `StringBuilder`.
3. **ReplaceEveryNthCharacter**: Write logic to replace every nth character in the input string with a specified character.
4. **GeneratePalindrome**: Implement logic to generate a palindrome by appending the reverse of the input string to itself.

Question 14 :

```
using System;
using System.Text;

public class AdvancedStringBuilderOperations
{
    // Function to insert a specific substring after every word in a sentence
    public static string InsertAfterEachWord(StringBuilder input, string toInsert)
    {
        // Implement logic here
        return string.Empty;
    }

    // Function to count the frequency of a specific character in the string
    public static int CountCharacterFrequency(StringBuilder input, char target)
    {
        // Implement logic here
        return 0;
    }

    // Function to replace all occurrences of a substring with another substring
    public static string ReplaceSubstring(StringBuilder input, string oldValue, string newValue)
    {
        // Implement logic here
        return string.Empty;
    }
```

```
    // Function to remove all spaces and compress the string using StringBuilder
    public static string CompressString(StringBuilder input)
    {
        // Implement logic here
        return string.Empty;
    }

    // Main method for testing
    public static void Main(string[] args)
    {
        StringBuilder testInput = new StringBuilder("hello world from csharp hello");

        Console.WriteLine("Insert After Each Word: " + InsertAfterEachWord(testInput, "-inserted"));
        Console.WriteLine("Frequency of 'o': " + CountCharacterFrequency(testInput, 'o'));
        Console.WriteLine("Replace 'hello' with 'hi': " + ReplaceSubstring(testInput, "hello", "hi"));
        Console.WriteLine("Compressed String: " + CompressString(testInput));
    }
}
```

Tasks to Implement:

1. **InsertAfterEachWord**: Write logic to insert a specific substring after each word in the sentence using `StringBuilder`.
2. **CountCharacterFrequency**: Implement logic to count the frequency of a specific character in the string using `StringBuilder`.
3. **ReplaceSubstring**: Write logic to replace all occurrences of one substring with another in the input using `StringBuilder`.
4. **CompressString**: Implement logic to remove all spaces from the string and return the compressed version using `StringBuilder`.

Question 15 :

```
using System;

public class StringFunctionOperations
{
    // Function to extract all words starting with a specific character using Split and Substring
    public static string[] ExtractWordsStartingWith(string input, char startingChar)
    {
        // Implement logic here
        return new string[0];
    }

    // Function to find all indices of a substring in a string
    public static int[] FindAllIndicesOfSubstring(string input, string substring)
    {
        // Implement logic here
        return new int[0];
    }

    // Function to check if all words in a sentence contain a specific character
    public static bool AllWordsContainCharacter(string input, char targetChar)
    {
        // Implement logic here
        return false;
    }

    // Function to replace all words containing a specific substring with another word
    public static string ReplaceWordsContainingSubstring(string input, string substring, string replacement)
    {
        // Implement logic here
        return string.Empty;
    }

    // Main method for testing
    public static void Main(string[] args)
    {
        string testString = "hello world from C# programming language";

        Console.WriteLine("Words Starting with 'h': " + string.Join(", ", ExtractWordsStartingWith(testString, 'h')));
        Console.WriteLine("Indices of 'world': " + string.Join(", ", FindAllIndicesOfSubstring(testString, "world")));
        Console.WriteLine("All Words Contain 'o': " + AllWordsContainCharacter(testString, 'o'));
        Console.WriteLine("Replace Words Containing 'pro': " + ReplaceWordsContainingSubstring(testString, "pro", "awesome"));
    }
}
```

Tasks to Implement:

1. **ExtractWordsStartingWith**: Write logic to split the input sentence into words and extract all words that start with a specific character using `Split` and `Substring`.
2. **FindAllIndicesOfSubstring**: Implement logic to find and return all starting indices of a given substring in the input string using `IndexOf`.
3. **AllWordsContainCharacter**: Write logic to check if every word in the input string contains a specific character using `Contains`.
4. **ReplaceWordsContainingSubstring**: Implement logic to replace all words containing a given substring with a specified replacement word.