**Find Intersection**

Have the function FindIntersection(**strArr**) read the array of strings stored in **strArr** which will contain 2 elements: the first element will represent a list of comma-separated numbers sorted in ascending order, the second element will represent a second list of comma-separated numbers (also sorted). Your goal is to return a comma-separated string containing the numbers that occur in elements of **strArr** in sorted order. If there is no intersection, return the string **false**.

**Examples**

Input: new string[] {"1, 3, 4, 7, 13", "1, 2, 4, 13, 15"}  
Output: 1,4,13

Code:

using System;

class MainClass {

  public static string FindIntersection(string[] strArr) {

    // Extract the first and second strings from the input array

    string string1 = strArr[0];

    string string2 = strArr[1];

    // Split each string by commas to create arrays of string numbers

    string[] char1 = string1.Split(',');

    string[] char2 = string2.Split(',');

    // Initialize an empty string to build the result

    string returnString = "";

    // Iterate through each number in the first array

    for(int i = 0; i < char1.Length; i++) {

      bool hasIntersection = false;

      int c;

      // Convert the current element of char1 to an integer

      int.TryParse(char1[i], out c);

      // Check if this number exists in the second array

      for(int j = 0; j < char2.Length; j++) {

        int c2;

        // Convert the current element of char2 to an integer

        int.TryParse(char2[j], out c2);

        // If the numbers match, there is an intersection

        if(c == c2) {

          hasIntersection = true;

          break; // No need to check further in char2

        }

      }

      // If an intersection was found, add the number to the result string

      if(hasIntersection) {

        // If returnString is empty, just add the number

        if(returnString == "") {

          returnString += c.ToString();

        } else {

          // Otherwise, append a comma followed by the number

          returnString += "," + c.ToString();

        }

      }

    }

    // If no intersection was found, set returnString to "false"

    if(returnString == "") {

      returnString = "false";

    }

    // Return the final result string

    return returnString;

  }

  static void Main() {

    // Read input from the console and pass it to FindIntersection

    Console.WriteLine(FindIntersection(Console.ReadLine()));

  }

}

**Questions Marks**

Have the function QuestionsMarks(**str**) take the **str** string parameter, which will contain single digit numbers, letters, and question marks, and check if there are exactly 3 question marks between every pair of two numbers that add up to 10. If so, then your program should return the string **true**, otherwise it should return the string **false**. If there aren't any two numbers that add up to 10 in the string, then your program should return **false** as well.  
  
For example: if **str** is "arrb6???4xxbl5???eee5" then your program should return **true** because there are exactly 3 question marks between 6 and 4, and 3 question marks between 5 and 5 at the end of the string.

using System;

class MainClass {

  public static string QuestionsMarks(string str) {

    bool foundPair = false;

    for(int i = 0; i < str.Length; i++) {

      if(Char.IsDigit(str[i])) {

        int num1 = int.Parse(str[i].ToString());

        for(int j = i + 1; j < str.Length; j++) {

          if(Char.IsDigit(str[j])) {

            int num2 = int.Parse(str[j].ToString());

            if(num1 + num2 == 10) {

              foundPair = true;

              string between = str.Substring(i + 1, j - i - 1);

              int questionMarkCount = 0;

              foreach(char c in between) {

                if(c == '?') {

                  questionMarkCount++;

                }

              }

              if(questionMarkCount != 3) {

                return "false";

              }

            }

          }

        }

      }

    }

    return foundPair ? "true" : "false";

  }

  static void Main() {

    // keep this function call here

    Console.WriteLine(QuestionsMarks(Console.ReadLine()));

  }

}

**First Reverse**

Have the function FirstReverse(**str**) take the **str** parameter being passed and return the string in reversed order. For example: if the input string is "Hello World and Coders" then your program should return the string **sredoC dna dlroW olleH**.

Code:

using System;

class MainClass {

  public static string FirstReverse(string str) {

    // code goes here

    int size=str.Length;

    string reversedstring="";

    for(int i=size-1;i>=0;i--)

    {

       reversedstring+=str[i];

    }

    return reversedstring;

  }

  static void Main() {

    // keep this function call here

    Console.WriteLine(FirstReverse(Console.ReadLine()));

  }

}

**First Factorial**

Have the function FirstFactorial(**num**) take the **num** parameter being passed and return the factorial of it. For example: if **num** = 4, then your program should return **(4 \* 3 \* 2 \* 1)** = 24. For the test cases, the range will be between 1 and 18 and the input will always be an integer.

Code:

using System;

class MainClass {

  public static int FirstFactorial(int num) {

    // code goes here

    int factorial=1;

    for(int i=1;i<=num;i++)

{

  factorial\*=i;

}

    return factorial;

  }

  static void Main() {

    // keep this function call here

    Console.WriteLine(FirstFactorial(Console.ReadLine()));

  }

}

**Longest Word**

Have the function LongestWord(**sen**) take the **sen** parameter being passed and return the longest word in the string. If there are two or more words that are the same length, return the first word from the string with that length. Ignore punctuation and assume **sen** will not be empty. Words may also contain numbers, for example "Hello world123 567"

Code:

using System;

using System.Text.RegularExpressions;

class MainClass {

  public static string LongestWord(string sen) {

    // Split sentence into words

    string[] words = sen.Split();

    int maxCount = 0;

    string maxWord = "";

    foreach (string word in words) {

      int count = 0;

      foreach (char c in word) {

        if (IsEnglishLetter(c)) {

          count++;

        }

      }

      if (count > maxCount) {

        maxCount = count;

        maxWord = word;

      }

    }

    return maxWord;

  }

  static bool IsEnglishLetter(char c) {

    return (c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z');

  }

  static void Main() {

    // keep this function call here

    Console.WriteLine(LongestWord(Console.ReadLine()));

  }

}