Задание.

В заданной строке записаны три слова. Возможно ли, переставляя и удаляя буквы ПЕРВОГО слова и добавляя буквы ВТОРОГО слова, получить ТРЕТЬЕ слово.

Решение.

```
using System;

namespace Words
{
    class Program
    {
        // Функция определяет, является ли символ с разделителем
        static bool isSeparator(char c)
        {
            return c == ' ';
        }

        // Функция находит слово в строке s, начиная с позиции from,
        // при этом продвигая указатель from за конец слова
        static String nextWord(String s, ref int from)
        {
            // Сначала пропускаем все разделители
            while (from < s.Length && isSeparator(s[from]))
                from++;
            // Теперь from указывает на первый символ слова, запомним его
            int start = from;
            // Пропускаем все символы, не являющиеся разделителями
            while (from < s.Length && !isSeparator(s[from]))
                from++;
            // Искомое слово будет между текущим положением from и запомненным
            // началом слова start. Возвращаем соответствующую подстроку
            return s.Substring(start, from - start);
        }

        // Функция сортирует массив chars методом выбора
        static void sortChars(char[] chars)
        {
            // В цикле проходим по массиву. На место каждого очередного элемента
            // будем записывать минимальный элемент из оставшегося за этим элементом подмассива.
            for (int i = 0; i < chars.Length - 1; i++)
            {
                // Сначала минимальный - сам элемент
                int idx = i;
                // В цикле по оставшемуся подмассиву ищем минимум
                for (int j = i + 1; j < chars.Length; j++)
                    // Если очередной элемент меньше текущего минимума
                    if (chars[j] < chars[idx])
                        // Обновим индекс минимума
                        idx = j;
                // Если индекс минимального элемента не равен текущему элементу массива,
                // обменяем их местами. Таким образом, в текущем элементе окажется минимум
                if (idx != i)
                {
                    char t = chars[i];
                    chars[i] = chars[idx];
```

```
                chars[idx] = t;
            }
        }
    }

    // Функция проверяет, можно ли составить третье слово, путем перестановки и удаления
    // букв первого слова и добавления букв второго слова. Фактически это означает, что
    // мы берем часть букв первого слова, переставляя их в произвольном порядке, и часть
    // букв второго слова, также добавляя их в произвольном порядке, т.е. задача сводится
    // к определению того, можно ли составить третье слово из букв первого и второго слова.
    static bool canCombine(String w1, String w2, String w3)
    {
        // В массиве u будем хранить буквы первого и второго слова
        char[] u = (w1 + w2).ToCharArray();
        // В массиве s будем хранить буквы третьего слова
        char[] s = w3.ToCharArray();
        // Отсортируем оба массива
        sortChars(u);
        sortChars(s);
        // Идея алгоритма следующая: в цикле пойдем по буквам массива третьего слова (s).
        // Для каждой буквы нам надо найти такую же букву в массиве u, после чего мы
        // "вычеркиваем" обе буквы. Результат успешный, если удалось найти "пару" для
        // всех букв третьего слова

        // Указатель в массиве букв первых двух слов
        int uIdx = 0;
        // Указатель в массиве букв третьего слова
        int sIdx = 0;
        // В цикле по буквам первого слова проводим проверку
        while (sIdx < s.Length)
        {
            // Сначала пропускаем все буквы первых двух слов, которые меньше текущей
            // буквы третьего слова, так как они гарантированно не встретятся нам далее
            // в связи с тем, что массив букв третьего слова отсортирован, и далее все
            // его буквы будут >= текущей
            while (uIdx < u.Length && u[uIdx] < s[sIdx])
                uIdx++;
            // Теперь проверяем, если текущая буква из первых двух слов равна
            // рассматриваемой букве третьего слова, мы нашли "пару"
            if (uIdx < u.Length && u[uIdx] == s[sIdx])
            {
                // Помечаем, что буква из первых двух слов дальше не используется,
                // продвигая указатель
                uIdx++;
                // Аналогично продвигаем указатель в массиве букв третьего слова
                sIdx++;
            }
            else
                // Если пара не найдена, то текущая буква из первых двух слов больше
                // буквы третьего слова. В связи с тем, что массив букв отсортирован,
                // далее все буквы тоже будут больше, т.е. для текущей буквы из третьего
                // слово невозможно найти "пару". Соответственно, прерываем цикл.
                break;
        }
        // Результат успешен, если смогли "пройти" все буквы третьего слова, т.е. для
        // каждой нашли "пару" из первых двух
        return sIdx == s.Length;
    }

    static void Main(string[] args)
```

```
        {
            // Вводим исходную строку
            Console.WriteLine("Введите строку, содержащую минимум 3 слова, разделенные пробелами:
");
            String s = Console.ReadLine();
            // Выбираем слова из строки
            int p = 0;
            String w1 = nextWord(s, ref p);
            String w2 = nextWord(s, ref p);
            String w3 = nextWord(s, ref p);
            // Проверяем, что слов как минимум 3, т.е. третье слово не пустое
            if (w3.Length == 0)
            {
                Console.WriteLine("Строка содержит менее 3 слов!");
                return;
            }
            // Выводим ответ
            Console.WriteLine("Третье слово " + (canCombine(w1, w2, w3) ? "" : "НЕ ") + "может быть
составлено из первых двух");
        }
    }
}
```