**Data Structures: Solving Problems with Arrays, Sets and Maps**

**Instructions**

In this challenge you will solve several problems that may use Sets, Maps and Arrays to facilitate the solution.

**Existing Files**

| **File path** | **Description** |
| --- | --- |
| src/anagram.js | Implement your solution to the anagram problem in this file. |
| src/anagramGroup.js | Implement your solution to the anagram groups problem in this file. |
| src/permutationPalindrome.js | Implement your solution to the permutation palindrome problem in this file. |
| src/firstSingleCharacter.js | Implement your solution to the first single character problem in this file. |
| \_\_tests\_\_/anagram.test.js | Tests for the anagram problem. You do not need to edit this file. |
| \_\_tests\_\_/anagramGroups.test.js | Tests for the anagram groups problem. You do not need to edit this file. |
| \_\_tests\_\_/permutationPalindrome.test.js | Tests for the permutation palindrome problem. You do not need to edit this file. |
| \_\_tests\_\_/firstSingleCharacter.test.js | Tests for the permutation palindrome problem. You do not need to edit this file. |

**Tasks**

**Anagram**

Given two strings "s1" and "s2", determine if the two strings are anagrams of each other or not.

Two words are anagrams of each other if they are both spelled with the letters. For example, the words "race" and "care" are anagrams.

Write a function that returns true if the two strings are anagrams of each other, false otherwise.

Write your solution in the file named src/anagram.js.

**Anagram Groups**

Given a list of words, group the words into anagram groups. That is, place all words in the list that are anagrams of each other into a group.

For example, if the input was ['east', 'cars', 'acre', 'arcs', 'teas', 'eats', 'race'], the output should be: [['east', 'teas', 'eats'], ['cars', 'arcs'], ['acre', 'race']].

Write your solution to this problem in the file named src/anagramGroups.js.

**Permutation Palindrome**

Write an algorithm to check whether any anagram of some string is a palindrome. Given some string, "acecarr", the algorithm should return true, because the letters in "acecarr" can be rearranged to the anagram "racecar", which itself is a palindrome. In contrast, given the word "north", the algorithm should return false, because there's no anagram for "north" that would be a palindrome.

Write your solution to this problem in the file named src/permutationPalindrome.js.

**First Single Character**

Given a string, find the first character in that string that occurs only once. For example, given the string "thinkful" return the character "t" since that is the first character in the string that is not repeated in the string.

Given the string "character" return "h" because the first character "c" is repeated later in the string so the first non-repeating character is "h".

If no single characters are found return null.

Write your solution to this problem in the file named src/firstSingleCharacter.js.