* **findSubstring** method takes in a string **s** and an array of strings **words**.
* The purpose of this method is to find all starting indices of substring(s) within the input string **s** that is a concatenation of each word in the **words** array, where order matters.
* It starts by initializing several variables:
  + **wordLength** is set to the length of the first string in the **words** array.
  + **totalWordsLength** is set to the length of the concatenation of all strings in the **words** array.
  + **hash** is a new map that maps each string in the **words** array to a unique integer.
  + **ans** is a list of integers that will eventually store the starting indices of the substrings.
  + **char[] s2** is a char array that stores the characters of the string s.
* Then it maps each string in the words array to a unique integer in the **hash** and create an array **count** to count the frequency of each word in words.
* Then, there are two nested loops:
  + The outer loop iterates **i** from 0 to **wordLength**, and the inner loop iterates **j** from **i** to **s.length() - totalWordsLength** with a step size of **wordLength**.
  + For each **j**, the innermost loop iterates **k** from **j + totalWordsLength - wordLength** to **j**.
  + For each **k**, the code creates a new string, **str**, by using the **String** constructor and passing in **s2**, **k**, and **wordLength** as arguments.
  + Then it check if the frequency of the word in words array is more than the frequency of the word in the substring we are checking, if frequency is greater, then continue to next word otherwise break the innermost loop.
  + If all the word are counted correctly, innermost loop completes without breaking and **j** is added to **ans**
* Finally the ans list is returned as output
* it's important to note that this implementation assumes that all words in the input **words** array have the same length and that **words** contains at least one element.