2. String Anagram

An anagram of a string is another string with the same characters in the same frequency, in any order. For example *'abc', 'bca', 'acb', 'bac', 'cba', 'cab'* are all anagrams of the string *'abc'*. Given two arrays of strings, for every string in one list, determine how many anagrams of it are in the other list. Write a function that receives *dictionary* and *query*, two string arrays. It should return an array of integers where each element *i* contains the number of anagrams of *query[i]* that exist in *dictionary*.

Example
dictionary = ['hack', 'a', 'rank', 'khac', 'ackh', 'kran', 'rankhacker', 'a', 'ab', 'ba', 'stairs', 'raits']
query = ["a", "nark", "bs", "hack", "stair"]

query[0] = 'a' has *2* anagrams in *dictionary*: *'a'* and *'a'*.
query[1] = 'nark' has *2* anagrams in *dictionary*: *'rank'* and *'kran'*.
query[2] = 'bs' has *0* anagrams in *dictionary*.
query[3] = 'hack' has *3* anagrams in *dictionary*: *'hack'*, *'khac'* and *'ackh'*.
query[4] = 'stair' has *1* anagram in *dictionary*: *'raits'*. While the characters are the same in *'stairs'*, the frequency of *'s'* differs, so it is not an anagram.
The final answer is *[2, 2, 0, 3, 1]*.

Function Description
Complete the function *stringAnagram* in the editor below.

stringAnagram has the following parameters:
string dictionary[n]: an array of strings to search in
string query[q]: an array of strings to search for

Returns
int[q]: an array of integers where the i^{th} value is the answer to *query[i]*

Constraints

- $1 \le length(dictionary), length(query) \le 10^5$
- $1 \le length(dictionary[i]) \le 15$
- $1 \le length(query[i]) \le 15$
- *Every string consists of lowercase English letters.*

▼ Input Format For Custom Testing

The first line of input contains an integer, *n*, the number of strings in *dictionary[]*.
Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains a string, *dictionary[i]*.
The next line contains an integer, *q*, the number of strings in *query[]*.
Each line *i* of the *q* subsequent lines (where $0 \le i < q$) contains a string, *query[i]*.

▼ Sample Case 0

Sample Input

```
STDIN       Function
-----       -----
5       →   dictionary[] size n = 5
heater  →   dictionary = ['heater', 'cold', 'clod', 'reheat', 'docl']
cold
clod
reheat
docl
3       →   query[] size q = 3
codl    →   query = ['codl', 'heater', 'abcd']
heater
abcd
```

Sample Output

```
3
2
0
```

Explanation
query[0] = 'codl' has *3* anagrams in *dictionary*: *'cold'*, *'clod'* and *'docl'*.
query[1] = 'heater' has *2* anagrams in *dictionary*: *'heater'* and *'reheat'*.
query[2] = 'abcd' has *0* anagrams in *dictionary*.
The final answer is *[3, 2, 0]*.

▼ Sample Case 1

Sample Input

```
STDIN       Function
-----       -----
8       →   dictionary[] size n = 8
listen  →   dictionary = ['listen', 'tow', 'silent', 'lisent', 'two', 'abc', 'no', 'on']
tow
silent
lisent
two
abc
no
on
4       →   query[] size q = 4
two     →   query = ['two', 'bca', 'no', 'listen']
bca
no
listen
```

Sample Output

```
2
1
2
3
```

Explanation

query[0] = 'two' has *2* anagrams in *dictionary*: *'tow' and 'two'*.
query[1] = 'bca' has *1* anagram in *dictionary*: *'abc'*.
query[2] = 'no' has *2* anagrams in *dictionary*: *'no' and 'on'*.
query[3] = 'listen' has *3* anagrams in *dictionary*: *'listen', 'silent' and 'lisent'*.
The final answer is *[2, 1, 2, 3]*.