**Autocorrect Prototype**Complete the implementation of an autocorrect function. Given a search query string, the function should return all words which are anagrams.Given 2 arrays, words[n], and queries[q], for each query, return an array of the strings that are anagrams, sorted alphabetically ascending.Note: An anagram is any string that can be formed by rearranging the letters of a stringExample n-4q-2words-["duel", "speed", "dule", "cars"],queries-["spede", "deul"] The only anagram of "spede" is "speed".Both "duel" and "dule" are anagrams of "deul".Return [["speed"], ["duel", "dule"]].

**Function Description**

Complete the function getSearchResults in the editor below.getSearchResults takes the following arguments: string words[n]: the list of words to search string queries(q): the words to search for list.Returnsstring[qil: the results for each search queryConstraints•1sn, q$5000• 1 s length of words length of queries[i] ≤ 100

It is guaranteed that each query word has at least one anagram in the words▾ Input Format For Custom TestingThe first line contains an integer, n the number of elements in words. Each of the next in lines contains a string, words!). The next line contains an integer, q, the number of elements in queries.Each of the next in lines contains a string queries[i]

Sample Case 0Sample Input For Custom Testing

STDIN FUNCTION

----- --------

6 → words[] size n = 6

allot → words = ["allot", "cat", "peach", "dusty", "act", "cheap"]

cat

peach

dusty

act

cheap

3 → queries[] size q = 3

tac → queries = ["tac", "study", "peahc"]

study

peahc

Sample Output

act cat

dusty

cheap peach

Explanation

The anagrams of "tac" in alphabetical order are "act" and "cat". The only anagram of "study" is "dusty". The anagrams of "peahc" are "cheap" and "peach".

Sample Case 1

Sample Input For Custom Testing

STDIN FUNCTION

----- --------

5 → words[] size n = 5

emits → words = ["emits", "items", "baker", "times", "break"]

items

baker

times

break

2 → queries[] size q = 2

mites → queries = ["mites", "brake"]

brake

Sample Output

emits items times

baker break

Explanation

The anagrams of "mites" are "emits", "items", and "times". The anagrams of "brake" are "baker" and "break".

**HACKERRANK SOLUTION**

*#!/bin/python3*

**import** math

**import** os

**import** random

**import** re

**import** sys

*#*

*# Complete the 'getSearchResults' function below.*

*#*

*# The function is expected to return a 2D\_STRING\_ARRAY.*

*# The function accepts following parameters:*

*#  1. STRING\_ARRAY words*

*#  2. STRING\_ARRAY queries*

*#*

**from** collections **import** defaultdict

**def** group(words):

    groups = defaultdict(**list**)

**for** word **in** words:

        sorted\_word = ''.join(**sorted**(word))

        groups[sorted\_word].append(word)

**return** groups

**def** getSearchResults(words, queries):

    groups = group(words)

    result = []

**for** query **in** queries:

        sorted\_query = ''.join(**sorted**(query))

**if** sorted\_query **in** groups:

            value = **sorted**(groups[sorted\_query])

            result.append(value)

**return** result

**if** **\_\_name\_\_** == '\_\_main\_\_':

    fptr = **open**(os.environ['OUTPUT\_PATH'], 'w')

    words\_count = **int**(**input**().strip())

    words = []

**for** \_ **in** **range**(words\_count):

        words\_item = **input**()

        words.append(words\_item)

    queries\_count = **int**(**input**().strip())

    queries = []

**for** \_ **in** **range**(queries\_count):

        queries\_item = **input**()

        queries.append(queries\_item)

    result = getSearchResults(words, queries)

    fptr.write('\n'.join([' '.join(x) **for** x **in** result]))

    fptr.write('\n')

    fptr.close()

**RUN CODE IN VS CODE**

from collections import defaultdict

def group\_anagrams(words):

    anagram\_groups = defaultdict(list)

    for word in words:

        sorted\_word = ''.join(sorted(word))

        anagram\_groups[sorted\_word].append(word)

    return anagram\_groups

def getSearchResults(words, queries):

    anagram\_groups = group\_anagrams(words)

    result = []

    for query in queries:

        sorted\_query = ''.join(sorted(query))

        if sorted\_query in anagram\_groups:

            anagrams = sorted(anagram\_groups[sorted\_query])

            result.append(anagrams)

    return result

# Example usage:

n = 6

words = ["allot", "cat", "peach", "dusty", "act", "cheap"]

q = 3

queries = ["tac", "study", "peahc"]

results = getSearchResults(words, queries)

for r in results:

    print(r)

**Longest Subarray**

A subarray of array a is defined as a contiguous block of al's elements having a length that is less than or equal to the length of the array. For example, the subarrays of array a-[1, 2, 3] are [1] [2] [3] [1, 2], [2, 3], and [1, 2, 3]. Given an integer, k-3, the subarrays having elements that sum to a number skare [1] and [1, 2]. The longest of these subarrays is [1, 2], which has a length of 2 Given an array, a, determine its longest subarray that sums to less than or equal to a given value k

**Function Description**

Complete the function maxLength in the editor below. The function must return an integer that represents the length of the longest subarray of athat sums to a number sk

maxLength has the following parameter(s): ala[0] a[n-1]] an array of integers kan integer

**Constraints**

1sns 105

• 1 sails 103

•1sks 109

Input Format For Custom Testing Input from stdin will be processed as follows and passed to the function.

The first line contains a single integer, n, that denotes the number of elements array a Each line / of the n subsequent lines (where Osi<n) contains an integer that describes element a[i].

The last line contains an integer, k

**Sample Case 0**

Sample Input For Custom Testing Sample Input 0

3

1

2

3

4

Sample Output 0

2

Explanation 0

The subarrays of [1, 2, 3] having elements that sum to a numbers (k<=4) are [1] [2] [3], and [1, 2]. The longest of these is [1, 2], which has a length of 2 Return 2 as the answer.

**Sample Case 1**

Sample Input For Custom Testing Sample Input 1

4

3

1

2

1

4

Sample Output 1

3

Explanation 1

The subarrays of [3, 1, 2, 1] having elements that sum to a numbers (k<=4) are [3], [1], [2], [1], (3, 1), (1, 2) [2, 1], and [1, 2, 1). The longest of these is [1, 2, 1], which has a length of 3. Return 3 as the answer.

**HACKERRANK SOLUTION**

*#!/bin/python3*

**import** math

**import** os

**import** random

**import** re

**import** sys

*#*

*# Complete the 'maxLength' function below.*

*#*

*# The function is expected to return an INTEGER.*

*# The function accepts following parameters:*

*#  1. INTEGER\_ARRAY a*

*#  2. INTEGER k*

*#*

**def** maxLength(a, k):

    max\_len = 0

    sum1 = 0

    l = 0

**for** r **in** **range**(**len**(a)):

        sum1 += a[r]

**while** sum1 > k:

            sum1 -= a[l]

            l += 1

        max\_len = **max**(max\_len, r - l + 1)

**return** max\_len

**if** **\_\_name\_\_** == '\_\_main\_\_':

    fptr = **open**(os.environ['OUTPUT\_PATH'], 'w')

    a\_count = **int**(**input**().strip())

    a = []

**for** \_ **in** **range**(a\_count):

        a\_item = **int**(**input**().strip())

        a.append(a\_item)

    k = **int**(**input**().strip())

    result = maxLength(a, k)

    fptr.write(**str**(result) + '\n')

    fptr.close()

**RUN CODE IN VS CODE**

def maxLength(a, k):

    max\_length = 0

    current\_sum = 0

    left = 0

    for right in range(len(a)):

        current\_sum += a[right]

        while current\_sum > k:

            current\_sum -= a[left]

            left += 1

        max\_length = max(max\_length, right - left + 1)

    return max\_length

# Sample input

n = 3

a = [1, 2, 3]

k = 4

result = maxLength(a, k)

print(result)  # Output: 2