**Competitive Programming Week-2 Exam**

1. **Unique morse code words:** International Morse Code defines a standard encoding where each letter is mapped to a series of dots and dashes, as follows:

"a" maps to ".-"

"b" maps to "-..."

"c" maps to "-.-.", and so on.

For convenience, the full table for the 26 letters of the English alphabet is given below:

[

A ->".-",

B->"-...",

C ->"-.-.",

D ->"-..",

E ->".",

F ->"..-.",

G ->"--.",

H ->"....",

I ->"..",

J ->".---",

K ->"-.-",

L ->".-..",

M ->"--",

N ->"-.",  
O ->"---",

P ->".--.",

Q ->"--.-",

R ->".-.",

S ->"...",

T ->"-",

U ->"..-",

V ->"...-",

W ->".--",

X ->"-..-",

Y ->"-.--",

Z ->"--.." ]

Now, given a list of words, each word can be written as a concatenation of the Morse code of each letter. For example, "cab" can be written as "-.-.-....-", (which is the concatenation "-.-." + "-..." + ".-"). We'll call such a concatenation, the transformation of a word.

Return the number of different transformations among all words we have.

Example:

**Input:** words = ["gin", "zen", "gig", "msg"]

**Output:** 2

**Explanation:**

The transformation of each word is:

"gin" -> "--...-."

"zen" -> "--...-."

"gig" -> "--...--."

"msg" -> "--...--."

There are 2 different transformations, "--...-." and "--...--.".

Test case 1:

Input: words = ["gin", "zen", "gig", "msg"]

Output: 2

Test case 2:

Input: words = ["a", "z", "g", "m"]

Output: 4

Test case 3:

Input: words = ["bhi", "vsv", "sgh", "vbi"]

Output: 3

Test case 4:

Input: words = ["a", "b", "c", "d"]

Output: 4

Test case 5:

Input: words = ["hig", "sip", "pot"]

Output: 2

2. **Queue reconstruction by height:** Suppose you have a random list of people standing in a queue. Each person is described by a pair of integers (h, k), where h is the height of the person and k is the number of people in front of this person who have a height greater than or equal to h. Write an algorithm to reconstruct the queue.

Note: The number of people is less than 1,100.

Test case 1:

Input:

[[7,0], [4,4], [7,1], [5,0], [6,1], [5,2]]

Output:

[[5,0], [7,0], [5,2], [6,1], [4,4], [7,1]]

Test case 2:

Input:

[[12,0],[6,3],[3,4],[9,2], [11,1],[1,5]]

Output:

[[12,0],[11,1],[9,2],[6,3],[3,4],[1,5]]

Test case 3:

Input:

[ [2,4], [5,1], [3,3], [1,5], [4,2], [6,0]]

Output:

[[6,0], [5,1], [4,2], [3,3], [2,4], [1,5]]

3. **Best meeting point:** A group of two or more people wants to meet and minimize the total travel distance. You are given a 2D grid of values 0 or 1, where each 1 marks the home of someone in the group. Find the total distance that needs to be traveled to reach the best meeting point (Total distance traveled is minimum).

Test Case 1:

Input: grid [][] = {{1, 0, 0, 0, 1},

{0, 0, 0, 0, 0},

{0, 0, 1, 0, 0}};

Output: 6

Test Case 2:

Input: grid [3][5] = {{1, 0, 1, 0, 1},

{0, 1, 0, 0, 0},

{0, 1, 1, 0, 0}};

Output: 11

Test Case 3:

Input: grid [2][2] = {{1, 1},

{1,1}};

Output: 4

Test Case 4:

Input: grid [2][2] = {{0, 0},

{0, 0}};

Output: 0

Test Case 5:

Input: grid [2][2] = {{1, 0},

{0, 0}};

Output: 0