**Problem 1**

**Background**: A group of friends embarks on a picnic and decides to engage in a friendly competition during their journey. The game they choose involves each participant stating a sentence about themselves. The objective is to have the sentence with the highest frequency of any single vowel. In the event of a tie, the winner is determined by the alphabetical precedence of the most frequent vowel.

**Objective**: Develop a program that assists in determining the winner of the game by analysing the sentences provided by each player. The program should accept a string input, which represents the sentence spoken by a player, and output the most frequent vowel(s) in that sentence. If multiple vowels tie for highest frequency, all such vowels should be printed in alphabetical order.

**Requirements**:

* The program must handle multiple sentences as input, one for each player.
* The output must clearly indicate the most frequent vowel(s) for each input sentence.
* In case of ties, vowels must be sorted alphabetically and displayed accordingly.

**Test cases:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| [  ["Alex", "I enjoy hiking in the mountains."],  ["Sam", "A lovely sunny day at the beach."],  ["Jamie", "Reading a book is my favorite pastime."],  ["Taylor", "I love playing video games on weekends."],  ["Chris", "Exploring new cities is exciting and fun."]  ] | {  'Alex': ['I'],  'Sam': ['A'],  'Jamie': ['A', 'I'],  'Taylor': ['E'],  'Chris': ['I']  } |

**Problem 2**

**Scenario**: Alex, an intrepid treasure hunter, is navigating through a labyrinth with seven checkpoints. At each checkpoint, there are multiple gates, each marked with a number representing the potential clues or challenges it holds. Only one gate at each checkpoint leads to the next, and the correct gate is determined by an equilibrium condition: the sum of all integers (clues/challenges) on its left must be equal to the sum of all integers on its right. If no such gate exists, the middle gate is deemed correct.

**Objective**: Develop a program that assists Alex in identifying the correct gates at each checkpoint to reach the final destination. The program should accept an array A of N integers, where each integer represents the number of clues or challenges associated with a gate, and output the index of the equilibrium gate.

**Note**: Equilibrium Value will be added to Right

**Requirements**:

* The program must process an array of integers as input for each checkpoint.
* The output must clearly indicate the index of the equilibrium gate.
* If no equilibrium gate exists, the program should output the index of the middle gate.

**Test cases:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| [  [2, 2, 1, 2, 1], # Checkpoint 1  [4, 2, 3, 1, 2, 1, 2, 3], # Checkpoint 2  [1, 1, 1, 1, 1], # Checkpoint 3  [3, 0, 3], # Checkpoint 4  [1, 2, 1, 1, 2, 1], # Checkpoint 5  [1, 1, 1, 2, 1], # Checkpoint 6  [5, 2, 1, 3, 1, 2, 5] # Checkpoint 7  ] | {  "Checkpoint 1": 2, # Equilibrium index  "Checkpoint 2": 3, # Equilibrium index  "Checkpoint 3": 2, # Middle index as no equilibrium  "Checkpoint 4": 1, # Equilibrium index  "Checkpoint 5": 3, # Equilibrium index  "Checkpoint 6": 3, # Equilibrium index  "Checkpoint 7": 3 # Middle index as no equilibrium  } |

Problem 3

You are given an integer array cards where cards[i] represents the value of the ith card. A pair of cards are matching if the cards have the same value.

Return the minimum number of consecutive cards you have to pick up to have a pair of matching cards among the picked cards. If it is impossible to have matching cards, return -1.

Input: cards = [3,4,2,3,4,7],[3,4,5,3,2,4,2,6,6]

Output: 4

Explanation: We can pick up the cards [3,4,2,3] which contain a matching pair of cards with value 3. Note that picking up the cards [4,2,3,4] is also optimal.