**Practice Questions:**

1. You are given an array of integers and a target integer. Write a function linearSearch that performs a linear search on the array and returns the index of the first occurrence of the target integer. If the target integer is not found in the array, the function should return -1.

2. You are given two arrays, arr1[] and arr2[]. Write a function arrIntersection that finds and prints the intersection of the two arrays. The intersection should include only the common elements (Considering there are no dupliactes) and print them in the order they appear in the first array.

3. You are given an array of integers. Write a function sumUniqueElements that calculates the sum of the elements in the array that appear only once.

4. You are given an array of integers. Write a function swapMinMax that swaps the minimum and maximum elements in the array. After the swap, print the modified array.

5. Given an array of integers, a lucky integer is an integer that has a frequency in the array equal to its value. Return the largest lucky integer in the array. If there is no lucky integer, return -1.

**Example 1:**

**Input: arr = { 2 , 2 , 3 , 4 }**

**Output: 2**

**Explanation: The only lucky number in the array is 2 because frequency[2] == 2.**

**Example 2:**

**Input: arr = { 1 , 2 , 2 , 3 , 3 , 3 }**

**Output: 3**

**Explanation: 1, 2 and 3 are all lucky numbers, return the largest of them.**

**Example 3:**

**Input: arr = { 2 , 2 , 2 , 3 , 3 }**

**Output: -1**

**Explanation: There are no lucky numbers in the array.**

6. Given an array of integers, return true if it is a valid mountain array. An array is a mountain array if and only if:

**arr.length >= 3**

**There exists some i with 0 < i < arr.length - 1 such that:**

**arr[0] < arr[1] < ... < arr[i - 1] < arr[i]**

**arr[i] > arr[i + 1] > ... > arr[arr.length - 1]**



Hope you all will get the idea after seeing this pic so make a function of bool type that tells whether the array passed to it is Mountain Array or not.

7.Given an integer array, move all 0's to the end of it while maintaining the relative order of the non-zero elements. You must do this in-place without making a copy of the array.

**Sample Execution 1:**

Input: [0, 1, 0, 3, 12]

Output: [1, 3, 12, 0, 0]

**Sample Execution 2:**

Input: [1, 2, 3]

Output: [1, 2, 3]  
  
  
**Char Arrays:**

8. Find and return the character that occurs the most in a string.

string -> momina

output : m

9. Write a function to shift each character of a string to the next one in the alphabet.

a → b, z → a, B → C, etc.