Q1: Given an integer array nums, return the third distinct maximum number in this array. If the third maximum does not exist, return the maximum number.

Example 1:

Input: nums = [3,2,1]

Output: 1

Explanation:

The first distinct maximum is 3.

The second distinct maximum is 2.

The third distinct maximum is 1.

Example 2:

Input: nums = [1,2]

Output: 2

Explanation:

The first distinct maximum is 2.

The second distinct maximum is 1.

The third distinct maximum does not exist, so the maximum (2) is returned instead.

Example 3:

Input: nums = [2,2,3,1]

Output: 1

Explanation:

The first distinct maximum is 3.

The second distinct maximum is 2 (both 2's are counted together since they have the same value).

The third distinct maximum is 1.

Constraints:

1 <= nums.length <= 104

-231 <= nums[i] <= 231 - 1

Q2: Write a function that takes in a string and returns the first non-repeating character in that string.

**Solution Q1**

// Q1: Given an integer array nums, return the third distinct maximum number in this array. If the third maximum does not exist, return the maximum number.

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

void dispArray(int arr[], int len) {

    for (int i = 0; i < len; i++) {

        cout << arr[i] << " ";

    }

}

int getLargest(int arr[], int n) {

    for (int i = 1; i < n; ++i) {

        if (arr[0] < arr[i])

            arr[0] = arr[i];

    }

    return arr[0];

}

int leastFrequent(int arr[], int n) {

    unordered\_map < int, int > hash;

    for (int i = 0; i < n; i++)

        hash[arr[i]]++;

    int min\_count = n + 1, res = -1;

    bool same\_count = true; // flag to check if all counts are the same

    for (auto i: hash) {

        if (min\_count > i.second) {

            res = i.first;

            min\_count = i.second;

            same\_count = false;

        } else if (min\_count == i.second) {

            same\_count = true;

        }

    }

    if (same\_count) {

        res = getLargest(arr, n);

    }

    return res;

}

int getThirdMax(int arr[], int n) {

    if (n < 3) {

        return getLargest(arr, n);

    } else {

        return leastFrequent(arr, n);

    }

}

bool elementInRange(int arr[], int n) {

    for (int i = 0; i < n; i++) {

        if ((-231 <= arr[i] && arr[i] <= 231) == false) {

            return false;

        }

    }

    return true;

}

int main() {

    int len;

    cout << "Enter Number of Elements of array : ";

    cin >> len;

    //1 <= nums.length <= 104

    //-231 <= nums[i] <= 231 - 1

    if (len >= 1 && len <= 104) {

        int arr[len];

        cout << "Enter Elememts of Array : " << endl;

        for (int i = 0; i < len; i++) {

            cin >> arr[i];

        }

        cout << "Array : ";

        dispArray(arr, len);

        if (elementInRange(arr, len)) {

            int result = getThirdMax(arr, len);

            cout << endl << "Distinct Maximum  : " << result;

        } else {

            cout << endl << "Invalid Inputs - 'Range Error in Elements of Array'";

        }

    } else {

        cout << endl << "Invalid Range - 'Length of array'";

    }

return 0;

}

**Solution 2:**

#include <iostream>

using namespace std;

void checkFirstNonRepeating(string str){

    int flag;

    for(int i = 0; i < str.length(); i++) {

     flag = 0;

     for(int j = 0; j < str.length(); j++) {

        if((str[i] == str[j]) && (i != j)) {

             flag = 1;

             break;

        }

     }

      if (flag == 0) {

          cout << "First non-repeating character is : "<<  str[i];

          break;

      }

 }

 if (flag == 1) {

     cout << "Didn't find any non-repeating character";

 }

}

int main() {

 string str;

 cout << "Enter String : ";

 cin >> str;

 checkFirstNonRepeating(str);

  return 0;

}