💡 **Q1.** Given an array of integer nums and an integer target, return indices of the two numbers such that they add up to the target.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

You can return the answer in any order.

Example: Input: nums = [2,7,11,15], target = 9 Output0 [0,1]

Explanation: Because nums[0] + nums[1] == 9, we return [0, 1]

ANS -

```
class Solution {

public:

   vector<int> twoSum(vector<int>& nums, int target) {



       for (int i = 0; i < nums.size(); i++) {

           for (int j = i + 1; j < nums.size(); j++) {

               if (nums[i] + nums[j] == target) {

                   return {i, j};

               }

           }

       }

       return {};

   }

};
```

💡 **Q2.** Given an integer array nums and an integer val, remove all occurrences of val in nums in-place. The order of the elements may be changed. Then return the number of elements in nums which are not equal to val.

Consider the number of elements in nums which are not equal to val be k, to get accepted, you need to do the following things:

- Change the array nums such that the first k elements of nums contain the elements which are not equal to val. The remaining elements of nums are not important as well as the size of nums.
- Return k.

Example : Input: nums = [3,2,2,3], val = 3 Output: 2, nums = [2,2,_,_]

Explanation: Your function should return k = 2, with the first two elements of nums being 2. It does not matter what you leave beyond the returned k (hence they are underscores)

ANS-

```
class Solution {

public:

   int removeElement(vector<int>& nums, int val) {



       int k=0;

       for(int i=0; i<nums.size(); i++){

           if(nums[i] != val){

               nums[k]=nums[i];

               k++;

           }

       }

       return k;

   }

};
```

💡 **Q3.** Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

Example 1: Input: nums = [1,3,5,6], target = 5

ANS -

```
class Solution {

public:

   int searchInsert(vector<int>& nums, int target) {

       int low=0;

       int high=nums.size();

       if(target>nums[high-1]){

           return high;

       }

       while(low<=high){

           int mid=(low+high)/2;

           if(target==nums[mid]){

               return mid;

           }

           if(target<nums[mid]){

               high=mid-1;

           }else

               low=mid+1;

       }

       return low;

   }

};
```

💡 **Q4.** You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's.

Increment the large integer by one and return the resulting array of digits.

Example 1: Input: digits = [1,2,3] Output: [1,2,4]

Explanation: The array represents the integer 123.

Incrementing by one gives 123 + 1 = 124. Thus, the result should be [1,2,4].

ANS -

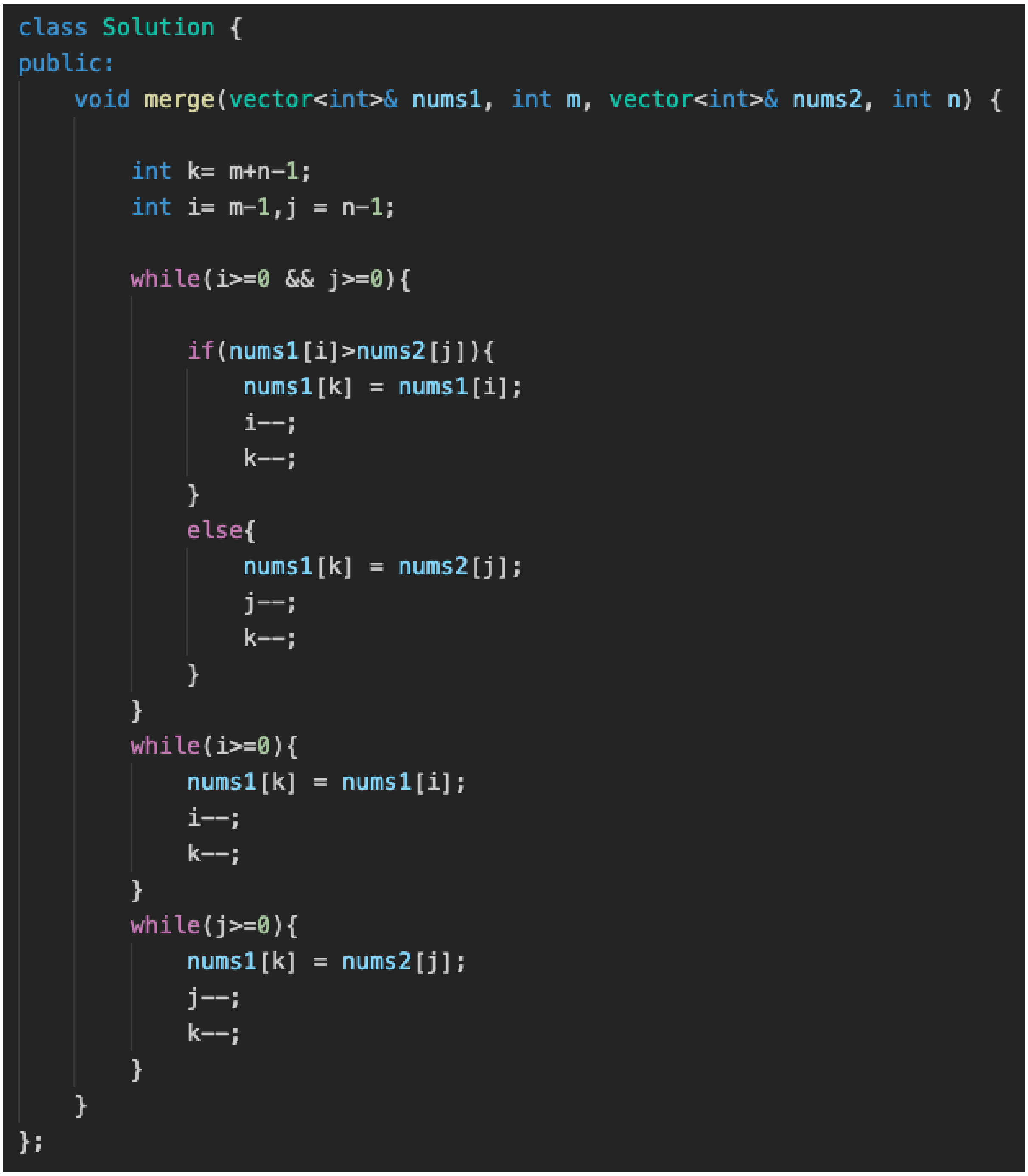
```
class Solution {

public:

   vector<int> plusOne(vector<int>& digits) {



       for(int i=digits.size()-1; i>=0; i--){

           if(digits[i]!=9){

               digits[i]++;

               return digits;               //just returning digits and
moving out of the loop means end

           }else

             digits[i]=0;

       }

       digits.insert(digits.begin(),1);     //adding 1 at the beginning

       return digits;

   }

};
```

💡 **Q5.** You are given two integer arrays nums1 and nums2, sorted in non-decreasing order, and two integers m and n, representing the number of elements in nums1 and nums2 respectively.

Example 1: Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3 Output: [1,2,2,3,5,6]

Explanation: The arrays we are merging are [1,2,3] and [2,5,6]. The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1.

ANS -

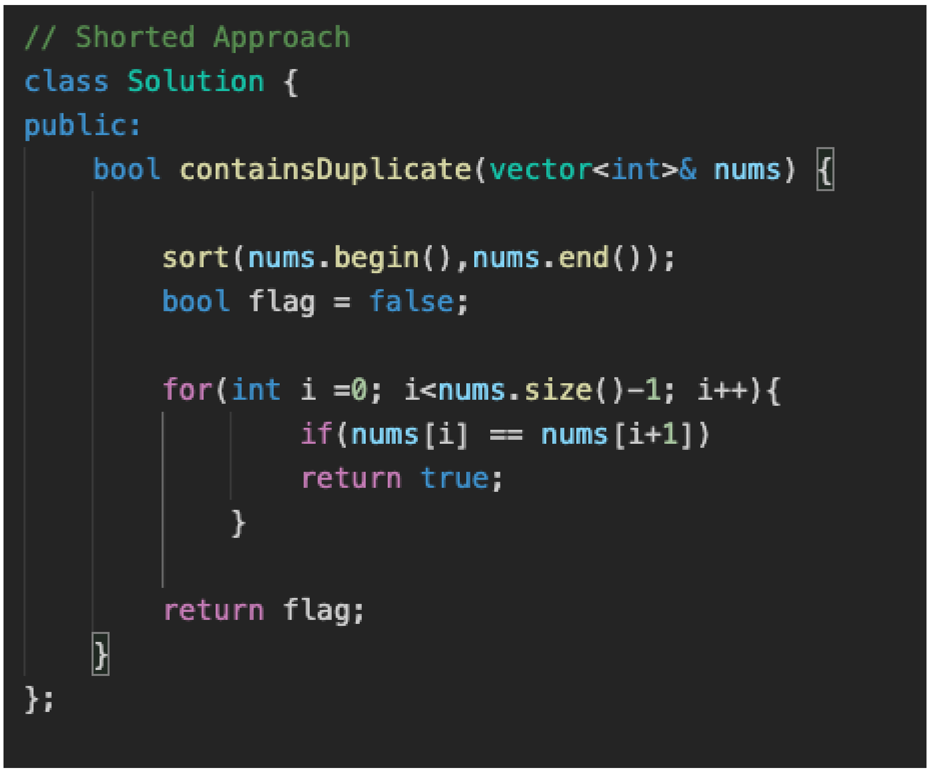


```cpp
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {

        int k= m+n-1;
        int i= m-1,j = n-1;

        while(i>=0 && j>=0){

            if(nums1[i]>nums2[j]){
                nums1[k] = nums1[i];
                i--;
                k--;
            }
            else{
                nums1[k] = nums2[j];
                j--;
                k--;
            }
        }
        while(i>=0){
            nums1[k] = nums1[i];
            i--;
            k--;
        }
        while(j>=0){
            nums1[k] = nums2[j];
            j--;
            k--;
        }
    }
};
```

💡 **Q6.** Given an integer array nums, return true if any value appears at least twice in the array, and return false if every element is distinct.

Example 1: Input: nums = [1,2,3,1] Output: true

ANS -



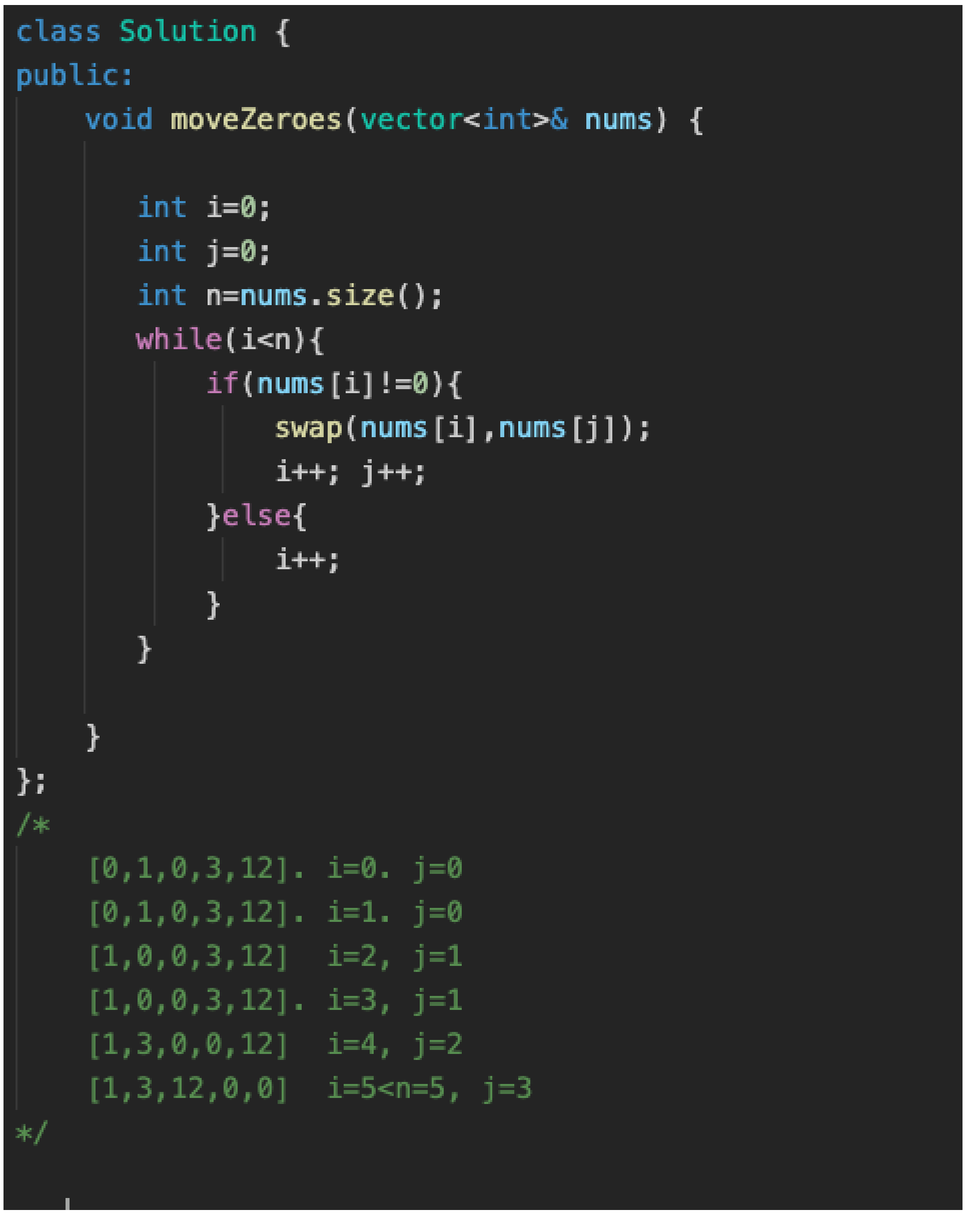
```cpp
// Shorted Approach
class Solution {
public:
    bool containsDuplicate(vector<int>& nums) {

        sort(nums.begin(),nums.end());
        bool flag = false;

        for(int i =0; i<nums.size()-1; i++){
            if(nums[i] == nums[i+1])
            return true;
        }

        return flag;
    }
};
```

💡 **Q7.** Given an integer array nums, move all 0's to the end of it while maintaining the relative order of the nonzero elements.

Note that you must do this in-place without making a copy of the array.

Example 1: Input: nums = [0,1,0,3,12] Output: [1,3,12,0,0]

ANS -



```cpp
class Solution {
public:
    void moveZeroes(vector<int>& nums) {

        int i=0;
        int j=0;
        int n=nums.size();
        while(i<n){
            if(nums[i]!=0){
                swap(nums[i],nums[j]);
                i++; j++;
            }else{
                i++;
            }
        }

    }
};
/*
    [0,1,0,3,12]. i=0. j=0
    [0,1,0,3,12]. i=1. j=0
    [1,0,0,3,12]  i=2, j=1
    [1,0,0,3,12]. i=3, j=1
    [1,3,0,0,12]  i=4, j=2
    [1,3,12,0,0]  i=5<n=5, j=3
*/
```

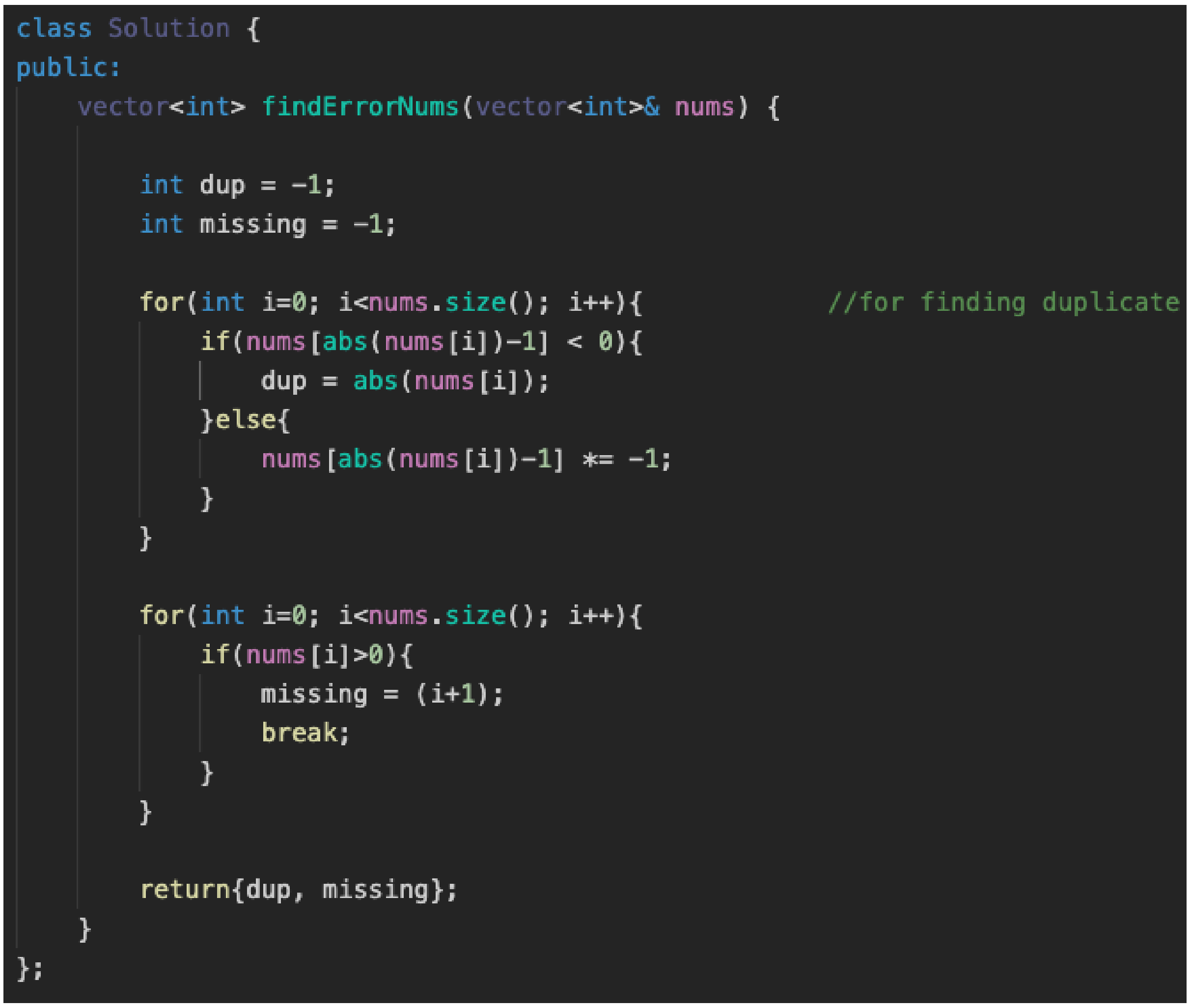
💡 **Q8.** You have a set of integers s, which originally contains all the numbers from 1 to n. Unfortunately, due to some error, one of the numbers in s got duplicated to another number in the set, which results in repetition of one number and loss of another number.

You are given an integer array nums representing the data status of this set after the error.

Find the number that occurs twice and the number that is missing and return them in the form of an array.

Example 1: Input: nums = [1,2,2,4] Output: [2,3]

ANS -



```
class Solution {
public:
    vector<int> findErrorNums(vector<int>& nums) {

        int dup = -1;
        int missing = -1;

        for(int i=0; i<nums.size(); i++){            //for finding duplicate
            if(nums[abs(nums[i])-1] < 0){
                dup = abs(nums[i]);
            }else{
                nums[abs(nums[i])-1] *= -1;
            }
        }

        for(int i=0; i<nums.size(); i++){
            if(nums[i]>0){
                missing = (i+1);
                break;
            }
        }

        return{dup, missing};
    }
};
```