Assignment 3

Question 1

Given an integer array nums of length n and an integer target, find three integers in nums such that the sum is closest to the target. Return the sum of the three integers. You may assume that each input would have exactly one solution. Example 1: Input: nums = [-1,2,1,-4], target = 1 Output: 2 Explanation: The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).

Ans =

```
class Solution {
public:
    int lengthOfLongestSubstring(string s) {
        int n = s.length();
        int maxLength = 0;
        unordered_set<char> charSet;
        int left = 0;

        for (int right = 0; right < n; right++) {
            if (charSet.count(s[right]) == 0) {
                charSet.insert(s[right]);
                maxLength = max(maxLength, right - left + 1);
            } else {
                while (charSet.count(s[right])) {
                    charSet.erase(s[left]);
                    left++;
                }
                charSet.insert(s[right]);
            }
        }

        return maxLength;
    }
}
```

Question 2

Given an array nums of n integers, return an array of all the unique quadruplets [nums[a], nums[b], nums[c], nums[d]] such that: ● 0 <= a, b, c, d < n ● a, b, c, and d are distinct. ● nums[a] + nums[b] + nums[c] + nums[d] == target You may return the answer in any order. Example 1: Input: nums = [1,0,-1,0,-2,2], target = 0 Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

Ans =

```
class Solution {

public:

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

        int n = nums.size();
```

```
        sort(nums.begin(), nums.end());
        set<vector<int>> set;
        vector<vector<int>> output;
        for(int i=0; i<n-3; i++){
            for(int j=i+1; j<n-2; j++){
                long long newTarget = (long long)target - (long long)nums[i] -
(long long)nums[j];
                int low = j+1, high = n-1;
                while(low < high){
                    if(nums[low] + nums[high] < newTarget){
                        low++;
                    }
```

Question 3

A permutation of an array of integers is an arrangement of its members into a sequence or linear order.

For example, for arr = [1,2,3], the following are all the permutations of arr: [1,2,3], [1,3,2], [2, 1, 3], [2, 3, 1], [3,1,2], [3,2,1].

The next permutation of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the next permutation of that array is the permutation that follows it in the sorted container.

If such an arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

● For example, the next permutation of arr = [1,2,3] is [1,3,2]. ● Similarly, the next permutation of arr = [2,3,1] is [3,1,2]. ● While the next permutation of arr = [3,2,1] is [1,2,3] because [3,2,1] does not have a lexicographical larger rearrangement.

Given an array of integers nums, find the next permutation of nums. The replacement must be in place and use only constant extra memory.

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

Ans = class Solution {

```
public:
    void nextPermutation(vector<int>& nums) {
        int n = nums.size(), index = -1;
        for(int i=n-2; i>=0; i--){
            if(nums[i] < nums[i+1]){
                index = i;
                break;
            }
        }
        for(int i=n-1; i>=index && index != -1; i--){
            if(nums[i] > nums[index]){
                swap(nums[i], nums[index]);
                break;
            }
        }
        reverse(nums.begin() + index + 1, nums.end());
    }
};
```

Question4

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. You must write an algorithm with O(log n) runtime complexity. Example 1: Input: nums = [1,3,5,6], target = 5 Output: 2.

Ans =

```
class Solution {
```

```
public:
    bool containsDuplicate(vector<int>& nums) {
        int n = nums.size();
        for (int i = 0; i < n - 1; i++) {
            for (int j = i + 1; j < n; j++) {
                if (nums[i] == nums[j])
                    return true;
            }
        }
        return false;
    }
};
```

Question 6

Given a non-empty array of integers nums, every element appears twice except for one. Find that single one. You must implement a solution with a linear runtime complexity and use only constant extra space. Example 1: Input: nums = [2,2,1] Output: 1

Ans =

```
class Solution {
public:
    int singleNumber(vector<int>& nums) {
        unordered_map<int,int> a;
      for(auto x: nums)
        a[x]++;
      for(auto z:a)
        if(z.second==1)
```

```
            return z.first;
        return -1;
    }
};
```