***Week – 8 (******06.06.2021 – 12.06.2021)***

***RANDOM CODES***

1. ***Distant Barcodes:***

class Solution {

public:

vector<int> rearrangeBarcodes(vector<int>& barcodes) {

sort(barcodes.begin(), barcodes.end());

int i=1, j=2;

while(i<barcodes.size() && j<barcodes.size())

{

if(barcodes[i-1] == barcodes[i])

{

if(barcodes[i] != barcodes[j])

{

swap(barcodes[i],barcodes[j]);

i++;

}

j++;

}

else

{

i++;

j=i+1;

}

}

i = barcodes.size()-2;

j = barcodes.size()-3;

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

{

if(barcodes[i+1] == barcodes[i])

{

if(barcodes[i] != barcodes[j])

{

swap(barcodes[i],barcodes[j]);

i--;

}

j--;

}

else

{

i--;

j=i-1;

}

}

return barcodes;

}

};

1. ***Increasing Decreasing String:***

class Solution {

public:

string sortString(string s) {

string res="";

int i;

vector<int> freq(26, 0);

for(i=0; i<s.size(); i++)

freq[s[i]-'a']++;

while(res.size()<s.size())

{

for(i=0; i<26; i++)

{

if(freq[i] > 0)

{

res.push\_back((char)'a'+i);

freq[i]--;

}

}

for(i=25; i>=0; i--)

{

if(freq[i] > 0)

{

res.push\_back((char)'a'+i);

freq[i]--;

}

}

}

return res;

}

};

1. ***Long Pressed Name:***

class Solution {

public:

bool isLongPressedName(string name, string typed) {

int i=0, j=0;

while(i<name.size() || j<typed.size())

{

if (i < name.size() && name[i] == typed[j]) ++i, ++j;

else if (i > 0 && name[i - 1] == typed[j]) ++j;

else return false;

}

return i == name.size();

}

};

1. ***Mean of Array After Removing Some Elements:***

class Solution {

public:

double trimMean(vector<int>& arr) {

if(arr.size()%20 != 0 || arr.size() < 20) return 0.000;

sort(arr.begin(), arr.end());

int remove\_element = arr.size()\*0.05, i;

double sum = 0;

for(i=remove\_element; i<arr.size()-remove\_element; i++)

sum += arr[i];

return (double)sum/(arr.size()-remove\_element-remove\_element);

}

};

1. ***Find Words That Can Be Formed by Characters:***

class Solution {

public:

int countCharacters(vector<string>& words, string chars) {

int i, j, count = 0, sum;

map<char, int> m;

for(i=0; i<words.size(); i++)

{

sum = 0;

for(auto p : chars)

m[p]++;

for(j=0; j<words[i].size(); j++)

{

auto pos = m.find(words[i][j]);

if(pos->second != 0)

{

m[words[i][j]]--;

sum++;

}

else break;

}

if(sum == words[i].size()) count = count + words[i].size();

m.clear();

}

return count;

}

};

1. ***Element Appearing More Than 25% In Sorted Array:***

class Solution {

public:

int findSpecialInteger(vector<int>& arr) {

int per = arr.size()\*0.25, i;

map<int,int> m;

for(i=0; i<arr.size(); i++)

m[arr[i]]++;

for(auto e : m)

if(e.second > per) return e.first;

return -1;

}

};

1. ***Number of Equivalent Domino Pairs:***

class Solution {

public:

int numEquivDominoPairs(vector<vector<int>>& dominoes) {

if(dominoes.size() < 2) return 0;

int i, j, count = 0;

map<pair<int,int>,int> m;

for(i=0; i<dominoes.size(); i++)

m[make\_pair(min(dominoes[i][0],dominoes[i][1]),max(dominoes[i][0],dominoes[i][1]))]++;

for(pair<pair<int,int>,int> p : m) count+=(p.second\*(p.second-1))/2;

return count;

}

};

1. ***Kth Missing Positive Number:***

class Solution {

public:

int findKthPositive(vector<int>& arr, int k) {

for(auto num : arr)

if(num<=k) k++;

return k;

}

};

1. ***Count Items Matching a Rule:***

class Solution {

public:

int countMatches(vector<vector<string>>& items, string ruleKey, string ruleValue) {

int i, j, count=0;

j = (ruleKey == "type") ? 0 : (ruleKey == "color") ? 1 : 2;

for(i=0; i<items.size(); i++)

if(items[i][j] == ruleValue) count++;

return count;

}

};

1. ***Check If All 1's Are at Least Length K Places Away:***

class Solution {

public:

bool kLengthApart(vector<int>& nums, int k) {

int i, pos=-1;

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

{

if(nums[i] == 1)

{

if(pos!=-1 and i-pos-1<k)return false;

pos = i;

}

}

return true;

}

};

1. ***Create Target Array in the Given Order:***

class Solution {

public:

vector<int> createTargetArray(vector<int>& nums, vector<int>& index) {

vector<int> res;

int i;

for(i=0; i<index.size(); i++)

res.insert(res.begin()+index[i],nums[i]);

return res;

}

};

1. ***How Many Numbers Are Smaller Than the Current Number:***

class Solution {

public:

vector<int> smallerNumbersThanCurrent(vector<int>& nums) {

vector<int> n = nums, res;

int i, temp;

sort(n.begin(), n.end());

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

{

temp = find(n.begin(), n.end(), nums[i]) - n.begin();

res.push\_back(temp);

}

return res;

}

};