***Week – 1 (28.03.2021 – 04.04.2021)***

***RANDOM CODES:***

1. ***Find Smallest Letter Greater Than Target:***

class Solution {

public:

char nextGreatestLetter(vector<char>& letters, char target) {

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

int i=0;

while(i<letters.size())

{

if(letters[i] > target) return letters[i];

i++;

}

return letters[0];

}

};

1. ***N-Repeated Element in Size 2N Array:***

class Solution {

public:

int repeatedNTimes(vector<int>& A) {

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

vector<int> sum;

int i, start, count = 0;

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

{

if(A[i] == A[i+(A.size()/2)-1]) return A[i];

}

return -1;

}

};

1. ***Majority Element:***

class Solution {

public:

int majorityElement(vector<int>& nums) {

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

int i, start, count;;

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

{

if(nums[i] == nums[i+(nums.size()/2)]) return nums[i];

}

return -1;

}

};

1. ***Remove Duplicates from Sorted Array II:***

class Solution {

public:

int removeDuplicates(vector<int>& nums) {

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

if(nums[i-1] == nums[i] && nums[i] == nums[i+1])

nums.erase(remove(nums.begin()+i+1,nums.end(),nums[i]),nums.end());

return nums.size();

}

};

1. ***Factorial Trailing Zeroes:***

class Solution {

public:

int trailingZeroes(int n) {

if(n == 0) return 0;

int i, count = 0;

while(n>0)

{

n = n/5;

count = count + n;

}

return count;

}

};

1. ***Add Digits:***

class Solution {

public:

int addDigits(int num) {

int i=0, r=0, n;

while(num>9)

{

r = 0;

while(num > 0)

{

n = num%10;

r = r + n;

num = num/10;

}

num = r;

}

return num;

}

};

1. ***Check If Two String Arrays are Equivalent:***

class Solution {

public:

bool arrayStringsAreEqual(vector<string>& word1, vector<string>& word2) {

int i;

string s1="", s2="";

for(i=0; i<word1.size(); i++)

s1 = s1 + word1[i];

for(i=0; i<word2.size(); i++)

s2 = s2 + word2[i];

if(s1 == s2) return true;

else return false;

}

};

1. ***Find Peak Element:***

class Solution {

public:

int findPeakElement(vector<int>& nums) {

int max;

max = \*max\_element(nums.begin(), nums.end());

auto pos = find(nums.begin(), nums.end(), max);

if(pos != nums.end()) return pos-nums.begin();

else return -1;

}

};

1. ***Third Maximum Number:***

class Solution {

public:

int thirdMax(vector<int>& nums) {

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

nums.erase(unique(nums.begin(), nums.end()), nums.end());

if(nums.size() < 3) return \*max\_element(nums.begin(), nums.end());

else return nums[nums.size()-3];

}

};

1. ***Valid Palindrome:***

#include <string.h>

class Solution {

public:

bool isPalindrome(string s) {

int i=0,j=s.size()-1;

while(i<=j)

{

if(!isalnum(s[i]))

{

i++;

continue;

}

if(!isalnum(s[j]))

{

j--;

continue;

}

if(tolower(s[i])!=tolower(s[j])) return false;

i++;

j--;

}

return true;

}

};

1. ***Maximum Product Subarray:***

class Solution {

public:

int maxProduct(vector<int>& nums) {

int i,j,m,p;

m = \*max\_element(nums.begin(), nums.end());

for(i=0; i<nums.size()-1; i++)

{

p = nums[i];

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

{

p = p \* nums[j];

m = max(m,p);

}

}

return m;

}

};

1. ***Ugly Number:***

class Solution {

public:

bool isUgly(int n) {

if(n<0) return false;

if(n==0) return false;

while(n>1)

{

cout<<n<<" ";

if(n%2 == 0) n = n/2;

else if(n%3 == 0) n = n/3;

else if(n%5 == 0) n = n/5;

else return false;

}

return true;

}

};