***Week – 4 (19.04.2021 – 25.04.2021)***

***RANDOM CODES***

1. ***Student Attendance Record I:***

class Solution {

public:

bool checkRecord(string s) {

return ((s.find("LLL") == string::npos) && (count(s.begin(), s.end(), 'A') < 2));

}

};

1. ZigZag Conversion:

class Solution {

public:

string convert(string s, int numRows) {

if(numRows == 1 || s.size() == 1) return s;

string res="";

int cycle = 2 \* numRows - 2, i, j;

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

{

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

{

res += s[j+i];

if(i!=0 && i!=numRows-1 && j+cycle-i<s.size())

res += s[j+cycle-i];

}

}

return res;

}

};

1. ***Valid Parentheses:***

class Solution {

public:

bool isValid(string s) {

stack <char> st;

int i;

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

{

if(s[i] == '{' || s[i] == '[' || s[i] == '(') st.push(s[i]);

if(s[i] == '}' || s[i] == ']' || s[i] == ')')

{

if(st.size() == 0) return false;

else if(st.top() == '{' && s[i] == '}') st.pop();

else if(st.top() == '(' && s[i] == ')') st.pop();

else if(st.top() == '[' && s[i] == ']') st.pop();

else return false;

}

}

if(st.size() == 0)return true;

else return false;

}

};

1. ***Longest Common Prefix:***

class Solution {

public:

string longestCommonPrefix(vector<string>& strs) {

if(strs.size() == 0) return "";

if(strs.size() == 1) return strs[0];

string res="";

int i,j;

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

{

char s = strs[0][i];

for(j=1; j<strs.size(); j++)

if(i == strs[j].length() || strs[j][i] != s) return res;

res = res + s;

}

return res;

}

};

1. ***Daily Temperatures:***

class Solution {

public:

vector<int> dailyTemperatures(vector<int>& T) {

vector<int> res;

stack<int> s;

int i;

for(i=T.size()-1; i>=0; i--)

{

while(!s.empty() && T[i] >= T[s.top()]) s.pop();

if(s.size() == 0) res.push\_back(0);

else if(T[s.top()]>T[i]) res.push\_back(s.top()-i);

s.push(i);

}

reverse(res.begin(), res.end());

return res;

}

};

1. ***Asteroid Collision:***

class Solution {

public:

vector<int> asteroidCollision(vector<int>& asteroids) {

int i;

vector<int> s;

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

{

if(asteroids[i]>0) s.push\_back(asteroids[i]);

else

{

while(!s.empty() && s.back()>=0 && s.back()<abs(asteroids[i])) s.pop\_back();

if(s.empty() || s.back()<0) s.push\_back(asteroids[i]);

else if(s.back()==abs(asteroids[i])) s.pop\_back();

}

}

return s;

}

};

1. ***K-diff Pairs in an Array:***

class Solution {

public:

int findPairs(vector<int>& nums, int k) {

vector<vector <int>> res;

int i, j;

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

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

if(abs(nums[i] - nums[j]) == k)

{

if(nums[i] <= nums[j]) res.push\_back({nums[i], nums[j]});

else res.push\_back({nums[j], nums[i]});

}

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

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

return res.size();

}

};

1. ***Reverse Vowels of a String:***

class Solution {

public:

bool isvowel(char c)

{

if(c=='a' || c=='A' || c=='e' || c=='E' || c=='i' || c=='I' || c=='o' || c=='O' || c=='u' || c=='U') return true;

else return false;

}

string reverseVowels(string s) {

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

char temp;

while(i<j)

{

if(isalpha(s[i]) && isvowel(s[i]))

for(k=j; k>=0 && k>i; k--)

{

if(isalpha(s[k]) && isvowel(s[k]))

{

temp = s[i];

s[i] = s[k];

s[k] = temp;

j = k;

break;

}

}

else if(isalpha(s[j]) && isvowel(s[j]))

for(k=i; k<s.size() && k<j; k++)

{

if(isalpha(s[k]) && isvowel(s[k]))

{

temp = s[k];

s[k] = s[j];

s[j] = temp;

i = k;

break;

}

}

i++;

j--;

}

return s;

}

};

1. ***Maximum Ascending Subarray Sum:***

class Solution {

public:

int maxAscendingSum(vector<int>& nums) {

int i, maxsum = INT\_MIN, sum = nums[0];

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

{

if(nums[i-1] < nums[i])

sum = sum + nums[i];

else

{

maxsum = max(maxsum, sum);

sum = nums[i];

}

}

maxsum = max(maxsum, sum);

return maxsum;

}

};

1. ***N-th Tribonacci Number:***

class Solution {

public:

int tribonacci(int n) {

if(n == 0) return 0;

if(n == 1 || n == 2) return 1;

int t0=0, t1=1, t2=1, nxt, i;

for(i=3; i<=n; i++)

{

nxt = t0+t1+t2;

t0 = t1;

t1 = t2;

t2 = nxt;

}

return nxt;

}

};

1. ***Count Primes:***

class Solution {

public:

int countPrimes(int n) {

int i, j, count=0;

vector<bool> v;

for(int i=0; i<n; i++)

v.push\_back(false);

for(i=2; i\*i<=n; i++)

if(v[i] == false)

for(int j = i\*i; j<=n; j=j+i)

v[j]=true;

for(i=2; i<n; i++)

if(v[i] == false) count++;

return count;

}

};

1. ***Number of Digit One:***

class Solution {

public:

int countDigitOne(int n) {

int countr = 0;

for (long long i = 1; i <= n; i \*= 10) {

long long divider = i \* 10;

countr += (n / divider) \* i + min(max(n % divider - i + 1, 0LL), i);

}

return countr;

}

};

1. ***Minimum Size Subarray Sum:***

class Solution {

public:

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

if (nums.size() == 0) return 0;

int ans = INT\_MAX;

vector<int> sums(nums.size());

sums[0] = nums[0];

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

sums[i] = sums[i - 1] + nums[i];

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

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

{

int sum = sums[j] - sums[i] + nums[i];

if (sum >= target)

{

ans = min(ans, (j - i + 1));

break;

}

}

return (ans != INT\_MAX) ? ans : 0;

}

};

1. ***Largest Number:***

class Solution {

public:

static bool bysum(const string& a, const string& b)

{

return (a + b) > (b + a);

}

string largestNumber(vector<int>& nums) {

if(nums.size() == 0) return "";

int i;

vector<string> vec;

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

vec.push\_back(to\_string(nums[i]));

sort(vec.begin(), vec.end(), bysum);

string res="";

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

res += vec[i];

if(res[0] == '0') return "0";

return res;

}

};

1. ***Minimum Subsequence in Non-Increasing Order:***

class Solution {

public:

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

if(nums.size() == 1) return nums;

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

vector<int> res;

int i=0, j=nums.size()-1, sum1 = nums[0], sum2 = nums[nums.size()-1];

res.push\_back(nums[nums.size()-1]);

while(i<j)

{

if(sum1 >= sum2)

{

j--;

sum2 = sum2 + nums[j];

res. push\_back(nums[j]);

}

if(sum1 < sum2)

{

i++;

sum1 = sum1 + nums[i];

}

}

return res;

}

};