***Week – 9 (******13.06.2021 – 19.06.2021)***

***RANDOM CODES***

1. ***Decrypt String from Alphabet to Integer Mapping:***

class Solution {

public:

vector<char> alpha{'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};

string freqAlphabets(string s) {

stack<int> st;

string res = "";

int i, val;

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

{

if(s[i] == '#')

{

val = st.top();

st.pop();

val = (st.top())\*10 + val;

st.pop();

st.push(val);

}

else st.push(s[i]-'0');

}

while(!st.empty())

{

res = alpha[st.top()-1] + res;

st.pop();

}

return res;

}

};

1. ***Defanging an IP Address:***

class Solution {

public:

string defangIPaddr(string address) {

string res = "";

int i;

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

{

if(address[i] == '.') res += "[.]";

else res += address[i];

}

return res;

}

};

1. ***Minimum Number of Steps to Make Two Strings Anagram:***

class Solution {

public:

int minSteps(string s, string t) {

map<char,int> m;

int i, count=0;

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

{

m[s[i]]++;

m[t[i]]--;

}

for(auto e : m)

if(e.second > 0) count += e.second;

return count;

}

};

1. ***Group Anagrams:***

class Solution {

public:

vector<vector<string>> groupAnagrams(vector<string>& strs) {

map<string,vector<string>> m;

vector<vector<string>> ans;

int i;

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

{

string temp = strs[i];

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

m[temp].push\_back(strs[i]);

}

for(auto e : m)

ans.push\_back(e.second);

return ans;

}

};

1. ***Palindrome Pairs:***

class Solution {

public:

bool check\_palindrome(string s)

{

int i;

for(i=0; i<s.length()/2; i++)

if(s[i]!=s[s.length()-i-1])

return false;

return true;

}

vector<vector<int>> palindromePairs(vector<string>& words) {

vector<vector<int>> ans;

unordered\_map<string,int> mp;

int i, j;

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

{

string temp=words[i];

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

mp[temp]=i;

}

if(mp.count(""))

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

if(!words[i].empty() && check\_palindrome(words[i]))

ans.push\_back({i,mp[""]});

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

{

string left,right;

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

{

left.push\_back(words[i][j]);

right=words[i].substr(j+1);

if(mp.count(left) && i!=mp[left] && check\_palindrome(right))

ans.push\_back({i,mp[left]});

if(mp.count(right) && i!=mp[right] && check\_palindrome(left))

ans.push\_back({mp[right],i});

}

}

return ans;

}

};

1. ***Reverse Words in a String III:***

class Solution {

public:

string reverseWords(string s) {

stack<string> st;

int i=0;

string str="", res="";

while(i<s.size())

{

if(s[i] == ' ')

{

st.push(str);

str = "";

}

else str += s[i];

i++;

}

st.push(str);

while(!st.empty())

{

string temp = st.top();

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

if(res.size()!=0) res = temp + ' ' + res;

else res = temp;

st.pop();

}

return res;

}

};

1. ***Second Largest Digit in a String:***

class Solution {

public:

int secondHighest(string s) {

int i;

vector<int> num;

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

if(isdigit(s[i])) num.push\_back(s[i]-'0');

sort(num.begin(), num.end(), greater<int>());

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

if(num.size() < 2) return -1;

return num[1];

}

};

1. ***Truncate Sentence:***

class Solution {

public:

string truncateSentence(string s, int k) {

int i=0;

string res="";

while(i<s.size() && k>0)

{

if(s[i] == ' ') k--;

if(k == 0) break;

res = res + s[i];

i++;

}

return res;

}

};

1. ***Check if the Sentence Is Pangram:***

class Solution {

public:

bool checkIfPangram(string sentence) {

if(sentence.size() < 26) return false;

vector<bool> chars(26,false);

int i;

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

chars[sentence[i]-'a'] = true;

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

if(chars[i] == false) return false;

return true;

}

};

1. ***Generate Parentheses:***

class Solution {

public:

void create(vector<string>& v, const string& curr, int open, int n)

{

if(curr.size() == n)

{

v.push\_back(curr);

return;

}

if(open < n-curr.size()) create(v, curr+'(', open+1, n);

if(open > 0) create(v, curr+')', open-1, n);

}

vector<string> generateParenthesis(int n) {

vector<string> v;

create(v, "", 0, n\*2);

return v;

}

};

1. ***Complement of Base 10 Integer:***

class Solution {

public:

int bitwiseComplement(int n) {

if(n == 0) return 1;

int bits = floor(log2(n))+1;

return ((1<<bits)-1)^n;

}

};

1. ***Closest Divisors:***

class Solution {

public:

vector<int> closestDivisors(int num) {

int i, num1=num+1, num2=num+2;

for(i=sqrt(num2); i>=1; i--)

{

if(num1%i==0)return {num1/i,i};

if(num2%i==0)return {num2/i,i};

}

return {-1,-1};

}

};