***Week – 5 (26.04.2021 – 02.05.2021)***

***RANDOM CODES***

1. ***Add Strings:***

class Solution {

public:

string addStrings(string num1, string num2) {

int sum = 0, carry = 0, i = num1.size()-1, j = num2.size()-1;

string res = "";

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

{

sum = carry;

carry = 0;

if(i>=0)

{

sum = sum + num1[i] - '0';

i--;

}

if(j>=0)

{

sum = sum + num2[j] - '0';

j--;

}

carry = sum / 10;

sum = sum % 10;

res = to\_string(sum) + res;

}

if(carry!=0) res = to\_string(carry) + res;

return res;

}

};

1. ***Add to Array-Form of Integer:***

class Solution {

public:

vector<int> addToArrayForm(vector<int>& num, int k) {

int i = num.size()-1, sum = 0, carry = 0;

vector<int> result;

while(i>=0 || k>0)

{

sum = carry;

carry = 0;

if(k>0)

{

sum = sum + (k%10);

k = k/10;

}

if(i>=0)

{

sum = sum + num[i];

i--;

}

carry = sum/10;

sum = sum%10;

result.push\_back(sum);

}

if(carry!=0)

{

while(carry>0)

{

sum = carry%10;

result.push\_back(sum);

carry = carry/10;

}

}

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

return result;

}

};

1. ***First Unique Character in a String:***

class Solution {

public:

int firstUniqChar(string s) {

int fre[26] = {0}, i;

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

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

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

if(fre[s[i] - 'a'] == 1) return i;

return -1;

}

};

1. ***Sum of Square Numbers:***

class Solution {

public:

bool judgeSquareSum(int c) {

int top = sqrt(c), i = 0, x;

while(i<=top)

{

x = i\*i;

if(x == c - top\*top) return true;

if(x > c - top\*top) top--;

if(x < c - top\*top) i++;

}

return false;

}

};

1. ***Valid Square:***

class Solution {

public:

bool validSquare(vector<int>& p1, vector<int>& p2, vector<int>& p3, vector<int>& p4) {

vector<int> res(6);

int x, i;

res[0] = (pow((p1[0]-p2[0]),2) + pow((p1[1]-p2[1]),2));

res[1] = (pow((p1[0]-p3[0]),2) + pow((p1[1]-p3[1]),2));

res[2] = (pow((p1[0]-p4[0]),2) + pow((p1[1]-p4[1]),2));

res[3] = (pow((p2[0]-p3[0]),2) + pow((p2[1]-p3[1]),2));

res[4] = (pow((p2[0]-p4[0]),2) + pow((p2[1]-p4[1]),2));

res[5] = (pow((p3[0]-p4[0]),2) + pow((p3[1]-p4[1]),2));

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

x = res[0];

if(x == 0 || res[1]!=x || res[2]!=x || res[3]!=x || res[4]!=2\*x || res[5]!=2\*x)

return false;

return true;

}

};

1. ***Number of Segments in a String:***

class Solution {

public:

int countSegments(string s) {

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

int i, count=0;

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

if((i == 0 || s[i-1] == ' ') && s[i] != ' ')

count++;

return count;

}

};