**Assignment Questions 7**

**Question 1**

Given two strings s and t, determine if they are isomorphic.

Two strings s and t are isomorphic if the characters in s can be replaced to get t.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.

**Example 1:**

**Input:** s = "egg", t = "add"

**Output:** true

Solve:-

class Solution {

public boolean isIsomorphic(String s, String t) {

int map1[]=new int[200];

int map2[]=new int[200];

if(s.length()!=t.length())

return false;

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

{

if(map1[s.charAt(i)]!=map2[t.charAt(i)])

return false;

map1[s.charAt(i)]=i+1;

map2[t.charAt(i)]=i+1;

}

return true;

}

}

**Question 2**

Given a string num which represents an integer, return true if num is a ***strobogrammatic number***.

A **strobogrammatic number** is a number that looks the same when rotated 180 degrees (looked at upside down).

**Example 1:**

**Input:** num = "69"

**Output:**

true

Solve:-

public boolean isStrobogrammatic(String num) {

if(num == null || num.length() == 0) return false;

int left = 0, right = num.length() - 1;

while(left <= right){

char c = num.charAt(left);

if(!map.containsKey(c) || map.get(c) != num.charAt(right)) return false;

left++;

right--;

}

return true;

}

private Map<Character, Character> map = new HashMap<Character, Character>(){

{

put('0', '0');

put('1', '1');

put('6', '9');

put('8', '8');

put('9', '6');

}

};

// better solution

public boolean isStrobogrammatic(String num) {

for (int i=0, j=num.length()-1; i <= j; i++, j--)

if (!"00 11 88 696".contains(num.charAt(i) + "" + num.charAt(j)))

return false;

return true;

}

**Question 3**

Given two non-negative integers, num1 and num2 represented as string, return the sum of num1 and num2 as a string.

You must solve the problem without using any built-in library for handling large integers (such as BigInteger). You must also not convert the inputs to integers directly.

**Example 1:**

**Input:** num1 = "11", num2 = "123"

**Output:**

"134"

Solve:-

class Solution {

public String addStrings(String num1, String num2) {

StringBuilder sb = new StringBuilder();

int i = num1.length() - 1, j = num2.length() - 1;

int carry = 0;

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

int sum = carry;

if (i >= 0) sum += (num1.charAt(i--) - '0');

if (j >= 0) sum += (num2.charAt(j--) - '0');

sb.append(sum % 10);

carry = sum / 10;

}

if (carry != 0) sb.append(carry);

return sb.reverse().toString();

}

}

Q**uestion 4**

Given a string s, reverse the order of characters in each word within a sentence while still preserving whitespace and initial word order.

**Example 1:**

**Input:** s = "Let's take LeetCode contest"

**Output:** "s'teL ekat edoCteeL tsetnoc"

Solve:-

class Solution {

public String reverseWords(String s) {

int l=0;

int r=0;

char a[]=s.toCharArray();

while(l<s.length()){

while(r<s.length() && a[r]!=' '){

r++;

}

//reverse part

//r-1 because r is pointing current at blank space

//Let's

// |

// r

reverse(a,l,r-1);

l=r+1;

r=l;

}

return String.valueOf(a);

}

public String reverse(char s[],int l,int r){

while(l<r){

char temp=s[l];

s[l]=s[r];

s[r]=temp;

l++;

r--;

}

return String.valueOf(s);

}

}

**Question 5**

Given a string s and an integer k, reverse the first k characters for every 2k characters counting from the start of the string.

If there are fewer than k characters left, reverse all of them. If there are less than 2k but greater than or equal to k characters, then reverse the first k characters and leave the other as original.

**Example 1:**

**Input:** s = "abcdefg", k = 2

**Output:**

"bacdfeg"

Solve:-

class Solution {

public String reverseStr(String s, int k) {

int i=0;

String ans="";

while(i<s.length()){

if(i+(2\*k)-1<s.length()||i+(2\*k)-1>=s.length()&&s.length()-i>=k){

int index=i+(2\*k)-1;

StringBuilder str=new StringBuilder();

String app=s.substring(i,i+k);

str.append(app);

str.reverse();

ans+=str.toString();

i=i+k;

while(i<s.length()&&i<=index){

ans+=s.charAt(i);

i++;

}

}

else{

StringBuilder temp2=new StringBuilder();

temp2.append(s.substring(i,s.length()));

temp2.reverse();

ans+=temp2.toString();

break;

}

}

return ans;

}

}

**Question** **6**

Given two strings s and goal, return true *if and only if* s *can become* goal *after some number of* ***shifts*** *on* s.

A **shift** on s consists of moving the leftmost character of s to the rightmost position.

* For example, if s = "abcde", then it will be "bcdea" after one shift.

**Example 1:**

**Input:** s = "abcde", goal = "cdeab"

**Output:**

True

Solve:- class Solution {

public String shift(String s){

String ans = "";

for(int i=1;i<s.length();i++){

ans += s.charAt(i);

if(i == s.length()-1){

ans += s.charAt(0);

}

}

return ans;

}

public boolean rotateString(String s, String goal) {

boolean ans = false;

int n = s.length();

while(n!=0){

if(s.equals(goal)){

ans = true;

break;

}else{

s = shift(s);

}

n--;

}

return ans;

}

}

**Question 7**

Given two strings s and t, return true *if they are equal when both are typed into empty text editors*. '#' means a backspace character.

Note that after backspacing an empty text, the text will continue empty.

**Example 1:**

**Input:** s = "ab#c", t = "ad#c"

**Output:** true

**Explanation:**

Both s and t become "ac".

Solve:-

class Solution {

public boolean backspaceCompare(String s, String t) {

return buildString(s).equals(buildString(t));

}

private String buildString(String str) {

Stack<Character> stack = new Stack<>();

for (char c : str.toCharArray()) {

if (c != '#') {

stack.push(c);

} else if (!stack.isEmpty()) {

stack.pop();

}

}

StringBuilder sb = new StringBuilder();

while (!stack.isEmpty()) {

sb.append(stack.pop());

}

return sb.toString();

}

}

**Question 8**

You are given an array coordinates, coordinates[i] = [x, y], where [x, y] represents the coordinate of a point. Check if these points make a straight line in the XY plane.

**Example 1:**

**Input:** coordinates = [[1,2],[2,3],[3,4],[4,5],[5,6],[6,7]]

**Output:** true

Solve:-

class Solution {

public boolean checkStraightLine(int[][] coordinates) {

int n = coordinates.length;

int x1 = coordinates[0][0];

int y1 = coordinates[0][1];

int x2 = coordinates[1][0];

int y2 = coordinates[1][1];

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

int x = coordinates[i][0];

int y = coordinates[i][1];

if ((y - y1) \* (x - x2) != (y - y2) \* (x - x1)) {

return false;

}

}

return true;

}

}