**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**

**Prgm:**

class Isomorphic\_String{

public static boolean isIsomorphic(String s, String t) {

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

return false;

int[] map1 = new int[256];

int[] map2 = new int[256];

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

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

return false;

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

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

}

return true;

}

public static void main(String[] args) {

String s = "egg";

String t = "add";

System.out.println(isIsomorphic(s,t));

}

}

**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**

**Prgm:**

class Strobogrammatic{

public static boolean isStrobogrammatic(String num) {

Map<Character, Character> map = new HashMap<Character, Character>();

map.put('6', '9');

map.put('9', '6');

map.put('0', '0');

map.put('1', '1');

map.put('8', '8');

int l = 0, r = num.length() - 1;

while (l <= r) {

if (!map.containsKey(num.charAt(l))) return false;

if (map.get(num.charAt(l)) != num.charAt(r))

return false;

l++;

r--;

}

return true;

}

public static void main(String[] args) {

String num = "69";

System.out.println(isStrobogrammatic(num));

}

}

**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"**

**Prgm:**

class Solution{

public static 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();

}

public static void main(String[] args) {

String num1 = "11";

String num2 = "123";

System.out.println(addStrings(num1,num2));

}

}

**Question 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"**

**Prgm:**

class Reverse{

public static String reverseWords(String s) {

int len = s.length();

if(len == 1)

return s;

int firstIndex, lastIndex;

char[] res = s.toCharArray();

char temp;

for(int index = 0 ; index < len ; index++){

firstIndex = index;

while(++index < len && res[index] != ' ');

lastIndex = index - 1;

while(firstIndex < lastIndex){

temp = res[firstIndex];

res[firstIndex++] = res[lastIndex];

res[lastIndex--] = temp;

}

return new String(res);

}

public static void main(String[] args) {

String s = "Let's take LeetCode contest";

System.out.println(reverseWords(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"**

**Prgm:**

class reverseString{

public static String reverseStr(String s, int k) {

char[] chars = s.toCharArray();

for (int i = 0; i < chars.length; i += (k << 1)) {

for (int st = i, ed = Math.min(chars.length - 1, i + k - 1); st < ed; ++st, --ed) {

char t = chars[st];

chars[st] = chars[ed];

chars[ed] = t;

}

}

return new String(chars);

}

public static void main(String[] args) {

String s = "abcdefg";

int k = 2;

System.out.println(reverseStr(s,k));

}

}

**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**

**Prgm:**

class Rotate\_STR{

public static boolean rotateString(String S, String goal) {

if (S == null) {

return goal == null;

}

if (S.length() == 0) {

return goal.length() == 0;

}

if (S.length() != goal.length()) {

return false;

}

char[] arrayA = S.toCharArray();

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

rotate(arrayA);

String rotatedA = String.valueOf(arrayA);

if (rotatedA.equals(goal)) {

return true;

}

}

return false;

}

private static void rotate(char[] A) {

char firstCh = A[0];

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

A[i - 1] = A[i];

}

A[A.length - 1] = firstCh;

}

public static void main(String[] args) {

String S = "abcde";

String goal = "cdeab";

System.out.println(rotateString(S,goal));

}

}

**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**

**Prgm:**

class Backspace{

public static boolean backspaceCompare(String S, String T) {

if (!getResStack(S).equals(getResStack(T)))

return false;

else

return true;

}

public static String getResStack(String s) {

StringBuilder sb = new StringBuilder();

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

if (c == '#'){

if (sb.length() > 0)

sb.deleteCharAt(sb.length() - 1);

} else {

sb.append(c);

}

}

return sb.toString();

}

public static void main(String[] args) {

String s = "ab#c";

String t = "ad#c";

System.out.println(backspaceCompare(s,t));

}

}

**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**

**Prgm:**

class Straight\_Line{

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

{

if(coordinates.length == 2)

return true;

int x0 = coordinates[0][0] , x1 = coordinates[1][0];

int y0 = coordinates[0][1] , y1 = coordinates[1][1];

int dx = x1 - x0 , dy = y1 - y0;

for(int i = 2 ; i < coordinates.length ; i++)

{

int x = coordinates[i][0] , y = coordinates[i][1];

if(dy \* (x - x0) != dx \* (y - y0))

return false;

}

return true;

}

public static void main(String args[])

{

int[][] coordinates = {{1 , 2} , {2 , 3} , {3 , 4} , {4 , 5} , {5 , 6},{6 , 7}};

System.out.println(checkStraightLine(coordinates) ? "true" : "false");

}

}