Assignment 7 Solution - String | DSA

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

Solution Code:

```
package in.ineuron.pptAssignment07;
public class IsomorphicStrings {

	public static boolean isIsomorphic(String s, String t) {
		if (s.length() != t.length()) {
			return false;
		}

		int[] sCharMap = new int[256];
		int[] tCharMap = new int[256];

		for (int i = 0; i < s.length(); i++) {
			char sChar = s.charAt(i);
			char tChar = t.charAt(i);

			if (sCharMap[sChar] != tCharMap[tChar]) {
				return false;
			}

			sCharMap[sChar] = i + 1;
			tCharMap[tChar] = i + 1;
		}

		return true;
	}

	public static void main(String[] args) {
		String s = "egg";
		String t = "add";
		boolean isIsomorphic = isIsomorphic(s, t);
		System.out.println(isIsomorphic); // Output: 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

Solution Code:

```
package in.ineuron.pptAssignment07;
public class StrobogrammaticNumber {

	public boolean isStrobogrammatic(String num) {
		int left = 0;
		int right = num.length() - 1;

		while (left <= right) {
			if (!isStrobogrammaticPair(num.charAt(left), num.charAt(right))) {
				return false;
			}

			left++;
			right--;
		}

		return true;
	}

	private boolean isStrobogrammaticPair(char c1, char c2) {
		return (c1 == '0' && c2 == '0') || (c1 == '1' && c2 == '1') || (c1 == '8' && c2 == '8')
				|| (c1 == '6' && c2 == '9') || (c1 == '9' && c2 == '6');
	}

	public static void main(String[] args) {
		StrobogrammaticNumber strobogrammaticNumber = new
		StrobogrammaticNumber();

		String num1 = "69";
		System.out.println(num1 + " is strobogrammatic: " +
		strobogrammaticNumber.isStrobogrammatic(num1));

		String num2 = "88";
		System.out.println(num2 + " is strobogrammatic: " +
		strobogrammaticNumber.isStrobogrammatic(num2));
```

```
        String num3 = "818";
        System.out.println(num3 + " is strobogrammatic: " +
        strobogrammaticNumber.isStrobogrammatic(num3));

        String num4 = "123";
        System.out.println(num4 + " is strobogrammatic: " +
        strobogrammaticNumber.isStrobogrammatic(num4));
    }
}
```

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"

Solution Code:

```
package in.ineuron.pptAssignment07;

public class AddStrings {
	public static String addStrings(String num1, String num2) {
		StringBuilder result = new StringBuilder();
		int i = num1.length() - 1;
		int j = num2.length() - 1;
		int carry = 0;

		while (i >= 0 || j >= 0 || carry > 0) {
			int digit1 = i >= 0 ? num1.charAt(i) - '0' : 0;
			int digit2 = j >= 0 ? num2.charAt(j) - '0' : 0;
			int sum = digit1 + digit2 + carry;
			result.insert(0, sum % 10);
			carry = sum / 10;
			i--;
			j--;
		}

		return result.toString();
	}

	public static void main(String[] args) {
		String num1 = "11";
		String num2 = "123";
		String sum = addStrings(num1, num2);
		System.out.println("Sum: " + sum);
	}
}
```

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"

Solution Code:

```
package in.ineuron.pptAssignment07;

public class ReverseWords {
	public static String reverseWords(String s) {
		String[] words = s.split(" ");
		StringBuilder result = new StringBuilder();

		for (String word : words) {
			StringBuilder reversedWord = new StringBuilder(word);
			result.append(reversedWord.reverse()).append(" ");
		}

		return result.toString().trim();
	}

	public static void main(String[] args) {
		String s = "Let's take LeetCode contest";
		String reversed = reverseWords(s);
		System.out.println("Reversed sentence: " + reversed);
	}
}
```

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"

Solution Code:

```
package in.ineuron.pptAssignment07;

public class ReverseString {
	public static String reverseStr(String s, int k) {
		char[] arr = s.toCharArray();
		int n = arr.length;

		for (int i = 0; i < n; i += 2 * k) {
			int start = i;
			int end = Math.min(i + k - 1, n - 1);

			while (start < end) {
				char temp = arr[start];
				arr[start] = arr[end];
				arr[end] = temp;
				start++;
				end--;
			}
		}

		return String.valueOf(arr);
	}

	public static void main(String[] args) {
		String s = "abcdefg";
		int k = 2;
		String reversed = reverseStr(s, k);
		System.out.println("Reversed string: " + reversed);
	}
}
```

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

Solution Code:

```
package in.ineuron.pptAssignment07;

public class StringShift {
	public static boolean canShift(String s, String goal) {
		if (s.length() != goal.length()) {
			return false;
		}

		String doubleS = s + s;
		return doubleS.contains(goal);
	}

	public static void main(String[] args) {
		String s = "abcde";
		String goal = "cdeab";
		boolean result = canShift(s, goal);
		System.out.println("Can shift: " + result);
	}
}
```

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

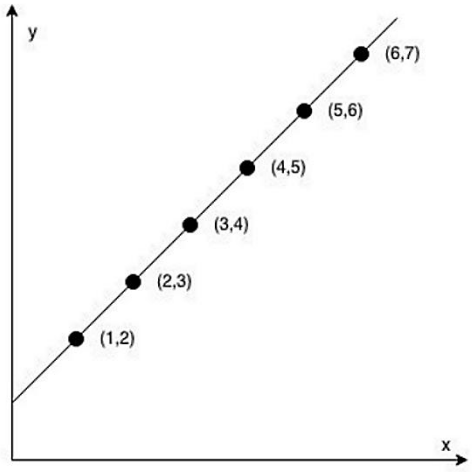
Explanation:
Both s and t become "ac".

Solution Code:

```
package in.ineuron.pptAssignment07;

public class BackspaceStringCompare {
	public static boolean backspaceCompare(String s, String t) {
		return buildString(s).equals(buildString(t));
	}

	private static String buildString(String str) {
		StringBuilder result = new StringBuilder();

		for (char c : str.toCharArray()) {
			if (c != '#') {
				result.append(c);
			} else if (result.length() > 0) {
				result.deleteCharAt(result.length() - 1);
			}
		}

		return result.toString();
	}

	public static void main(String[] args) {
		String s = "ab#c";
		String t = "ad#c";
		boolean result = backspaceCompare(s, t);
		System.out.println("Are equal: " + result);
	}
}
```

Question 8

You are given an array coordinate, 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

Solution Code:

```
package in.ineuron.pptAssignment07;

public class CheckStraightLine {
	public static boolean checkStraightLine(int[][] coordinates) {
		int deltaX = coordinates[1][0] - coordinates[0][0];
		int deltaY = coordinates[1][1] - coordinates[0][1];

		for (int i = 2; i < coordinates.length; i++) {
			int currDeltaX = coordinates[i][0] - coordinates[i - 1][0];
			int currDeltaY = coordinates[i][1] - coordinates[i - 1][1];

			if (deltaX * currDeltaY != deltaY * currDeltaX) {
				return false;
			}
		}

		return true;
	}
	public static void main(String[] args) {
		int[][] coordinates = { { 1, 2 }, { 2, 3 }, { 3, 4 }, { 4, 5 }, { 5, 6 }, { 6, 7 } };
		boolean result = checkStraightLine(coordinates);
		System.out.println("Is straight line: " + result);
	}
}
```