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

code:-
```
class Solution {
    public boolean isIsomorphic(String s, String t) {
        if (s == null || t == null)
        {
            return false;
        }
        if (s.length() == 0 && t.length() == 0)
        {
            return true;
        }
        if (s.length() != t.length())
        {
            return false;
        }
        HashMap<Character, Character> map = new HashMap<Character, Character>();
        for (int i = 0; i < s.length(); i++)
        {
            char c1 = s.charAt(i);
            char c2 = t.charAt(i);

            Character c = getKey(map, c2);
            if (c != null && c != c1)
            {
                return false;
            }
            else if (map.containsKey(c1))
            {
                if(c2 != map.get(c1))
                {
                    return  false;
                }
            }
            else
            {
                map.put(c1, c2);
            }
        }
        return true;
    }



public Character getKey(HashMap<Character, Character> map, Character target)
{
    for(Map.Entry<Character, Character> entry : map.entrySet())
    {
        if(entry.getValue().equals(target))
        {
            return entry.getKey();
        }
    }
    return null;
}

}
```

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

code:-

```
class Solution {
    public 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;
```

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"

code:-

```
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();
    }
}
```

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"

code:-
```
class Solution {
    public String reverseWords(String s) {
        String arr[] = s.split(" ");
        StringBuilder sb = new StringBuilder();
        for(String x : arr) {
            sb.append(reverse(x)).append(" ");
        }
        return sb.toString().trim();
    }
    public String reverse(String s) {
        int i = 0;
        int j = s.length() - 1;
        char arr[] = s.toCharArray();
        while(i < j) {
            char temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
            i++;
            j--;
        }
        return new String(arr);
    }
}
```

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"

code:-

```
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

code:-

```
class Solution {
    public boolean rotateString(String s, String goal) {
        if(s ==null || goal == null){
            return false;
        }
        if(s.length() != goal.length()) return false;
        if(s.length() == 0){
            return true;
        }
        int i =0, j =0;
        while(i < s.length() && j < goal.length()){
            if(s.charAt(i) == goal.charAt(j)){
                i++; j++;
            }
            else{
                if(j == 0){
                    i++;
                }
                else{
                    j= 0;
                }
            }
        }
        return (s.substring(0,goal.length() - j).equals(goal.substring(j)));
    }
}
```

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".

code:-

```
class Solution {
    public boolean backspaceCompare(String s, String t) {
        int sLen = s.length()-1;
        int tLen = t.length()-1;
        while(sLen >= 0 || tLen >= 0)
        {
            //sSkip
            int sSkip = 0;
            while(sLen >= 0)
            {
                if(s.charAt(sLen) == '#')
                {
                    sSkip++;
                    sLen--;
                }
                else if(sSkip > 0)
                {
                    sSkip--;
                    sLen--;
                }
                else
                {
                    break;
                }
            }
            //tSkip
            int tSkip = 0;
            while(tLen >= 0)
            {
                 if(t.charAt(tLen) == '#')
                {
                    tSkip++;
                    tLen--;
                }
                else if(tSkip > 0)
                {
                    tSkip--;
                    tLen--;
                }
                else
                {
                    break;
                }
            }

            //compare the current index.
            if(sLen >= 0 && tLen >= 0 && s.charAt(sLen) == t.charAt(tLen)) {
                sLen--;
                tLen--;
            }
            else{
                return sLen == -1 && tLen == -1;
            }


        }
        return true;
    }
}
```

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

code:-
```
class Solution {
    public boolean checkStraightLine(int[][] coordinates) {
        // Calculate the slope between the first two points
    int x0 = coordinates[0][0];
    int y0 = coordinates[0][1];
    int x1 = coordinates[1][0];
    int y1 = coordinates[1][1];

    // Iterate through the remaining points and check if the slope remains
consistent
    for (int i = 2; i < coordinates.length; i++) {
        int x = coordinates[i][0];
        int y = coordinates[i][1];
```

```
            // If the current point doesn't satisfy the slope formula, return false
            if ((y1 - y0) * (x - x0) != (y - y0) * (x1 - x0)) {
                return false;
            }
        }

        // All points satisfy the slope formula, so they form a straight line
        return true;
        }
    }
```