Answer 1)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

public class HelloWorld {
    public static int minimumDeleteSum(String s1, String s2) {
        int m = s1.length();
        int n = s2.length();

        int[][] dp = new int[m + 1][n + 1];

        // Calculate the ASCII sum for deleting characters in s1
        for (int i = 1; i <= m; i++) {
            dp[i][0] = dp[i - 1][0] + s1.charAt(i - 1);
        }

        // Calculate the ASCII sum for deleting characters in s2
        for (int j = 1; j <= n; j++) {
            dp[0][j] = dp[0][j - 1] + s2.charAt(j - 1);
        }

        // Calculate the minimum ASCII sum
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (s1.charAt(i - 1) == s2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1];
                } else {
                    dp[i][j] = Math.min(dp[i - 1][j] + s1.charAt(i - 1), dp[i][j - 1] + s2.charAt(j - 1));
                }
            }
        }

        return dp[m][n];
    }

    public static void main(String[] args) {
        String s1 = "sea";
        String s2 = "eat";
        System.out.println(minimumDeleteSum(s1, s2));  // Output: 231
    }
}
```

Answer 2)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

public class HelloWorld {
    public static boolean checkValidString(String s) {
        int minOpen = 0;  // Minimum number of open parentheses
        int maxOpen = 0;  // Maximum number of open parentheses

        for (char c : s.toCharArray()) {
            if (c == '(') {
                minOpen++;
                maxOpen++;
            } else if (c == ')') {
                minOpen = Math.max(minOpen - 1, 0);
                maxOpen--;
                if (maxOpen < 0) {
                    return false;  // More closing parentheses than opening parentheses
                }
            } else if (c == '*') {
                minOpen = Math.max(minOpen - 1, 0);
                maxOpen++;
            }
        }

        return minOpen == 0;
    }

    public static void main(String[] args) {
        String s = "()";
        System.out.println(checkValidString(s));  // Output: true
    }
}
```

Answer 3)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

public class HelloWorld {
    public static int minDistance(String word1, String word2) {
        int m = word1.length();
        int n = word2.length();

        // Create a 2D array to store the minimum number of steps
        int[][] dp = new int[m + 1][n + 1];

        // Initialize the base cases
        for (int i = 0; i <= m; i++) {
            dp[i][0] = i;
        }
        for (int j = 0; j <= n; j++) {
            dp[0][j] = j;
        }

        // Calculate the minimum number of steps
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (word1.charAt(i - 1) == word2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1];
                } else {
                    dp[i][j] = Math.min(dp[i - 1][j] + 1, dp[i][j - 1] + 1);
                }
            }
        }

        return dp[m][n];
    }

    public static void main(String[] args) {
        String word1 = "sea";
        String word2 = "eat";
        System.out.println(minDistance(word1, word2));  // Output: 2
    }
}
```

Answer 4)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

import java.util.*;
```

```
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int val) {
        this.val = val;
    }
}

public class ConstructBinaryTreeFromString {
    public static TreeNode str2tree(String s) {
        if (s.isEmpty()) {
            return null;
        }

        int i = 0;
        while (i < s.length() && (Character.isDigit(s.charAt(i)) || s.charAt(i) == '-')) {
            i++;
        }

        int num = Integer.parseInt(s.substring(0, i));
        TreeNode root = new TreeNode(num);

        if (i < s.length()) {
            int count = 0;
            int j = i;
            while (j < s.length()) {
                if (s.charAt(j) == '(') {
                    count++;
                } else if (s.charAt(j) == ')') {
                    count--;
                }

                if (count == 0) {
                    break;
                }

                j++;
            }

            root.left = str2tree(s.substring(i + 1, j));
            if (j + 1 < s.length()) {
                root.right = str2tree(s.substring(j + 2, s.length() - 1));
            }
        }

        return root;
    }

    public static void inorderTraversal(TreeNode root) {
        if (root == null) {
            return;
        }

        inorderTraversal(root.left);
        System.out.print(root.val + " ");
        inorderTraversal(root.right);
    }

    public static void main(String[] args) {
        String s = "4(2(3)(1))(6(5))";
        TreeNode root = str2tree(s);

        System.out.print("Inorder Traversal: ");
        inorderTraversal(root);
    }
}
```

Answer 5)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

public class HelloWorld {
    public static int compress(char[] chars) {
        int n = chars.length;
        int i = 0;
        int index = 0;

        while (i < n) {
            char currentChar = chars[i];
            int count = 0;

            while (i < n && chars[i] == currentChar) {
                i++;
                count++;
            }

            chars[index++] = currentChar;

            if (count > 1) {
                String countString = String.valueOf(count);

                for (char c : countString.toCharArray()) {
                    chars[index++] = c;
                }
            }
        }

        return index;
    }

    public static void main(String[] args) {
        char[] chars = {'a', 'a', 'b', 'b', 'c', 'c', 'c'};
        int compressedLength = compress(chars);

        System.out.print("Compressed Array: [");
        for (int i = 0; i < compressedLength; i++) {
            System.out.print("'" + chars[i] + "'");
            if (i != compressedLength - 1) {
                System.out.print(", ");
            }
        }
        System.out.println("]");
    }
}
```

Answer 6)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

import java.util.ArrayList;
import java.util.List;

public class HelloWorld {
    public static List<Integer> findAnagrams(String s, String p) {
        List<Integer> result = new ArrayList<>();

        if (s.length() < p.length()) {
            return result;
        }

        int[] pCount = new int[26];
        int[] sCount = new int[26];

        // Count the frequency of characters in string p
        for (char c : p.toCharArray()) {
            pCount[c - 'a']++;
        }

        // Initialize the sliding window
        for (int i = 0; i < p.length(); i++) {
            sCount[s.charAt(i) - 'a']++;
        }

        // Check each window of length p.length()
        for (int i = 0; i <= s.length() - p.length(); i++) {
            // Compare the frequency of characters in the current window with pCount
            if (matches(pCount, sCount)) {
                result.add(i);
            }

            // Slide the window by decrementing the count of the leftmost character
            sCount[s.charAt(i) - 'a']--;

            // Slide the window by incrementing the count of the next character
            if (i + p.length() < s.length()) {
                sCount[s.charAt(i + p.length()) - 'a']++;
            }
        }

        return result;
    }

    private static boolean matches(int[] pCount, int[] sCount) {
        for (int i = 0; i < 26; i++) {
            if (pCount[i] != sCount[i]) {
                return false;
            }
        }
        return true;
    }

    public static void main(String[] args) {
        String s = "cbaebabacd";
        String p = "abc";

        List<Integer> indices = findAnagrams(s, p);
        System.out.println("Start indices of p's anagrams in s: " + indices);
    }
}
```

Answer 7)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

import java.util.Stack;

public class HelloWorld {
    public static String decodeString(String s) {
        Stack<Integer> countStack = new Stack<>();
        Stack<String> stringStack = new Stack<>();
        StringBuilder currentString = new StringBuilder();
        int count = 0;

        for (char ch : s.toCharArray()) {
            if (Character.isDigit(ch)) {
                count = count * 10 + (ch - '0');
            } else if (ch == '[') {
                countStack.push(count);
                stringStack.push(currentString.toString());
                currentString = new StringBuilder();
                count = 0;
            } else if (ch == ']') {
                StringBuilder decodedString = new StringBuilder(stringStack.pop());
                int repeatCount = countStack.pop();
                for (int i = 0; i < repeatCount; i++) {
                    decodedString.append(currentString);
                }
                currentString = decodedString;
            } else {
                currentString.append(ch);
            }
        }

        return currentString.toString();
    }

    public static void main(String[] args) {
        String s = "3[a]2[bc]";
        String decodedString = decodeString(s);
        System.out.println("Decoded String: " + decodedString);
    }
}
```

Answer 8)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

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

        int firstMismatchIndex = -1;
        int secondMismatchIndex = -1;

        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) != goal.charAt(i)) {
                if (firstMismatchIndex == -1) {
                    firstMismatchIndex = i;
                } else if (secondMismatchIndex == -1) {
                    secondMismatchIndex = i;
                } else {
                    return false; // More than 2 mismatches, cannot be made equal by swapping
                }
            }
        }

        if (firstMismatchIndex != -1 && secondMismatchIndex != -1) {
            // Check if swapping the characters at the mismatched indices makes the strings equal
            return s.charAt(firstMismatchIndex) == goal.charAt(secondMismatchIndex) &&
                    s.charAt(secondMismatchIndex) == goal.charAt(firstMismatchIndex);
        }

        return false;
    }

    public static void main(String[] args) {
        String s = "ab";
        String goal = "ba";
        boolean canSwap = canSwapStrings(s, goal);
        System.out.println("Can swap strings? " + canSwap);
    }
}
```