Assignment No. 6

PAGE NO.
DATE:

Que 1. Answer:-

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

    int [][] dp = new int[m+1][n+1];
for(int i = 1; i <= m; i++){

    dp[i][0] = dp[i-1][0] + s1.charAt(i-1);

}
    for(int j = 1; j <= m; j++) {

    dp[0][j] = dp[0][j-1] + s2.charAt(j-1);

}
    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) {
Solution solution = new Solution();
String s1 = "sea";
String s2 = "eat";
int result = solution.minimumDeleteSum(s1, s2);
System.out.println(result);
}
}

Que 2 Answer:-

```
import java.util.Stack;
public class Solution {
public boolean checkValidString(String s) {

Stack<Integer> leftStack = new Stack<>();

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

    if (c == '(') {
        leftStack.push(i);
    }
    else if (c == '*')
    {
        starStack.push(i);
    } else {
        if (!leftStack.isEmpty()) {
            leftStack.pop();
        } else if (!starStack.isEmpty()) {
            starStack.pop();
        } else {
            return false;
        }
    }
```

```
while (!leftStack.isEmpty() && !starStack.isEmpty()) {
    if (leftStack.pop() > starStack.pop()) {
        return false;
    }
}
return leftStack.isEmpty();
}
public static void main(String[] args) {
    Solution solution = new Solution();
    String s = "( )";
    boolean isValid = solution.checkValidString(s);
    System.out.println(isValid);
}
}
```

Que 3. Answer :-

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

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

    for (int i=1; i<=m; i++) {
        dp[i][0] = i;
    }
    for (int j=1; j<=n; j++) {
        dp[0][j] = j;
    }
```

```
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], dp[i]
                                        [j-1]) + 1;
        }
    }
}
return dp[m][n];
}
public static void main(String[] aegr) {
    Solution solution = new Solution();
    String word1 = "sea";
    String word2 = "eat";
    int minStepr = solution.minDistance(
                                word1, word2);
    System.out.println(minStepr);
}
}
```

Que 4 Answer:-

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

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

```
public class Solution {
    public TreeNode str2tree(String s) {
    if (s.isEmpty()) {
        return null;
    }
int firstParen = s.indexOf("(");
int val = (firstParen == -1) ? Integer.parseInt(s) :
                    Integer.parseInt(s.substring
                        (0, firstParen));
TreeNode root = new
                    TreeNode(val);
if (firstParen == -1) {
    return root;
}
int start = firstParen, count = 0;
for (int i = start; i < s.length(); i++) {
    if (s.charAt(i) == '(') {
        count++;
    } else if (s.charAt(i) == ')') {
        count--;
    }
    if (count == 0 && start == firstParen) {
        root.left = str2tree(s.substring
                        (start + 1, i));
        start = i + 1;
    } else if (count == 0) {
        root.right = str2tree(s.substring(start + 1, i));
    }
}
return root;
}
public void inorderTraversal(TreeNode root) {
    if (root == null) {
        return;
    }
```

```
inorderTraversal(root.left);
System.out.println(root.val+" ");
inorderTraversal(root.right);
}
public static void main(String[] args){
    Solution solution = new Solution();
    String s = "4(2(3)(1))(6(5))";
    TreeNode root = solution.str2tree(s);
    solution.inorderTraversal(root);
}
}
```

The 5 Answer:

```
public class Solution {
    public int compress(char[] chars) {
        int anchor = 0;
        int writeIndex = 0;

        for (int readIndex = 0; readIndex < chars.length;
                                        readIndex++) {
            if (readIndex == chars.length - 1 || chars[readIndex]
                                != chars[readIndex + 1]) {
                chars[writeIndex++] = chars[anchor];
                if (readIndex > anchor) {

                    String count = String.valueOf(readIndex - anchor
                                                    + 1);
                    for (char c : count.toCharArray()) {
                        chars[writeIndex++] = c;
                    }
                }
                anchor = readIndex + 1;
            }
        }
        return writeIndex;
    }
```

```
public static void main (String [] args) {
    Solution solution = new Solution();
    char[] chars = {'a', 'a', 'b', 'b', 'c', 'c', 'c'};
    int newLength = solution.compress(chars);
    System.out.println("New length: " + newLength);

    System.out.println("New length: " + newLength);
    System.out.print("updated array: ");

    for (int i = 0; i < newLength; i++) {
        System.out.print(chars[i] + " ");
    }
}
}
```

Que 6. Answer

```
import java.util.ArrayList;
import java.util.List;
public class Solution {
    public List<Integer> findAnagrams(String s,
                                      String p) {
        List<Integer> result = new ArrayList<>();

        int[] pfreq = new int[26];
        int[] sfreq = new int[26];

        for (char c : p.toCharArray()) {
            pFreq[c - 'a']++;
        }
        int plength = p.length();
        int slength = s.length();

        for (int i = 0; i < plength; i++) {
            sfreq[s.charAt(i) - 'a']++;
        }
```

```
length - plength; i++) {
    if (Arrays.equals(pfreq, sfreq)) {
        result.add(i);
    }
    sfreq[s.charAt(i) - 'a']--;
    if (i + plength < slength) {
        sfreq[s.charAt(i + plength) - 'a']++;
    }
}
return result;
}
public static void main(String[] args) {
    Solution solution = new Solution();
    String s = "cbaebabacd";
    String p = "abc";
    List<Integer> indices = solution.findAnagrams(s, p);
    System.out.println(indices);
}
}
```

ue 7:- Answer:-

```
import java.util.Stack;
public class Solution {
    public class Solution {
        public String decodeString(String s) {
            Stack<Integer> numStack = new Stack<>();
            Stack<String> strStack = new Stack<>();
            StringBuilder currStr = new StringBuilder();
            int num = 0;
            for (char c : s.toCharArray()) {
                if (Character.isDigit(c)) {
                    num = num * 10 + (c - '0');
                } else if (c == '[') {
```

```
        numStack.push(num);
        strStack.push(currStr.toString());
        num = 0;
        currStr.setLength(0);
    } else if (c == ']') {
        int repeatTimes = numStack.pop();

        StringBuilder newStr = new StringBuilder
                                (strStack.pop());
        for (int i = 0; i < repeatTimes; i++) {
            newStr.append(currStr);
        }
        currStr = newStr;
    } else {
        currStr.append(c);
    }
    return currStr.toString();
}
public static void main(String[] args) {
    Solution solution = new Solution();
    String s = "3[a]2[bc]";
    System.out.println(solution.decodeString(s));
}
}
```

output

aaabcbc

Que :- → Answer :-

```
public class Solution {
    public boolean buddyStrings(String s, String goal) {
        if (s.length() != goal.length()) {
            return false;
        }
        if (s.equals(goal)) {
            int[] count = new int[26];
            for (char c : s.toCharArray()) {
                if (++count[c - 'a'] > 1) {
                    return true;
                }
            }
            return false;
        }
        int firstIndex = -1;
        int firstIndex = -1;

        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) != goal.charAt(i)) {
                if (firstIndex == -1) {
                    firstIndex = i;
                } else if (secondIndex == -1) {
                    secondIndex = i;
                } else {
                    return false;
                }
            }
        }
        return (secondInder != -1 && s.charAt
                (firstIndex) == goal.
                charAt(secondIndex))
```

```
        && s.charAt(secondIndex)
            == goal.charAt(firstIndex));
    }
    public static void main(String[] args){
        Solution solution = new Solution();
        String s = "ab";
        String goal = "ba";
        System.out.println(solution.buddyStrings
                                    (s, goal));
    }
}
```

output :-

true