Data Structure and Operations: Day-15-16(Tower of Hanoi Solver, Traveling Salesman Problem, Job Sequencing Problem):

Task 1: Knapsack Problem

Write a function int Knapsack(int W, int[] weights, int[] values) in C# that determines the maximum value of items that can fit into a knapsack with a capacity W. The function should handle up to 100 items. Find the optimal way to fill the knapsack with the given items to achieve the maximum total value. You must consider that you cannot break items, but have to include them whole.

```
package wipro.com.assignment11;
public class KnapsackProblem {
    // Function to determine the maximum value of items that can fit into the
knapsack
    public static int Knapsack(int W, int[] weights, int[] values) {
        int n = weights.length;
        int[][] dp = new int[n + 1][W + 1];

        // Build the dp array bottom-up
        for (int i = 1; i <= n; i++) {
            for (int w = 0; w <= W; w++) {
                if (weights[i - 1] <= w) {
                    dp[i][w] = Math.max(dp[i - 1][w], dp[i - 1][w - weights[i - 1]] +
values[i - 1]);
                } else {
                    dp[i][w] = dp[i - 1][w];
                }
            }
        }

        // The maximum value that can be put in a knapsack of capacity W is in
dp[n][W]
        return dp[n][W];
    }

    public static void main(String[] args) {
        int W = 50;  // Capacity of knapsack
        int[] weights = {10, 20, 30};  // Weights of items
        int[] values = {60, 100, 120};  // Values of items

        int maxValue = Knapsack(W, weights, values);
        System.out.println("The maximum value of items that can fit into the knapsack
is: " + maxValue);
    }
}
```

Output:

```
The two non-repeating elements are: 5 and 3
```

Task 2: Longest Common Subsequence

Implement int LCS(string text1, string text2) to find the length of the longest common subsequence between two strings.

```
package wipro.com.assignment11;

public class LongestCommonSubsequence {
    // Function to find the length of the longest common subsequence
    public static int LCS(String text1, String text2) {
        int m = text1.length();
        int n = text2.length();

        // Create a 2D array to store the lengths of longest common subsequence
        int[][] dp = new int[m + 1][n + 1];

        // Build the dp array from bottom-up
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (text1.charAt(i - 1) == text2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1] + 1;
                } else {
                    dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
                }
            }
        }

        // The length of the longest common subsequence is in dp[m][n]
        return dp[m][n];
    }

    public static void main(String[] args) {
        String text1 = "abcde";
        String text2 = "ace";

        int lcsLength = LCS(text1, text2);
        System.out.println("The length of the longest common subsequence is: " +
lcsLength);
    }
}
```

Output:

```
The length of the longest common subsequence is: 3
```