Day 24 Assignment

Name: Mehul Anjikhane Email: mehulanjikhane13@gmail.com

Task 1: Knapsack Problem

Write a function int Knapsack(int W, int[] weights, int[] values) in Java 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 algorithm;

import java.util.Arrays;

public class KnapsackProblem {

     public static int knapsack(int W, int[] weights, int[] values)
{
                int n = weights.length;
                int[][] dp = new int[n + 1][W + 1];

                // Initialize first row and column for base cases
                for (int i = 0; i <= W; i++) {
                           dp[0][i] = 0; // No items, max value is 0
                }

                for (int i = 1; i <= n; i++) {
                      dp[i][0] = 0; // Capacity 0, max value is 0
                }

                // Build DP table
                for (int i = 1; i <= n; i++) {
                           for (int w = 1; w <= W; w++) {
                                      if (weights[i - 1] > w) {
                                      // If weight exceeds capacity,
inherit value from previous item
                                            dp[i][w] = dp[i - 1][w];
                                      } else {
                                                 // Choose the max
value: include or exclude current item
                                            dp[i][w] = Math.max(dp[i
- 1][w], values[i - 1] + dp[i - 1][w - weights[i - 1]]);
                                      }
                           }
                }
```

```
            return dp[n][W];
        }

    public static void main(String[] args) {
            int W = 30;
            int[] weights = { 10, 20, 30 };
            int[] values = { 60, 100, 120 };

            System.out.println("Knapsack Capacity: " + W);
            System.out.println("Weights of Items: " +
Arrays.toString(weights));
            System.out.println("Values of Items: " +
Arrays.toString(values));

            int maxValue = knapsack(W, weights, values);
            System.out.println("Maximum value in knapsack: " +
maxValue);
        }
}
```

Output:

```
Knapsack Capacity: 30
Weights of Items: [10, 20, 30]
Values of Items: [60, 100, 120]
Maximum value in knapsack: 160
```

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 algorithm;

public class LongestCommonSubsequence {

    public static int lcs(String text1, String text2) {
            int m = text1.length();
            int n = text2.length();
            int[][] dp = new int[m + 1][n + 1];

            // Initialize first row and column for base cases
            for (int i = 0; i <= m; i++) {
                dp[i][0] = 0; // Empty first string, LCS is 0
            }
            for (int j = 0; j <= n; j++) {
                dp[0][j] = 0; // Empty second string, LCS is 0
            }
            // Build DP table
            for (int i = 1; i <= m; i++) {
                    for (int j = 1; j <= n; j++) {
                            if (text1.charAt(i - 1) ==
text2.charAt(j - 1)) {
            // Characters match, consider the previous match + 1
                                dp[i][j] = dp[i - 1][j -
1] + 1;
                            } else {
            // Characters don't match, take the max LCS from
excluding either character
                                    dp[i][j] =
Math.max(dp[i - 1][j], dp[i][j - 1]);
                            }
                    }
            }
            return dp[m][n];
        }

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

        System.out.println("String 1: "+text1+", String 2: " +
text2);
        int lcsLength = lcs(text1, text2);

        System.out.println("Length of Longest Common Subsequence:
" + lcsLength);
        }
}
```

Output:

```
String 1: AGGTAB, String 2: GXTXAYBA
Length of Longest Common Subsequence: 4
```