Himanshu Kamane
kamanehimanshu76@gmail.com

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.

Solution:

```
package com.app;
public class Knapsack {
  public static int knapsack(int W, int[] weights, int[] values) {
    int n = weights.length;
    // dp[i][j] will store the maximum value that can be attained with weight <= j
using the first i items
    int[][] dp = new int[n + 1][W + 1];
    // Build the dp array from bottom up
    for (int i = 1; i <= n; i++) {
      for (int w = 0; w <= W; w++) {
        if (weights[i - 1] <= w) {
          // If the current item's weight is less than or equal to the current
capacity w
          dp[i][w] = Math.max(dp[i - 1][w], dp[i - 1][w - weights[i - 1]] +
values[i - 1]);
        } else {
          // If the current item's weight is more than the current capacity w
          dp[i][w] = dp[i - 1][w];
        }
      }
    }
    // The answer will be in dp[n][W], which means using all n items and capacity
W
```

```
        return dp[n][W];
    }
    public static void main(String[] args) {
        int W = 50; // Knapsack capacity
        int[] weights = {10, 20, 30};
        int[] values = {60, 100, 120};
        System.out.println("Maximum value in Knapsack = " + knapsack(W, weights,
values));
    }
}
```

Output:

```
Maximum value in Knapsack = 220
```

Task 2: Longest Common Subsequence
Implement int LCS(string text1, string text2) to find the length of the longest common subsequence between two strings.

Solution:

```
package com.app;
public class LongestCommonSubsequence {
    public static int LCS(String text1, String text2) {
        int m = text1.length();
        int k = text2.length();
        // dp[i][j] will store the length of LCS of text1[0...i-1] and text2[0...j-1]
        int[][] dp = new int[m + 1][k + 1];
        // Build the dp array from bottom up
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= k; 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 LCS will be in dp[m][k]
    return dp[m][k];
  }
  public static void main(String[] args) {
    String text1 = "abcde";
    String text2 = "ace";
    System.out.println("Length of Longest Common Subsequence = " +
LCS(text1, text2));
  }
}
```

Output:

```
Length of Longest Common Subsequence = 3
```