Digvijay Thakare
Day 15-16
digvijaythakare2017@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.

Code

```
package WiproEP;
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];
        // Build table dp[][] in bottom up manner
        for (int i = 0; i <= n; i++) {
            for (int w = 0; w <= W; w++) {
                if (i == 0 || w == 0) {
                    dp[i][w] = 0;
                } else if (weights[i - 1] <= w) {
                    dp[i][w] = Math.max(values[i - 1] + dp[i -
1][w - weights[i - 1]], dp[i - 1][w]);
                } else {
                    dp[i][w] = dp[i - 1][w];
                }
            }
        }
        // dp[n][W] contains the maximum value that can be put in
a knapsack of capacity W
        return dp[n][W];
    }
    public static void main(String[] args) {
        int W = 50; // Knapsack capacity
        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("Maximum value that can be obtained: "
+ maxValue);
    }
}
```

Output
```
Maximum value that can be obtained: 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.**

Code
```
package WiproEP;
public class LongestCommonSubsequence {

    public static int LCS(String text1, String text2) {
        int m = text1.length();
        int n = text2.length();

        // Create a table to store the lengths of LCS for
substrings
        int[][] dp = new int[m + 1][n + 1];

        // Fill the dp table in a bottom-up manner
        for (int i = 0; i <= m; i++) {
            for (int j = 0; j <= n; j++) {
                if (i == 0 || j == 0) {
                    dp[i][j] = 0;
                } else 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]);
                }
            }
        }
```

```java
        // The length of the longest common subsequence is stored
in dp[m][n]
        return dp[m][n];
    }
    public static void main(String[] args) {
        String text1 = "abcde";
        String text2 = "ace";
        int lengthOfLCS = LCS(text1, text2);
        System.out.println("Length of Longest Common Subsequence:
" + lengthOfLCS);
    }
}
```

Output

```
Length of Longest Common Subsequence: 3
```