Name : Tejaswini Anil Kamble

Email : teju000kamble@gmail.com

Assignments : Day 15 and 16:

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.

Ans : Source Code

```java
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];

        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];
                }}}

        return dp[n][W];
    }

    public static void main(String[] args) {
        int W = 8;
        int[] values = {1, 2, 5, 6};
        int[] weights = {2, 3, 4, 5};

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

Output:

```
package com.wipro.da1516;

Console
<terminated> KnapsackProblem [Java Application] C:\Program Files\Java\jdk-17
Maximum value that can be obtained: 8
```

Task 2: Longest Common Subsequence

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

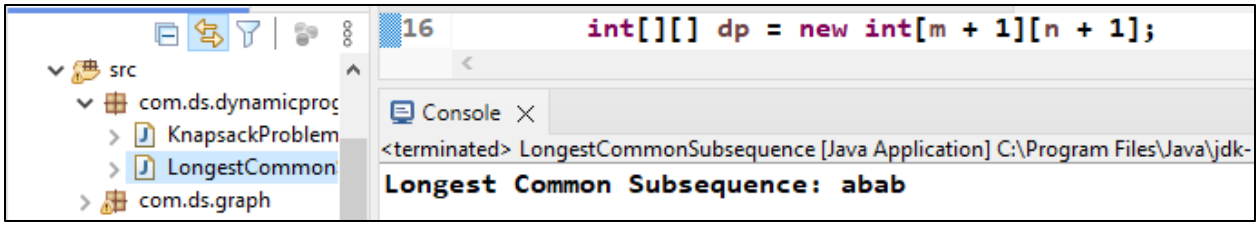
Ans : Source Code

FS_JavaProgramming - Data_strutures/src/com/ds/dynamicprogramming/LongestCommonSubsequence.java - Eclipse IDE

```java
package com.ds.dynamicprogramming;

public class LongestCommonSubsequence {

    public static void main(String[] args) {
        String str1 = "babbab";
        String str2 = "abaaba";

        String lcs = longestCommonSubsequence(str1, str2);
        System.out.println("Longest Common Subsequence: " + lcs);
    }

    private static String longestCommonSubsequence(String str1, String str2) {
        int m = str1.length();
        int n = str2.length();
        int[][] dp = new int[m + 1][n + 1];

        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 (str1.charAt(i - 1) == str2.charAt(j - 1)) {
                    dp[i][j] = 1 + dp[i - 1][j - 1];
                } else {
                    dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
                }
            }
        }
        StringBuilder lcsBuilder = new StringBuilder();
        int i = m, j = n;
        while (i > 0 && j > 0) {
```

```java
        StringBuilder lcsBuilder = new StringBuilder();
        int i = m, j = n;
        while (i > 0 && j > 0) {
            if (str1.charAt(i - 1) == str2.charAt(j - 1)) {
                lcsBuilder.insert(0, str1.charAt(i - 1));
                i--;
                j--;
            } else if (dp[i - 1][j] > dp[i][j - 1]) {
                i--;
            } else {
                j--;
            }
        }

        return lcsBuilder.toString();
    }
}
```

Output:


src
com.ds.dynamicprog
KnapsackProblem
LongestCommon
com.ds.graph
16 int[][] dp = new int[m + 1][n + 1];
Console
<terminated> LongestCommonSubsequence [Java Application] C:\Program Files\Java\jdk-
Longest Common Subsequence: abab