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.

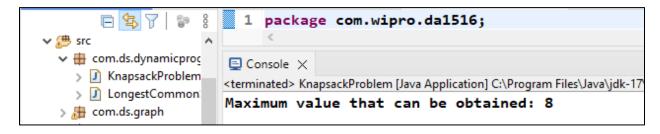
Ans: Source Code

```
🌉 FS_JavaProgramming - DSA_JavaAssignments/src/com/wipro/da1516/KnapsackProblem.java - Eclipse IDE
 File Edit Source Refactor Navigate Search Project Run Window
Q 😭 😢 🐉
🚹 Project Explorer 🗴 📅 🗖 🚺 TrieNodejava 🚺 Triejava 📗 BitManipula... 🖟 UniqueElemen... 🖟 Swap8its.java 🖟 CountSet8it... 🖟 KnapsackPro... 🗴 🖟 KnapsackPro... 🗡 🤼 KnapsackPro...
                                                                                                                                                                                            - - -
          strutures $ 2 2 strutures $ 3 import java.util.Arrays;
 > ➡ JRE System Library [Java

✓ ∰ src
     > ∰ com.ds.dynamicproc 6⊝
> ∰ com.ds.graph 7
> ∰ com.ds.linkedlist 8
                                        public static int knapsack(int W, int[] weights, int[] values) {
      > 🔠 com.ds.graph
> 🔠 com.ds.linkedlist
                                             int n = weights.length;
int[][] dp = new int[n + 1][W + 1];
      > 🌐 com.ds.patterns
      > # com.ds.queue
> # com.ds.searching_alç
                                             for (int i = 0; i <= n; i++) {
                              11
12
                                                    for (int w = 0; w <= W; w++)
if (i == 0 || w == 0) {
      > 🔠 com.ds.sortingAlgori
                                                        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]);</pre>
      → ∰ com.ds.stack
      > # com.ds.tree
      > # com.wipro.day12
> # com.wipro.graphalge
                                                             dp[i][w] = dp[i - 1][w];
      > 🔠 computalgo
  > I module-info.java
                              19
20
21
                                             return dp[n][W];
   > M JRE System Library [Java
   ✓ ∰ src
✓ ∰ com.wipro.da1516
                                       }
                                        public static void main(String[] args) {
        > In KnapsackProblem
                              23⊖
     > # com.wipro.day11
> # com.wipro.day20
                                              int[] values = {1, 2, 5, 6};
                              26
27
      > A com.wipro.day7and8
                                             int[] weights = {2, 3, 4, 5};
      > # com.wipro.generic
> # com.wipro.linkedlist
                               28
                                              int maxValue = knapsack(W, weights, values);
      → ∰ com.wipro.nonlinear
                                              System.out.println("Maximum value that can be obtained: " + maxValue);
       com.wipro.threads
module-info.java
                               30
                               31 }
      primes.txt
                              32
```

Output:



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
File Edit Source Refactor Navigate Search Project Run Window Help
 🎦 Project Explorer 🗶 📅 🗖 🔃 KnapsackProblem01.java 🔑 KnapsackProblem.java 🖟 LongestCommonSubsequence.java 🗴
       E 🕏 7 | 🔊 8
                         1 package com.ds.dynamicprogramming;
  3 public class LongestCommonSubsequence {

→ 

⊕ com.ds.dynamicproc

      >    LongestCommon
                        5⊝
                                public static void main(String[] args) {
    > Æ com.ds.graph
                                     String str1 = "babbab";
String str2 = "abaaba";
    > 🔠 com.ds.linkedlist
    > # com.ds.patterns
                         8
    > 🌐 com.ds.queue
                         9
                                     String lcs = longestCommonSubsequence(str1, str2);
    > 🌐 com.ds.searching_alç
                         10
                                     System.out.println("Longest Common Subsequence: " + lcs);
    > 🔠 com.ds.sortingAlgori
                         11
    > # com.ds.stack
                         12
    > # com.ds.timeAndSpac
                        13⊖
                                private static String longestCommonSubsequence(String str1, String str2) {
    > # com.ds.tree
                                    int m = str1.length();
    > 🌐 com.wipro.day12
                         15
                                   int n = str2.length();
    > 🔠 com.wipro.graphalgc
                         16
                                     int[][] dp = new int[m + 1][n + 1];
    > 🔠 computalgo
     module-info.java
                                     for (int i = 0; i <= m; i++) {
 for (int j = 0; j <= n; j++) {
   if (i == 0 || j == 0) {
                         19
  > 🔼 JRE System Library [Java
                         20

✓ 

Æ src

                                                  dp[i][j] = 0;
    21
                         22
                                              } else if (str1.charAt(i - 1) == str2.charAt(j - 1)) {
      > M KnapsackProblem
    > # com.wipro.day11
                         23
                                                  dp[i][j] = 1 + dp[i - 1][j - 1];
    > # com.wipro.day20
                         24
                                              } else {
    > # com.wipro.day7and8
                         25
                                                  dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
    > # com.wipro.generic
                         26
    > # com.wipro.linkedlist
                         27
                                         }
      e com.wipro.nonlinear
                         28

⊕ com.wipro.threads

                         29
                                     StringBuilder lcsBuilder = new StringBuilder();
      module-info.java
                         30
                                     int i = m, j = n;
    primes.txt
                      v 31
                                     while (i > 0 && j > 0) {
```

```
StringBuilder lcsBuilder = new StringBuilder();
   > # com.wipro.day12
                                     int i = m, j = n;
   > A com.wipro.graphalgo
                         31
                                     while (i > 0 && j > 0) {
   > # computalgo
                                         if (str1.charAt(i - 1) == str2.charAt(j - 1)) {
    module-info.java
                         33
                                              lcsBuilder.insert(0, str1.charAt(i - 1));
34
 > 🛋 JRE System Library [Java
                         35
                                              i--:
                                         } else if (dp[i - 1][j] > dp[i][j - 1]) {
   36
     > 🕖 KnapsackProblem
                         37
                                             i--;
   > # com.wipro.day11
                         38
                                         } else {
   > # com.wipro.day20
                         39
                                              j--;
   > 🔠 com.wipro.day7and8
                         40
   > # com.wipro.generic
   > 🌐 com.wipro.linkedlist
                         42
     com.wipro.nonlinear
                         43
                                     return lcsBuilder.toString();
     # com.wipro.threads
                        44
     module-info.java
                       45 }
```

Output:

