

# RPS DAY 15-16 Assignments

## Assignment 7

**Name: Akshada Baad**

**Batch - CPPE**

**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.**

```
public class Knapsack {  
    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 = 50;
```

```
    int[] weights = {10, 20, 30};
```

```
    int[] values = {60, 100, 120};
```

```
    System.out.println("Maximum value in Knapsack = " + knapsack(W, weights, values));
```

```
}
```

```
}
```

```
Wipro.java - Java/src/com/wipro/java/Knapsack.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer x Knapsack.java x Outline Console x
DataStructures
Java
  JRE System Library [J
  src
    com.wipro.java
      BalancedBinari
      BFS.java
      BitManipulatio
      BoyerMoore.java
      CircularQueue
      DFS.java
      Dijkstra.java
      Edge.java
      Graph.java
      Graph1.java
      GraphCycle.jav
      Job.java
      JobSequencing
      Knapsack.java
      LinkedListMid
      ListNode.java
      MergeSortedList
      MinHeap.java
      NaivePatternSe
      QueueSorting
      RabinKarp.java
      RemoveDuplicat
      StackSequence
      StackSorting.ja
      StringOperatio
      TowerOfHanoi
      TravelingSales
      TreeNode.java
      Trie.java
      TrieNode.java
      UnionFind.java

1 package com.wipro.java;
2
3 public class Knapsack {
4     public static int knapsack(int W, int[] weights, int[] values) {
5         int n = weights.length;
6         int[] dp = new int[n + 1][W + 1];
7
8         for (int i = 0; i <= n; i++) {
9             for (int w = 0; w <= W; w++) {
10                 if (i == 0 || w == 0) {
11                     dp[i][w] = 0;
12                 } else if (weights[i - 1] <= w) {
13                     dp[i][w] = Math.max(values[i - 1] + dp[i - 1][w - weights[i - 1]],
14                                         dp[i - 1][w]);
15                 } else {
16                     dp[i][w] = dp[i - 1][w];
17                 }
18             }
19         }
20         return dp[n][W];
21     }
22
23     public static void main(String[] args) {
24         int W = 50;
25         int[] weights = {10, 20, 30};
26         int[] values = {60, 100, 120};
27         System.out.println("Maximum value in Knapsack = " + knapsack(W, weights, values));
28     }
29 }
30

Problems x JavaDoc Declaration
0 errors, 40 warnings, 0 others
Description Resource Path Location Type
Warnings (40 items)

Terminated - Knapsack [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\java.exe (05-Jun-2024, 10:21)
Maximum value in Knapsack = 220

Writabla Smart Insert 30:1:918
ENG IN 1021 05-06-2024
```

## Task 2: Longest Common Subsequence

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

```
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];  
  
        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]);  
                }  
            }  
        }  
  
        return dp[m][n];  
    }  
  
    public static void main(String[] args) {  
        String text1 = "AGGTAB";  
        String text2 = "GXTXAYB";  
        System.out.println("Length of LCS is " + lcs(text1, text2)); // Expected output: 4  
    }  
}
```

}

```
1 package com.wipro.java;
2
3 public class LongestCommonSubsequence {
4     public static int lcs(String text1, String text2) {
5         int m = text1.length();
6         int n = text2.length();
7         int[][] dp = new int[m + 1][n + 1];
8
9         for (int i = 0; i <= m; i++) {
10             for (int j = 0; j <= n; j++) {
11                 if (i == 0 || j == 0) {
12                     dp[i][j] = 0;
13                 } else if (text1.charAt(i - 1) == text2.charAt(j - 1)) {
14                     dp[i][j] = dp[i - 1][j - 1] + 1;
15                 } else {
16                     dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
17                 }
18             }
19         }
20         return dp[m][n];
21     }
22
23     public static void main(String[] args) {
24         String text1 = "AGGTAB";
25         String text2 = "GXTXBA";
26         System.out.println("Length of LCS is " + lcs(text1, text2)); // Expected output
27     }
28 }
29
30
```

Console Output:

```
-terminated: LongestCommonSubsequence [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe
Length of LCS is 4
```

Problems: 0 errors, 40 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (40 items)				