Day 15 and 16 Assignment

1. **Write a function int Knapsack(int W, int[] weights, int[] values) in Java 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.**

```java
package m5_core_java_programming.day_19_part_2;

/*
   Write a function int Knapsack(int W, int[] weights, int[] values) in Java
   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.
*/

import java.util.Arrays;

public class Assignment_1 {

   public static int Knapsack(int W, int[] weights, int[] values) {
       int[][] dp = new int[W + 1][values.length + 1];

       for (int i = 0; i < W + 1; i++) {
           Arrays.fill(dp[i], -1);
       }

       int res = findMaxVal(W, weights, values, values.length, dp);

       return (res >= 0) ? res : 0;
   }

   private static int findMaxVal(int W, int[] weights, int[] values, int n,
int[][] dp) {
       if (W < 0) {
           return 0;
       }
       if (n <= 0) {
           return 0;
       }
       if (weights[n - 1] > W) {
           return findMaxVal(W, weights, values, n - 1, dp);
       }
       dp[W][n] = Math.max(findMaxVal(W, weights, values, n - 1, dp),
findMaxVal(W - weights[n - 1], weights, values, n - 1, dp) + values[n - 1]);
```

```
        return dp[W][n];
    }

    public static void main(String[] args) {
        int[] weights = {2, 3, 4, 5, 9, 7, 8, 6, 1, 2};
        int[] values = {3, 4, 8, 8, 10, 7, 12, 5, 1, 6};
        int W = 15;
        System.out.println("Maximum value for 15kg of weight limit is " +
Knapsack(W, weights, values));
    }
}
```

Output

```
C:\Users\coolr\.jdks\openjdk-22.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains
Weights : [2, 3, 4, 5, 9, 7, 8, 6, 1, 2]
Values : [3, 4, 8, 8, 10, 7, 12, 5, 1, 6]
Maximum value for 15kg of weight limit is 27

Process finished with exit code 0
```

---

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

```
package m5_core_java_programming.day_19_part_2;

/*
   Implement int LCS(string text1, string text2)  to find the length of the
   longest common subsequence between two strings.
*/
public class Assignment_2 {

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


        int[][] dp = new int[n + 1][m + 1];

        for (int i = 0; i < n + 1; i++) {
            for (int j = 0; j < m + 1; 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[n][m];
    }

    public static void main(String[] args) {
        String text1 = "ACCGGTCGAGTGCGCGGAAGCCGGCCGAA";
        String text2 = "GTCGTTCGGAATGCCGTTGCTCTGTAAA";

        System.out.println(text1);
        System.out.println(text2);
        System.out.println("Length longest subsequence for the give two Strings
is " + LCS(text1, text2));
    }
}
```

Output

```
C:\Users\coolr\.jdks\openjdk-22.0.1\bin\java.exe "-javaagent:C:\Program File
ACCGGTCGAGTGCGCGGAAGCCGGCCGAA
GTCGTTCGGAATGCCGTTGCTCTGTAAA
Length longest subsequence for the give two Strings is 20

Process finished with exit code 0
```

Tools Used :
IntelliJ IDE
java version "1.8.0_411"
Java(TM) SE Runtime Environment (build 1.8.0_411-b09)
Java HotSpot(TM) Client VM (build 25.411-b09, mixed mode, sharing)