Day 15 and 16:

Task 1: Knapsack Problem

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.

```
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 = 1; i <= n; i++) {
   for (int w = 1; w <= W; w++) {
    if (weights[i - 1] <= w) {
     dp[i][w] = Math.max(dp[i - 1][w], values[i - 1] + dp[i - 1][w - weights[i - 1]]);
    }
    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};

    int maxValue = knapsack(W, weights, values);

    System.out.println("The maximum value of items that can fit in the knapsack is: " + maxValue);

  }

}
```

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 = 1; i <= m; i++) {
      for (int j = 1; j <= n; j++) {
        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";
    int lcsLength = LCS(text1, text2);
    System.out.println("Length of LCS between '" + text1 + "' and '" + text2 + "' is: " +
lcsLength);
  }
}
```