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.

Task 2: Longest Common Subsequence

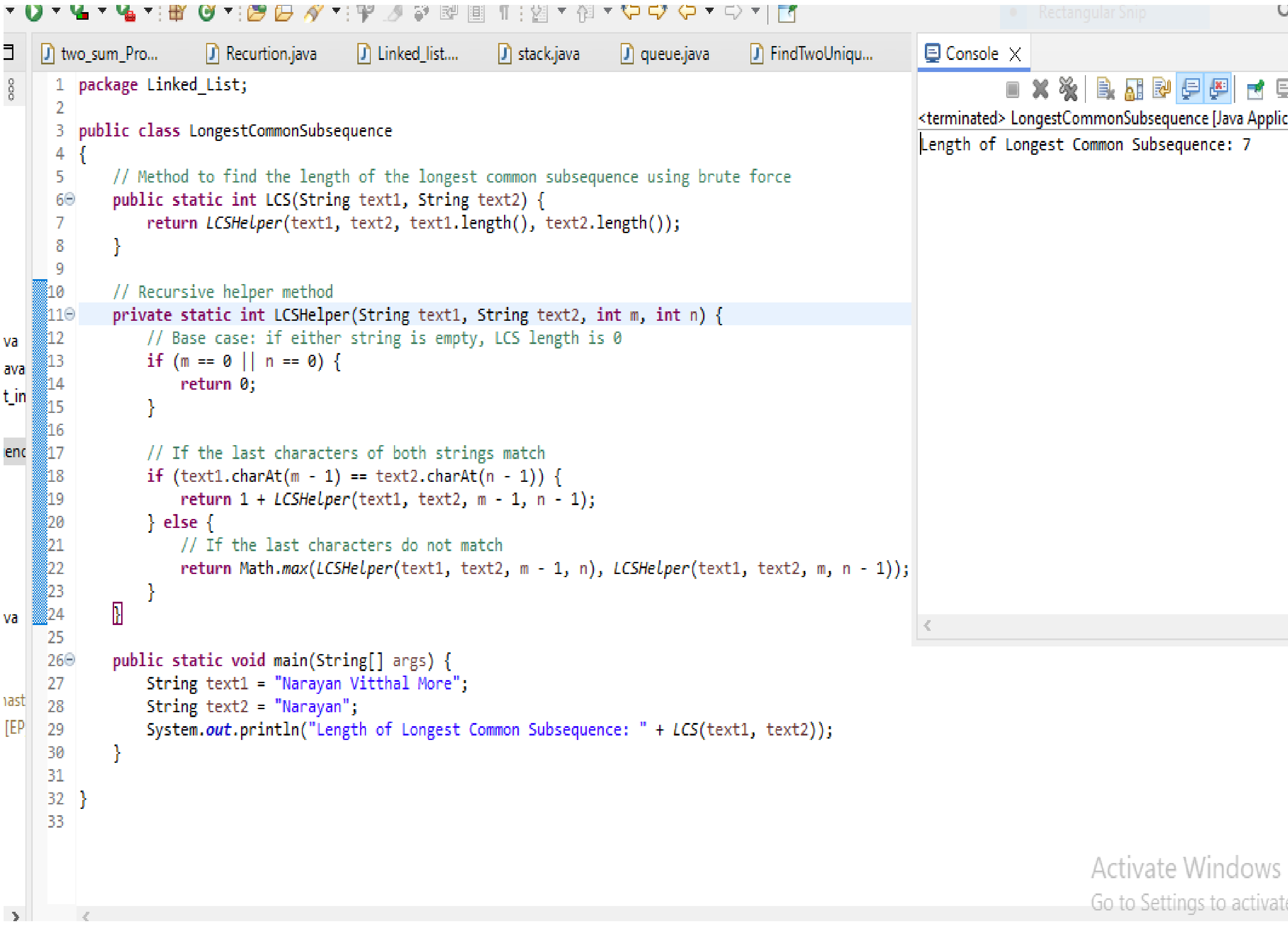
Implement int LCS(string text1, string text2) to find the length of the longest common subsequence between two strings.

1. **Base Case**:
   - If either text1 or text2 is empty (m == 0 or n == 0), the LCS length is 0.
2. **Recursive Case**:
   - If the last characters of both strings match (text1.charAt(m - 1) == text2.charAt(n - 1)), then these characters are part of the LCS. We recursively call the helper function for the remaining parts of both strings (m - 1 and n - 1) and add 1 to the result.
   - If the last characters do not match, we recursively call the helper function twice: once excluding the last character of text1 (m - 1) and once excluding the last character of text2 (n - 1). We take the maximum of these two results.
3. **Main Method**:
   - Create two strings, text1 and text2.
   - Call the LCS method to find the length of the LCS and print the result.

Example

Given text1 = "Narayan More" and text2 = "Narayan":

- The LCS is "Narayan".
- The length of the LCS is 7.



```java
package Linked_List;

public class LongestCommonSubsequence
{
    // Method to find the length of the longest common subsequence using brute force
    public static int LCS(String text1, String text2) {
        return LCSHelper(text1, text2, text1.length(), text2.length());
    }

    // Recursive helper method
    private static int LCSHelper(String text1, String text2, int m, int n) {
        // Base case: if either string is empty, LCS length is 0
        if (m == 0 || n == 0) {
            return 0;
        }

        // If the last characters of both strings match
        if (text1.charAt(m - 1) == text2.charAt(n - 1)) {
            return 1 + LCSHelper(text1, text2, m - 1, n - 1);
        } else {
            // If the last characters do not match
            return Math.max(LCSHelper(text1, text2, m - 1, n), LCSHelper(text1, text2, m, n - 1));
        }
    }

    public static void main(String[] args) {
        String text1 = "Narayan Vitthal More";
        String text2 = "Narayan";
        System.out.println("Length of Longest Common Subsequence: " + LCS(text1, text2));
    }

}
```

```
<terminated> LongestCommonSubsequence [Java Applic
Length of Longest Common Subsequence: 7
```