Task 1: Knapsack Problem

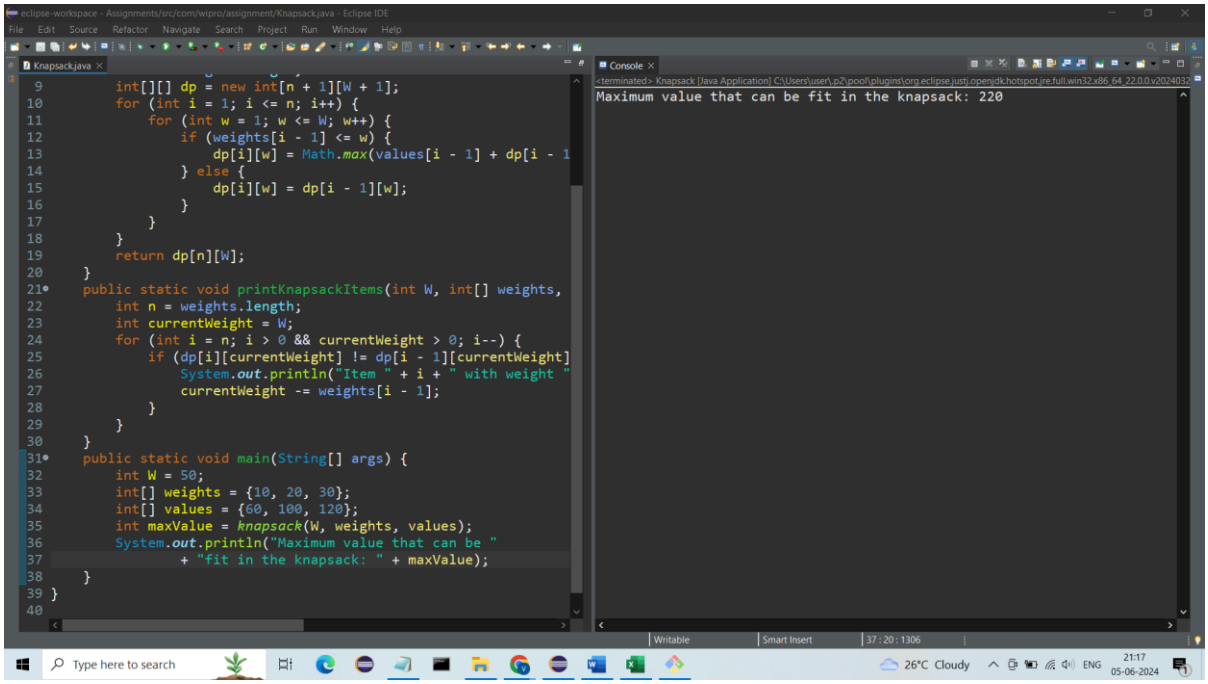
Write a function int Knapsack(int W, int[] weights, int[] values) in java that determines the maximum value of items that can fit into a knapsack with a capacity W. The function should handle up to 100 items. Find the optimal way to fill the knapsack with the given items to achieve the maximum total value. You must consider that you cannot break items, but have to include them whole.

Function: -

```java
public static int knapsack(int W, int[] weights, int[] values) {
    if (weights == null || values == null || weights.length != values.length || weights.length > 100) {
        throw new IllegalArgumentException("Invalid input arguments");
    }
    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(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];
}
```

```java
public static void printKnapsackItems(int W, int[] weights, int[] values, int[][] dp) {
    int n = weights.length;
    int currentWeight = W;
    for (int i = n; i > 0 && currentWeight > 0; i--) {
        if (dp[i][currentWeight] != dp[i - 1][currentWeight]) {
            System.out.println("Item " + i + " with weight " + weights[i - 1] + " and value " + values[i - 1] + " is picked.");
            currentWeight -= weights[i - 1];
        }
    }
}
```

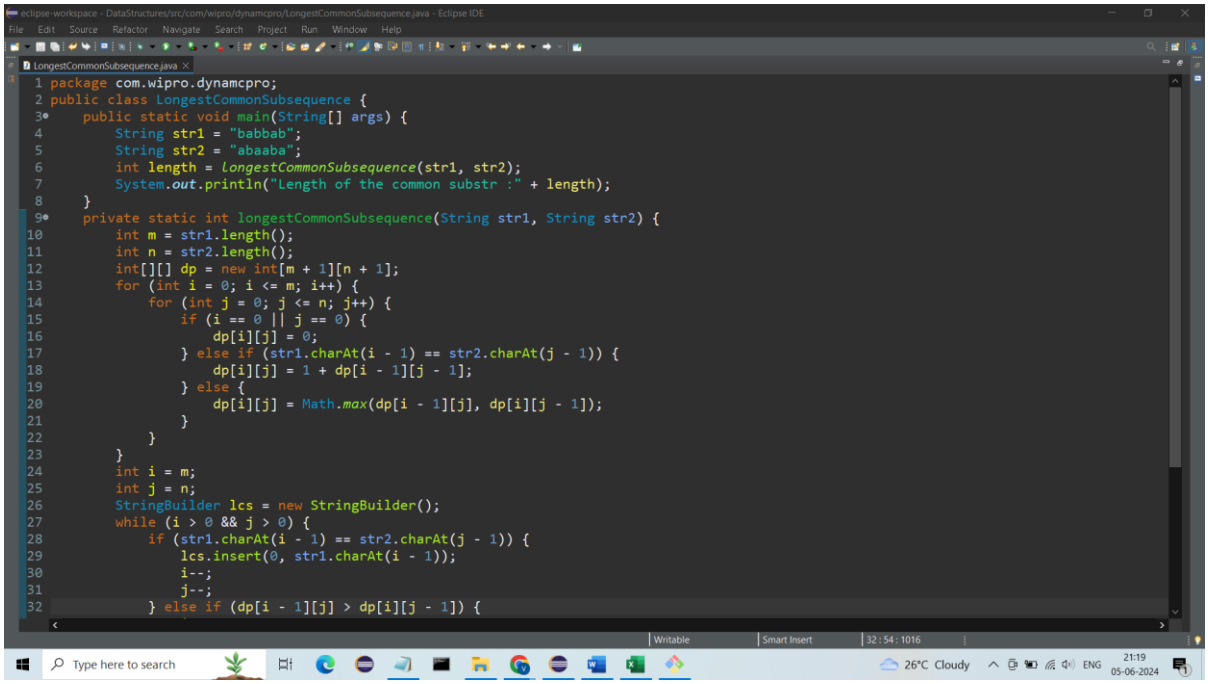
Output: -



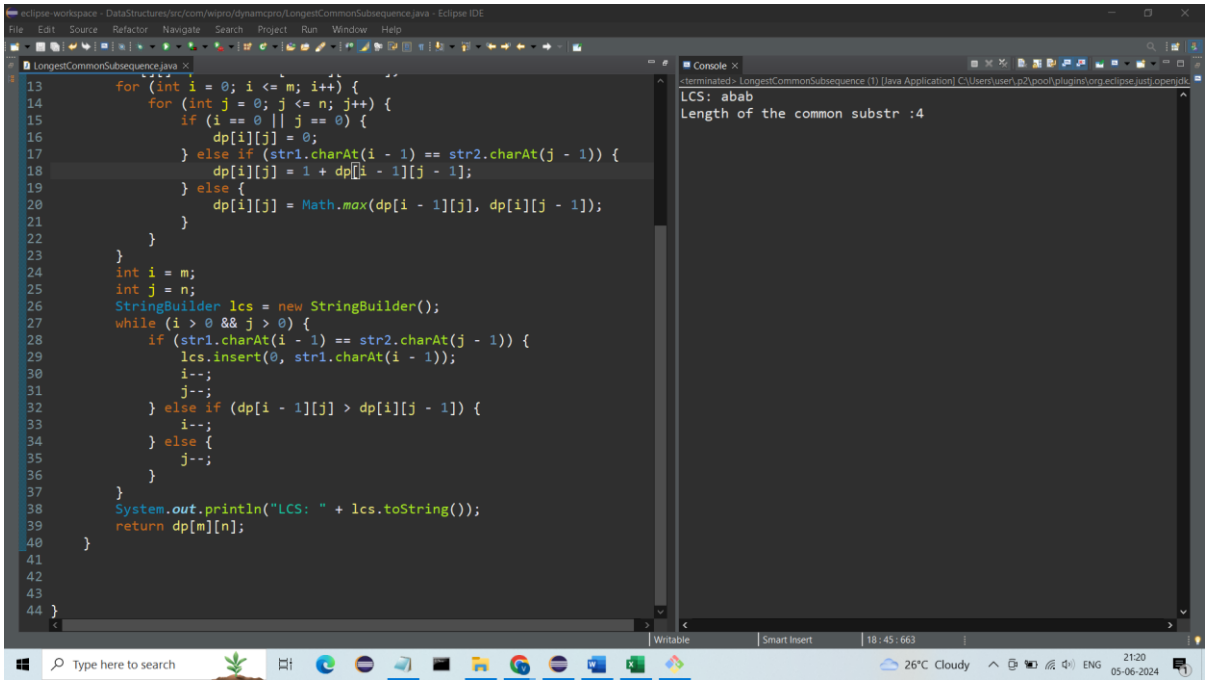
Task 2: Longest Common Subsequence

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

Program: -



Output: -


eclipse-workspace - DataStructures/src/com/wipro/dynamcpro/LongestCommonSubsequence.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
LongestCommonSubsequence.java
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 (str1.charAt(i - 1) == str2.charAt(j - 1)) {
dp[i][j] = 1 + dp[i - 1][j - 1];
} else {
dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
}
}
}
int i = m;
int j = n;
StringBuilder lcs = new StringBuilder();
while (i > 0 && j > 0) {
if (str1.charAt(i - 1) == str2.charAt(j - 1)) {
lcs.insert(0, str1.charAt(i - 1));
i--;
j--;
} else if (dp[i - 1][j] > dp[i][j - 1]) {
i--;
} else {
j--;
}
}
System.out.println("LCS: " + lcs.toString());
return dp[m][n];
}
}
Console
<terminated> LongestCommonSubsequence (1) [Java Application] C:\Users\user\.p2\pool\plugins\org.eclipse.justj.openjdk.
LCS: abab
Length of the common substr :4
Writable
Smart Insert
18 : 45 : 663
Type here to search
26°C Cloudy
ENG
21:20
05-06-2024