LCS Implementation using Dynamic Programming

The longest common subsequence consists of the longest sequence in the order they are arranged in an array, but it may or may not be continuous. For example:

X = [A, B, C, B, D, A, B] and Y = [B, D, C, A, B, A], then the program's goal is to get a LCS and store it in a new array Z. Thus Z will contain the array [B, C, B, A].

The way the program works would be to store two 2D arrays b and c with the directions and the values respectively. Each cell in 'b' will store the direction for a combination of an element in X and Y. 'c' will store the value for the same combination, except it will have an extra row and column at the beginning that will store the value 0 by default.

If the combination of X and Y happen to have the same element, the value will store the value of the variable at the top left + 1, that is the reason an extra row and column has the value of 0, to take care of the base case. The direction will be "top-left" and "tl" will be stored in 'b' in that cell.

For the remaining cases, the number above the cell in question will be compared to the number to the left of the cell in 'c'. The maximum of those two numbers will be stored in that cell in array 'c', this demonstrates the dynamic programming principle where a value of a cell depends on the value of another cell, here filling the value of that other cell is a subproblem.

The direction for that cell in 'b' will point in the direction of the higher value of the cell. It can be "top" or "left". If both the values are identical then the direction can point either "top" or "left", both will result in a correct solution. As the for loops to traverse the entire 2D array runs from 1 to the length of m times n, it is a bottom up approach. This will take O(mn) running time due to two loops.

The whole point of the exercise is when the print_lcs function is called that takes the initial parameters as m and n, the 'b' array will help to navigate to reach the common elements as the direction will determine whether the next function cell will take the parameters as row-1 or column-1 or both. When a diagonal direction is reached in the 'b' array the subsequence gets stored in the stack. As the LCS gets stored in the stack in the reverse order, when the stack is popped towards the end of the program execution, the LCS will be printed in the correct order. The running time for print_lcs is O(m+n) as it will either decrement the row or column or both in every step.