Program Name- To find the length of the longest common increasing subsequence [LCIS] and print one of such sequences (multiple sequences may exist)

Prerequisites-

Longest Common Subsequence - *http://www.geeksforgeeks.org/dynamic-programming-set-4-longest-common-subsequence/*

Longest Increasing Subsequence- http://www.geeksforgeeks.org/dynamic-programming-set-3-longest-increasing-subsequence/

Project Category- Dynamic Programming

Programming Paradigm Used- Dynamic Programming

Examples-

Suppose we consider two arrays -
arr1[] = {3, 4, 9, 1} and arr2[] = {5, 3, 8, 9, 10, 2, 1}

We have to find the longest subsequence which is common in both the arrays and which is increasing also.

If we have been asked about the longest common subsequence only then our answer would have been- 3 as there exists a subsequence – {3, 9, 1} which is common in both the arrays.

On the other hand if we have been asked the longest common subsequence which is increasing then our answer will change as {3, 9, 1} is common but not increasing subsequence.

Our answer would become {3, 9} as this is the longest common subsequence which is increasing also.

Algorithm-

We will use dynamic programming to find the length of longest common increasing subsequence.

We will store the longest common increasing sub-sequence ending at each index of arr2[].
For that we use two iterating variables – i and j, where i traverses all the element of arr1[] and j through all the elements of arr2[].

So, we check that when both the array elements are same (i.e- arr1[i] == arr2[j]). And when this condition gets true then there is a common subsequence ending at j. So there is a possibility that this common subsequence is the longest. So we check this by the condition (if (current + 1) > table[j]). If this condition is true then we update table[j] with the new value and to keep track of the longest increasing subsequence ending at j by setting parent[j] = last

The second if condition is when jth indexed element of arr2[] is smaller than the ith indexed element of arr1[]. If this condition gets satisfied then we update current as the maximum of the current and the length of the LCIS ending at index j.

Then after the completion of filling the DP table- table[], we find the maximum element of table[] array. This element is the length of the LCIS and the index of this element is the ending point of the LCIS

To print the longest common increasing subsequence we will keep track of the parent of each element in the longest common increasing subsequence.

Time Complexity-

O(NM), where N and M are the lengths of the arrays.

Space Complexity-

We will create a 1-Dimensional DP table to store the results of sub-problems. Therefore we will use O(M) extra space, where M = length of arr2[]