// C++ program to find the longest repeated

// subsequence

#include <bits/stdc++.h>

using namespace std;

// This function mainly returns LCS(str, str)

// with a condition that same characters at

// same index are not considered.

string longestRepeatedSubSeq(string str)

{

    // THIS PART OF CODE IS SAME AS BELOW POST.

    // IT FILLS dp[][]

    // OR the code mentioned above.

    int n = str.length();

    int dp[n+1][n+1];

    for (int i=0; i<=n; i++)

        for (int j=0; j<=n; j++)

            dp[i][j] = 0;

    for (int i=1; i<=n; i++)

        for (int j=1; j<=n; j++)

            if (str[i-1] == str[j-1] && i != j)

                dp[i][j] =  1 + dp[i-1][j-1];

            else

                dp[i][j] = max(dp[i][j-1], dp[i-1][j]);

    // THIS PART OF CODE FINDS THE RESULT STRING USING DP[][]

    // Initialize result

    string res = "";

    // Traverse dp[][] from bottom right

    int i = n, j = n;

    while (i > 0 && j > 0)

    {

        // If this cell is same as diagonally

        // adjacent cell just above it, then

        // same characters are present at

        // str[i-1] and str[j-1]. Append any

        // of them to result.

        if (dp[i][j] == dp[i-1][j-1] + 1)

        {

           res = res + str[i-1];

           i--;

           j--;

        }

        // Otherwise we move to the side

        // that that gave us maximum result

        else if (dp[i][j] == dp[i-1][j])

            i--;

        else

            j--;

    }

    // Since we traverse dp[][] from bottom,

    // we get result in reverse order.

    reverse(res.begin(), res.end());

    return res;