```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
```

Cyclic Longest Common Subsequence:

```cpp
// maximum of lcs(any cyclic shift of s, any cyclic shift of t) in O(n*m)
const int N = 2010;
int dp[N * 2][N], from[N * 2][N];

// it is work for string or for normal vector
// vector<int> s , t;
string s , t;

int yo() {
    int n = s.size(), m = t.size();
    auto eq = [&](int a, int b) {
        return s[(a - 1) % n] == t[(b - 1) % m];
    };
    dp[0][0] = from[0][0] = 0;
    for (int i = 0; i <= n * 2; ++i) {
        for (int j = 0; j <= m; ++j) {
            dp[i][j] = 0;
            if (j && dp[i][j - 1] > dp[i][j]) {
                dp[i][j] = dp[i][j - 1];
                from[i][j] = 0;
            }
            if (i && j && eq(i, j) && dp[i - 1][j - 1] + 1 > dp[i][j]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
                from[i][j] = 1;
            }
            if (i && dp[i - 1][j] > dp[i][j]) {
                dp[i][j] = dp[i - 1][j];
                from[i][j] = 2;
            }
        }
    }
    int ret = 0;
    for (int i = 0; i < n; ++i) {
        ret = max(ret, dp[i + n][m]);
        // re-root
        int x = i + 1, y = 0;
        while (y <= m && from[x][y] == 0) ++y;
        for (; y <= m && x <= n * 2; ++x) {
            from[x][y] = 0, --dp[x][m];
            if (x == n * 2) break;
            for (; y <= m; ++y) {
                if (from[x + 1][y] == 2) break;
                if (y + 1 <= m && from[x + 1][y + 1] == 1) {
                    ++y;
                    break;
                }
            }
        }
    }
    return ret;
}
```