```
#include <iostream>
#include <vector>
#include <string>
using namespace std;

// A function to find the maximum number of substrings that can be removed from the string
int findMaxSubstringsRemoved(string mainStr, const vector<string>& subStrs) {
    int maxRemovals = 0;  // Variable to keep track of the most substrings removed

    // Iterate over all possible substrings
    for (const string& sub : subStrs) {
        size_t pos = mainStr.find(sub);  // Try to locate the substring in the main string

        if (pos != string::npos) {  // If the substring is found
            string updatedStr = mainStr;  // Create a copy of the current string
            updatedStr.erase(pos, sub.length());  // Remove the substring from the copied string

            // Recursively calculate the number of substrings that can be removed from the updated
string
            maxRemovals = max(maxRemovals, 1 + findMaxSubstringsRemoved(updatedStr,
subStrs));
        }
    }

    return maxRemovals;  // Return the result
}

int main() {
    int numSubstrings;  // Number of substrings to check for removal
    cin >> numSubstrings;  // Read the number of substrings

    vector<string> substrings(numSubstrings);  // A container to hold the substrings
    for (int i = 0; i < numSubstrings; i++) {
        cin >> substrings[i];  // Read each substring
    }

    string mainString;  // The main string from which substrings will be removed
    cin >> mainString;  // Read the main string

    // Call the function to compute and print the result
    int result = findMaxSubstringsRemoved(mainString, substrings);
    cout << result;  // Output the maximum number of substrings that can be removed

    return 0;
}
```