Problem Solving (CPP21)

هذا البرنامج التدريبي مُصاغ بعناية لتمكين المتدربين من تطوير قدراتهم الفكرية على غرار المبرمجين المحترفين، والتعاون بكفاءة ضمن فريق محترف في شركة "جمال تك" أو أي مؤسسة متعددة الجنسيات أخرى. نظرًا لأهمية اللغة الإنجليزية في بيئة العمل العالمية، يتم تقديم المحتوى التدريبي بالإنجليزية. لا يشترط إتقان اللغة بشكل كامل، لكن من الضروري امتلاك القدرة الكافية لفهم المتطلبات وتنفيذها بشكل فعّال. يُمكن للمتدربين استخدام مترجم جوجل أو الاستعانة بـ "شات جي بي تي" للتغلب على أية عقبات لغوية، المهم هو الفهم الدقيق للمطلوب وتحقيقه بنجاح.

لتعظيم الاستفادة من التدريب، يُنصح بمحاولة حل التمارين بشكل مستقل لمدة ساعة واحدة على الأقل قبل الرجوع إلى الحل المرفق في نهاية الملف.

قد يتضمن الحل كودًا برمجيًا غير مفسر بعد، والغرض من ذلك هو تشجيعك على محاولة فهم الأكواد البرمجية الجديدة التي لم تتعرض لها من قبل. هذه المهارة ضرورية في سوق العمل، حيث تتطور لغات البرمجة باستمرار ويظهر كل يوم لغات جديدة. ستواجه دائمًا أكوادًا لم تدرسها من قبل، ومن المهم أن تكون قادرًا على فهمها بنفسك دون الحاجة إلى دراسة مسبقة. يمكنك الاستعانة بمحرك البحث جوجل، أو استخدام ChatGPT، أو حتى اللجوء لأصدقائك للمساعدة. الهدف الأساسي هو أن تصل إلى فهم معنى كل كود بأي طريقة ممكنة لتتمكن من إيجاد موقعك في سوق العمل.

إن وجود كود برمجي غير مفسر يشكل تحديًا يتوجب عليك إيجاد حل له. هذا النوع من التدريبات يعد جزءًا أساسيًا من تدريبات 'Problem Solving'، التي تهدف إلى تمكينك من أداء عملك بفاعلية بغض النظر عن التحديات والعقبات. هذه القدرة على حل المشكلات هي ما يتمتع به العاملون في 'جمال تك'، ومن الضروري أن تطور في نفسك هذه المهارة لتصبح عضوًا فعالًا في فريق عمل 'جمال تك'.

Gammal Tech's Smart Auto-Correct System

Problem Statement:

Gammal Tech, a pioneering leader in software innovation, has developed a cutting-edge Smart Auto-Correct System. This system is designed to revolutionize how we interact with digital text inputs. Your task is to contribute to this project by creating an algorithm that enhances the system's string manipulation capabilities.

The Smart Auto-Correct System works as follows: Given an input word, the system first checks if the word is correctly spelled based on an internal dictionary. If the word is correct, it prints the word as it is. However, if the word is misspelled, the system intelligently generates a list of possible correct alternatives based on the input.

Input Format:

> The first line contains an integer `N`, the number of words in Gammal Tech's dictionary.
> The next `N` lines each contain a single word from the dictionary.

The next line contains an integer `Q`, the number of queries.
The following `Q` lines each contain a single query word.

Output Format:

For each query, if the word is correct according to the dictionary, print the word. If the word is incorrect, print a comma-separated list of possible correct alternatives. If there are no alternatives, print `"No suggestions"`.

Constraints:

- $1 \leq N \leq 10{,}000$
- $1 \leq Q \leq 1{,}000$
- Each word in the dictionary and each query is a lowercase string of length between 1 and 100.

Sample Input:

```
5
algorithm
binary
computer
data
network
3
algoritm
date
networks
```

Sample Output:

```
algorithm
data
No suggestions
```

Explanation:

- "algoritm" is a misspelling of "algorithm".
- "date" is a misspelling of "data".
- "networks" has no close match in the dictionary.

لتعظيم الاستفادة من التدريب، يُنصح بمحاولة حل التمرين بشكل مستقل لمدة ساعة واحدة على الأقل قبل الرجوع إلى الحل المرفق

C++ Programming Solution:

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

// Function to calculate the edit distance between two strings
int editDistance(const string &a, const string &b) {
    int m = a.size(), n = b.size();
    vector<vector<int>> dp(m + 1, vector<int>(n + 1));

    for (int i = 0; i <= m; i++) dp[i][0] = i;
    for (int j = 0; j <= n; j++) dp[0][j] = j;

    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (a[i - 1] == b[j - 1])
                dp[i][j] = dp[i - 1][j - 1];
            else
                dp[i][j] = 1 + min({dp[i - 1][j], dp[i][j - 1], dp[i -
1][j - 1]});
        }
    }
    return dp[m][n];
}
```

```cpp
int main() {
    int N;
    cin >> N;
    vector<string> dictionary(N);
    for (int i = 0; i < N; i++) {
        cin >> dictionary[i];
    }

    int Q;
    cin >> Q;
    while (Q--) {
        string query;
        cin >> query;
        bool found = false;
        vector<string> suggestions;

        for (const string &word : dictionary) {
            if (word == query) {
                cout << query << endl;
                found = true;
                break;
            } else if (editDistance(word, query) == 1) {
                suggestions.push_back(word);
            }
        }

        if (!found) {
            if (suggestions.empty()) {
                cout << "No suggestions" << endl;
            } else {
                for (size_t i = 0; i < suggestions.size(); i++) {
                    if (i > 0) cout << ",";
                    cout << suggestions[i];
                }
                cout << endl;
            }
        }
    }
    return 0;
}
```