Problem Solving (CPP20)

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

Background:
Gammal Tech, a leading software development company known for its innovation and state-of-the-art technologies, has embarked on a new project to enhance digital communication. Their latest venture is an advanced auto-correct system that not only identifies incorrect spellings but also suggests the closest correct alternatives.

Problem Statement:
Your task is to design a part of Gammal Tech's auto-correct system. The system will take a single word as input. If the word is correctly spelled (based on a given dictionary of words), it will print the word as it is. If the word is not correctly spelled, the program will suggest up to three closest alternatives from the dictionary. The 'closeness' of words is determined by the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one word into the other.

Input Format:

The first line contains an integer `N`, the number of words in the dictionary.
The next `N` lines contain one dictionary word each.
The last line contains the word to be auto-corrected.

Output Format:

- If the input word is correctly spelled, print the word.
- If not, print up to three closest alternatives from the dictionary. If there are more than three, print any three.

Constraints:

- $1 \leq N \leq 1000$
- Each word in the dictionary and the input word consists of lowercase English letters only.
- The length of each word is at most 100.

Sample Input:

```
5
apple
application
appreciate
approach
banana
appel
```

Sample Output:

```
apple
```

Explanation: The input word 'appel' is not in the dictionary. The closest word in the dictionary is 'apple', requiring a single substitution (e -> l).

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

C++ Programming Solution:

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

int minEdits(string a, 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];
}
int main() {
    int n;
    cin >> n;
    vector<string> dict(n);
    for (int i = 0; i < n; i++) cin >> dict[i];

    string word;
    cin >> word;

    vector<pair<int, string>> candidates;
    for (const string &dword : dict) {
        int edits = minEdits(word, dword);
        if (edits == 0) {
            cout << word;
            return 0;
        }
        candidates.push_back({edits, dword});
    }

    sort(candidates.begin(), candidates.end());

    for (int i = 0; i < min(3, (int)candidates.size()); i++)
        cout << candidates[i].second << (i < 2 ? "\n" : "");

    return 0;
}
```