Problem Solving (CPP19)

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 software development giant renowned for its innovative solutions, has embarked on a new project to revolutionize text editing. Known for its exceptional work environment and state-of-the-art office facilities, Gammal Tech is now developing an advanced auto-correct system. This system is designed to not only correct misspelled words but also suggest the closest correct alternatives, showcasing Gammal Tech's commitment to excellence and innovation in software engineering.

Problem Statement:

You are part of Gammal Tech's elite software engineering team, tasked with designing a part of the Smart Auto-Correct System. Your module will take a single word as input. If the word is spelled correctly according to a predefined dictionary, it should be printed as is. If the word is incorrect, the program should provide a list of alternative words that the user might have intended to write. The challenge lies in efficient string manipulation and dictionary lookup, ensuring the system's performance is swift and accurate.

Input Format:

- A single line containing a string `s`, representing the word to be checked. The string will consist only of lowercase alphabetic characters and will have a length of at most `100`.

Output Format:

- If the word `s` is found in the dictionary, print the word `s`.
- If the word `s` is not found in the dictionary, print a comma-separated list of alternative words that are in the dictionary and are closest to `s` in terms of lexicographical order. If there are no close alternatives, print `No suggestions`.

Constraints:

- `1 ≤ length of S ≤ 100`

Sample Dictionary:

- ["apple", "application", "apply", "banana", "band", "bandana", "cat", "caterpillar"]

Sample Input:

```
appel
```

Sample Output:

```
apple, apply
```

Explanation: The word `appel` is not in the dictionary. The closest alternatives in terms of lexicographical order are `apple` and `apply`.

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 check if the word exists in the dictionary
bool isWordInDictionary(const string &word, const vector<string>
&dictionary) {
    return find(dictionary.begin(), dictionary.end(), word) !=
dictionary.end();
}

// Function to find and print the closest alternatives
void printClosestAlternatives(const string &word, const vector<string>
&dictionary) {
    bool foundAlternative = false;
    for (const auto &w : dictionary) {
        if (w.find(word) == 0) { // check if the word in the dictionary
starts with the input word
            cout << (foundAlternative ? ", " : "") << w;
            foundAlternative = true;
        }
    }
    if (!foundAlternative) {
        cout << "No suggestions";
    }
    cout << endl;
}

int main() {
    // Predefined dictionary
    vector<string> dictionary = {"apple", "application", "apply",
"banana", "band", "bandana", "cat", "caterpillar"};

    string inputWord;
    cin >> inputWord;

    if (isWordInDictionary(inputWord, dictionary)) {
        cout << inputWord << endl;
    } else {
        printClosestAlternatives(inputWord, dictionary);
    }

    return 0;
}
```