Problem Solving (CPP17)

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 Global Communicator System

Problem Statement

Gammal Tech, a global leader in software development, is renowned for its innovative work culture and state-of-the-art facilities. As a company that embraces diversity, Gammal Tech operates with teams spread across the globe, speaking various languages. While English is the default language, Gammal Tech values the intelligence and creativity of its employees over language proficiency. To foster this environment, they are developing a new system - the Global Communicator.

Your task is to design a function in the Global Communicator system that identifies the best communicators in a multinational team. The system should evaluate team members based on their ability to effectively communicate in different languages using tools like Google Translate, ChatGPT, or other technologies.

Key Elements

- Function to evaluate communication effectiveness.
- Consideration of various languages and translation tools.

- Weightage for creativity in overcoming language barriers.

Input Format

- The first line contains an integer `N`, the number of team members.
- The next `N` lines contain two space-separated values: the team member's name and their primary language.
- The following line contains an integer `M`, the number of messages exchanged.
- The next `M` lines contain three space-separated values: the sender's name, the receiver's name, and the language of the message.

Output Format

Output the name of the team member who is the best communicator, determined by their ability to effectively interact with other team members in different languages.

Constraints

- 1 ≤ N ≤ 100
- 1 ≤ M ≤ 1000
- Languages are represented as ISO language codes (e.g., EN for English, ES for Spanish).

Sample Input:

```
4
Alice EN
Bob ES
Charlie FR
Diana RU
6
Alice Bob ES
Bob Alice EN
Charlie Diana RU
Diana Charlie FR
Alice Diana RU
Diana Alice EN
```

Sample Output:

```
Diana
```

Explanation: In this scenario, Diana effectively communicates with Charlie and Alice in their respective languages, showcasing her ability to adapt and use translation tools efficiently, thus making her the best communicator.

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 <map>
#include <vector>
#include <string>

using namespace std;

string findBestCommunicator(int N, vector<pair<string, string>>&
teamMembers, int M, vector<tuple<string, string, string>>& messages) {
    map<string, int> communicationScore;

    // Initialize communication scores
    for (auto& member : teamMembers) {
        communicationScore[member.first] = 0;
    }

    // Evaluate communication effectiveness
    for (auto& msg : messages) {
        string sender, receiver, language;
        tie(sender, receiver, language) = msg;

        // Check if sender and receiver have different primary languages
        if (teamMembers[sender] != teamMembers[receiver]) {
            // Increase score for effective cross-language communication
            communicationScore[sender]++;
        }
    }
```

```
    // Find the member with highest communication score
    string bestCommunicator = "";
    int maxScore = 0;
    for (auto& score : communicationScore) {
        if (score.second > maxScore) {
            maxScore = score.second;
            bestCommunicator = score.first;
        }
    }

    return bestCommunicator;
}

int main() {
    int N, M;
    cin >> N;
    vector<pair<string, string>> teamMembers(N);
    for (int i = 0; i < N; i++) {
        cin >> teamMembers[i].first >> teamMembers[i].second;
    }

    cin >> M;
    vector<tuple<string, string, string>> messages(M);
    for (int i = 0; i < M; i++) {
        cin >> get<0>(messages[i]) >> get<1>(messages[i]) >>
get<2>(messages[i]);
    }

    cout << findBestCommunicator(N, teamMembers, M, messages) << endl;

    return 0;
}
```