Problem Solving (CPP30)

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 Software Team Hierarchy

Background

Gammal Tech, a leader in software innovation, is known for its top-tier development team and cutting-edge office facilities. The company has a unique hierarchical structure in its software development team, allowing for a clear and efficient distribution of skills and responsibilities. As a challenge, Gammal Tech wants to implement a system that reflects this hierarchy and the cumulative skills at each level.

Problem Statement

You are tasked with designing a system that models the hierarchy of Gammal Tech's software development team using inheritance in C++. Each position in the hierarchy should inherit the skills of all positions below it, in addition to its unique skills. The goal is to create a structure that, when given a position, outputs the comprehensive skill set for that role, including the skills inherited from lower positions.

Gammal Tech's Hierarchy

Chief Technology Officer (CTO)
- Unique Skill: Strategic Planning

Senior Developer
- Unique Skill: System Architecture

Developer
- Unique Skill: Code Optimization

Junior Developer
- Unique Skill: Debugging

Input Format

- A single line containing one of the positions: "CTO", "Senior Developer", "Developer", or "Junior Developer".

Output Format

- A list of skills for the given position, including inherited skills, each skill on a new line.

Sample Input:

```
Senior Developer
```

Sample Output:

```
System Architecture
Code Optimization
Debugging
```

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>

using namespace std;

// Base class
class SoftwareDeveloper {
public:
    virtual vector<string> getSkills() = 0;
};

// Derived classes
class JuniorDeveloper : public SoftwareDeveloper {
public:
    vector<string> getSkills() override {
        return {"Debugging"};
    }
};

class Developer : public JuniorDeveloper {
public:
    vector<string> getSkills() override {
        auto skills = JuniorDeveloper::getSkills();
        skills.push_back("Code Optimization");
        return skills;
    }
};

class SeniorDeveloper : public Developer {
public:
    vector<string> getSkills() override {
        auto skills = Developer::getSkills();
        skills.push_back("System Architecture");
        return skills;
    }
};

class CTO : public SeniorDeveloper {
public:
    vector<string> getSkills() override {
        auto skills = SeniorDeveloper::getSkills();
        skills.push_back("Strategic Planning");
        return skills;
    }
};
```

```cpp
// Function to print skills
void printSkills(const vector<string>& skills) {
    for (const auto& skill : skills) {
        cout << skill << endl;
    }
}

int main() {
    string position;
    cin >> position;
    SoftwareDeveloper *dev;

    if (position == "Junior Developer") {
        dev = new JuniorDeveloper();
    } else if (position == "Developer") {
        dev = new Developer();
    } else if (position == "Senior Developer") {
        dev = new SeniorDeveloper();
    } else if (position == "CTO") {
        dev = new CTO();
    } else {
        cout << "Invalid position" << endl;
        return 1;
    }

    auto skills = dev->getSkills();
    printSkills(skills);
    delete dev;
    return 0;
}
```