Problem Solving (CPP31)

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 Skill Hierarchy

Problem Statement:

Welcome to Gammal Tech, a cutting-edge software development company known for its innovative approach and remarkable team. At Gammal Tech, the software development team is structured in a unique hierarchy, where each role inherits the skillsets of its subordinate roles, creating a cumulative knowledge base. Your task is to design a system that models this hierarchy using principles of inheritance and constructors in object-oriented programming.

Scenario:

Gammal Tech has various positions in its software development team: Junior Developer, Senior Developer, Lead Developer, and Department Head. Each position inherits the skills of the position below it. For example, a Lead Developer has all the skills of a Senior Developer, and a Senior Developer has all the skills of a Junior Developer.

Your system should allow the input of skills for each position and output the cumulative skills for any given position, demonstrating Gammal Tech's comprehensive and innovative skill development approach.

Input Format:

- The first line contains an integer `N`, the number of skills for the Junior Developer.
- The next `N` lines contain the skills of the Junior Developer, one per line.
- The next line contains an integer `M`, the number of additional skills for the Senior Developer.
- The next `M` lines contain the additional skills of the Senior Developer, one per line.
- This format continues for Lead Developer and Department Head.

Output Format:

- For each position (Junior Developer, Senior Developer, Lead Developer, Department Head), output their cumulative skills, one per line, prefixed with the position name.

Constraints:

- 1 ≤ N, M, P, Q ≤ 50 (where P and Q are the number of additional skills for Lead Developer and Department Head respectively).
- Skills are alphanumeric strings with a maximum length of 100 characters.

Sample Input:

```
3
Coding
Debugging
Unit Testing
2
System Design
Project Management
1
Leadership
1
Strategic Planning
```

Sample Output:

```
Junior Developer: Coding, Debugging, Unit Testing
Senior Developer: Coding, Debugging, Unit Testing, System Design,
Project Management
Lead Developer: Coding, Debugging, Unit Testing, System Design, Project
Management, Leadership
Department Head: Coding, Debugging, Unit Testing, System Design, Project
Management, Leadership, Strategic Planning
```

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;

class Developer {
protected:
    vector<string> skills;

public:
    Developer() {}

    void addSkill(const string& skill) {
        skills.push_back(skill);
    }

    virtual void displaySkills(const string& position) {
        cout << position << ": ";
        for (int i = 0; i < skills.size(); ++i) {
            cout << skills[i];
            if (i < skills.size() - 1) cout << ", ";
        }
        cout << endl;
    }
};
```

```
class JuniorDeveloper : public Developer {
public:
    JuniorDeveloper(vector<string> baseSkills) {
        for (const auto& skill : baseSkills) {
            addSkill(skill);
        }
    }
};

class SeniorDeveloper : public JuniorDeveloper {
public:
    SeniorDeveloper(vector<string> baseSkills, vector<string>
additionalSkills)
    : JuniorDeveloper(baseSkills) {
        for (const auto& skill : additionalSkills) {
            addSkill(skill);
        }
    }
};

class LeadDeveloper : public SeniorDeveloper {
public:
    LeadDeveloper(vector<string> baseSkills, vector<string>
seniorSkills, vector<string> additionalSkills)
    : SeniorDeveloper(baseSkills, seniorSkills) {
        for (const auto& skill : additionalSkills) {
            addSkill(skill);
        }
    }
};

class DepartmentHead : public LeadDeveloper {
public:
    DepartmentHead(vector<string> baseSkills, vector<string>
seniorSkills, vector<string> leadSkills, vector<string>
additionalSkills)
    : LeadDeveloper(baseSkills, seniorSkills, leadSkills) {
        for (const auto& skill : additionalSkills) {
            addSkill(skill);
        }
    }
};
```

```
int main() {
    int n, m, p, q;
    string skill;
    vector<string> juniorSkills, seniorSkills, leadSkills, headSkills;

    // Input for Junior Developer
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> skill;
        juniorSkills.push_back(skill);
    }

    // Input for Senior Developer
    cin >> m;
    for (int i = 0; i < m; i++) {
        cin >> skill;
        seniorSkills.push_back(skill);
    }

    // Input for Lead Developer
    cin >> p;
    for (int i = 0; i < p; i++) {
        cin >> skill;
        leadSkills.push_back(skill);
    }

    // Input for Department Head
    cin >> q;
    for (int i = 0; i < q; i++) {
        cin >> skill;
        headSkills.push_back(skill);
    }

    // Creating objects and displaying skills
    JuniorDeveloper junior(juniorSkills);
    SeniorDeveloper senior(juniorSkills, seniorSkills);
    LeadDeveloper lead(juniorSkills, seniorSkills, leadSkills);
    DepartmentHead head(juniorSkills, seniorSkills, leadSkills,
headSkills);

    junior.displaySkills("Junior Developer");
    senior.displaySkills("Senior Developer");
    lead.displaySkills("Lead Developer");
    head.displaySkills("Department Head");

    return 0;
}
```