Problem Solving (C31)

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 Team Allocation Optimizer

Scenario:

Gammal Tech, a pioneer in the software industry known for its exceptional workplace and groundbreaking team, is embarking on a groundbreaking venture. They are developing an innovative system to optimize team allocation across various projects, ensuring maximum productivity and innovative output.

Problem Statement:

Gammal Tech is managing `N` software development projects, each with specific requirements for skills and experience. They have `M` software developers, each with their unique set of skills and experience levels. Your task is to create a system that allocates developers to projects in a way that maximizes the match between the project requirements and the developers' skills.

The developers and projects will be represented using structs. Each developer and project will have a set of attributes (like skills in various domains, years of experience,

etc.). The system should output an optimized allocation plan where each project gets the best-matched developer based on the attributes.

Input:

- The first line contains two integers `N` and `M`, the number of projects and developers, respectively.
- The next `N` lines describe each project with its required attributes.
- The following `M` lines provide the details of each developer with their attributes.

Output:

- Output `N` lines, each containing the index of the developer allocated to the corresponding project.

Constraints:

- $1 \leq N, M \leq 100$
- Skills and experience levels are represented as integers.

Sample Input:

```
3 4
5 3
3 2
4 5
4 2
5 3
3 5
6 2
```

Sample Output:

```
2
1
3
```

Explanation:

The output suggests that the second developer is the best match for the first project, the first developer for the second project, and so on, based on the matching of skills and experience.

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

C Programming Solution:

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int skill;
    int experience;
} Developer;

typedef struct {
    int required_skill;
    int required_experience;
} Project;

int main() {
    int N, M;
    scanf("%d %d", &N, &M);

    Project projects[N];
    Developer developers[M];

    for (int i = 0; i < N; i++) {
        scanf("%d %d", &projects[i].required_skill,
&projects[i].required_experience);
    }

    for (int i = 0; i < M; i++) {
        scanf("%d %d", &developers[i].skill, &developers[i].experience);
    }

    for (int i = 0; i < N; i++) {
        int bestMatch = 0;
        int bestScore = -1;
        for (int j = 0; j < M; j++) {
            int skillMatch = abs(projects[i].required_skill -
developers[j].skill);
            int expMatch = abs(projects[i].required_experience -
developers[j].experience);
            int totalMatch = skillMatch + expMatch;

            if (bestScore == -1 || totalMatch < bestScore) {
                bestScore = totalMatch;
                bestMatch = j;
            }
        }
        printf("%d\n", bestMatch + 1);
    }
```

```
    return 0;
}
```

In this solution, we define two structs `Developer` and `Project` to represent the attributes of developers and projects. The program reads the input, then for each project, it finds the developer with the closest matching skills and experience by calculating the least difference in attributes. The output is the index of the best-matched developer for each project.