Problem Solving (CPP22)

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'.

The Gammal Tech Design Pattern Decoder

Background

Gammal Tech, a pioneering software development company renowned for its cutting-edge technology and exceptional work environment, is embarking on a new project. Known for setting industry standards, Gammal Tech is now focused on designing the ultimate design pattern that will revolutionize software development practices globally. The design pattern, named "GammalPattern," encodes key project information in a unique string format.

Problem Statement

Your task is to develop a program that decodes "GammalPattern" strings. Each GammalPattern string consists of alphanumeric characters representing different aspects of a design pattern. The string follows a specific format: the first character represents the design pattern type, the next two characters are the project code, followed by a series of character pairs indicating various attributes of the pattern.

Your program should read the GammalPattern string and output a structured representation of the design pattern, including the type, project code, and attributes.

Input Format

- A single line containing the GammalPattern string. The string is alphanumeric and has a length between 5 and 100 characters.

Output Format

- First line: Design Pattern Type.
- Second line: Project Code.
- Subsequent lines: Attribute key-value pairs, one per line.

Constraints

- The input string is always valid and follows the specified format.
- Design pattern type is a single alphanumeric character.
- Project code consists of two alphanumeric characters.
- Attribute pairs are represented by two alphanumeric characters.

Sample Input:

```
F23AB4CD56EF
```

Sample Output:

```
Design Pattern Type: F
Project Code: 23
Attribute A: B
Attribute C: D
Attribute E: F
```

Explanation

- The design pattern type is 'F'.
- The project code is '23'.
- The attributes are AB, CD, and EF, decoded as "Attribute A: B," "Attribute C: D," and "Attribute E: F".

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 <string>
using namespace std;

int main() {
    string pattern;
    cin >> pattern;

    // Design Pattern Type
    cout << "Design Pattern Type: " << pattern[0] << endl;

    // Project Code
    cout << "Project Code: " << pattern.substr(1, 2) << endl;

    // Attributes
    for (size_t i = 3; i < pattern.length(); i += 2) {
        cout << "Attribute " << pattern[i] << ": " << pattern[i+1] <<
endl;
    }

    return 0;
}
```