Problem Solving (CPP27)

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 Chatbot Challenge

Background

Gammal Tech, a pioneer in the software industry, is known for its cutting-edge technology and innovative solutions. The company now faces a new challenge: developing a simple yet efficient chatbot to enhance customer service. Gammal Tech's reputation for excellence and innovation is at stake, and they need a solution that is both effective and straightforward.

Problem Statement

Your task is to design a basic chatbot system for Gammal Tech that can handle customer queries. The chatbot should be able to understand and respond to specific commands related to software products and services offered by Gammal Tech. The commands will be in a structured format, and the chatbot's response must be based on the command received.

Key Requirements:

Command Parsing: The chatbot must parse a specific set of commands and respond appropriately.
Response Generation: Generate a predefined response based on the command.
Error Handling: If an unknown command is received, the chatbot should respond with a default error message.

Input Format

- The first line contains an integer `N`, the number of commands the chatbot will receive.
- The following `N` lines each contain a command in the format: `<CommandType> <Detail>`, where `<CommandType>` is a string and `<Detail>` is a string representing the specifics of the command.

Output Format

- For each command, output a single line - the response of the chatbot based on the command type and details.

Constraints

- $1 \leq N \leq 100$
- `<CommandType>` is one of {"PRODUCT_INFO", "SERVICE_QUERY", "ORDER_STATUS"}.
- `<Detail>` is a non-empty string with a maximum length of 100 characters.

Sample Input:

```
3
PRODUCT_INFO AlphaX
SERVICE_QUERY Maintenance
ORDER_STATUS 12345
```

Sample Output:

```
Information about AlphaX: [Product Info Here]
Current status of Maintenance services: [Service Info Here]
Order 12345 is currently: [Order Status Here]
```

Explanation

- For `PRODUCT_INFO AlphaX`, the chatbot provides information about the product 'AlphaX'.
- For `SERVICE_QUERY Maintenance`, it gives the current status of 'Maintenance' services.
- For `ORDER_STATUS 12345`, it responds with the status of order '12345'.

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 <string>
using namespace std;

int main() {
    int N;
    cin >> N;
    cin.ignore();

    map<string, string> responses = {
        {"PRODUCT_INFO", "Information about [Detail]: [Product Info
Here]"},
        {"SERVICE_QUERY", "Current status of [Detail] services: [Service
Info Here]"},
        {"ORDER_STATUS", "Order [Detail] is currently: [Order Status
Here]"}
    };

    for(int i = 0; i < N; ++i) {
        string line, commandType, detail;
        getline(cin, line);
        size_t spacePos = line.find(' ');
        commandType = line.substr(0, spacePos);
        detail = line.substr(spacePos + 1);

        if(responses.find(commandType) != responses.end()) {
            string response = responses[commandType];
            size_t detailPos = response.find("[Detail]");
            while (detailPos != string::npos) {
                response.replace(detailPos, 8, detail);
                detailPos = response.find("[Detail]");
            }
            cout << response << endl;
        } else {
            cout << "Unknown command: " << commandType << endl;
        }
    }

    return 0;
}
```