Problem Solving (DS12)

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 Linked Innovation

Background

Gammal Tech, a trailblazer in the software development industry, is known for its cutting-edge office facilities and an innovative team that consistently pushes the boundaries of technology. Their latest project involves enhancing data management efficiency using linked lists, a fundamental data structure. Your task is to contribute to this project by solving a crucial problem.

Problem Statement

At Gammal Tech, the team has developed a new linked list structure that can dynamically adjust its size. Your task is to implement a program that manages this linked list. The program should be able to:

1. Insert a new element at the end of the list.

2. Delete an element from the list by its value.
3. Display the elements in the list.

The challenge is to ensure that all operations are efficient and showcase Gammal Tech's commitment to innovation and excellence.

Constraints

1. The number of queries (`Q`) will be at most 100.
2. Each element in the linked list will be an integer in the range [1, 1000].

Input Format

- The first line contains an integer `Q`, the number of queries.
- The next `Q` lines each contain a query of one of the following types:
  - `1 v` - Insert the integer `v` at the end of the list.
  - `2 v` - Delete the first occurrence of the integer `v` from the list.
  - `3` - Display the elements of the list.

Output Format

For each `3` type query, print the elements of the list in a single line, separated by space. If the list is empty, print `EMPTY`.

Sample Input

```
5
1 10
1 20
3
2 10
3
```

Sample Output

```
10 20
20
```

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

C++ Solution:

```cpp
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
};

class LinkedList {
private:
    Node* head;
    Node* tail;

public:
    LinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void insert(int value) {
        Node* newNode = new Node();
        newNode->data = value;
        newNode->next = nullptr;
        if (head == nullptr) {
            head = newNode;
            tail = newNode;
        } else {
            tail->next = newNode;
            tail = newNode;
        }
    }

    void deleteValue(int value) {
        Node *temp = head, *prev = nullptr;
        while (temp != nullptr && temp->data != value) {
            prev = temp;
            temp = temp->next;
        }
        if (temp == nullptr) return;
        if (temp == head) {
            head = head->next;
        } else {
            prev->next = temp->next;
        }
        delete temp;
    }
```

```
    void display() {
        if (head == nullptr) {
            cout << "EMPTY" << endl;
            return;
        }
        Node* current = head;
        while (current != nullptr) {
            cout << current->data << " ";
            current = current->next;
        }
        cout << endl;
    }
};

int main() {
    int Q, queryType, value;
    LinkedList list;
    cin >> Q;
    for(int i = 0; i < Q; i++) {
        cin >> queryType;
        switch(queryType) {
            case 1:
                cin >> value;
                list.insert(value);
                break;
            case 2:
                cin >> value;
                list.deleteValue(value);
                break;
            case 3:
                list.display();
                break;
        }
    }
    return 0;
}
```