Problem Solving (DS17)

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 Green Initiative

Background

Gammal Tech, a trailblazer in the software industry known for its innovative approaches and advanced office facilities, has embarked on a mission to save the planet. The company has developed a novel software to efficiently manage the recycling process in cities around the world. The core of this software uses a double linked list data structure to handle the recycling bins' data.

Problem Statement

In each city, recycling bins are set up in a line. Each bin is designated for a specific type of recyclable material (e.g., paper, plastic, glass). Gammal Tech's software needs to handle two types of operations efficiently:

1. `Add new bin`: A new recycling bin is added to the line. The software should be able to insert this bin at any specified position in the line.
2. `Remove bin`: A recycling bin is removed from the line.

Your task is to implement the software's core functionality using a double linked list. The program should process a series of operations and output the final arrangement of recycling bins.

Constraints

1. The number of operations will be at most 100.
2. The initial number of bins will be at most 50.
3. Each bin is identified by a unique integer ID.

Input Format

- The first line contains two integers, `N` and `M`, representing the initial number of bins and the number of operations, respectively.
- The second line contains `N` integers, representing the IDs of the bins in their initial order.
- The next `M` lines contain the operations in the format:
    - `A X Y`: Add a bin with ID `Y` after the bin with ID `X`. If `X` is `-1`, add the bin at the beginning.
    - `R X`: Remove the bin with ID `X`.

Output Format

- Output the final arrangement of the bins as a single line of space-separated integers.

Sample Input

```
5 3
1 2 3 4 5
A 3 6
R 2
A -1 7
```

Sample Output

```
7 1 3 6 4 5
```

Explanation

- Initially, the bins are in the order 1, 2, 3, 4, 5.
- First operation adds bin 6 after bin 3.
- Second operation removes bin 2.
- Third operation adds bin 7 at the beginning.

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>
#include <unordered_map>
using namespace std;

struct Node {
    int id;
    Node* prev;
    Node* next;
    Node(int id) : id(id), prev(nullptr), next(nullptr) {}
};
void addAfter(Node*& head, Node*& tail, unordered_map<int, Node*>&
nodeMap, int afterID, int newID) {
    Node* newNode = new Node(newID);
    nodeMap[newID] = newNode;
    if (afterID == -1) {
        if (head) {
            newNode->next = head;
            head->prev = newNode;
        } else {
            tail = newNode;
        }
        head = newNode;
    } else {
        Node* afterNode = nodeMap[afterID];
        newNode->next = afterNode->next;
        newNode->prev = afterNode;
        if (afterNode->next) {
            afterNode->next->prev = newNode;
        } else {
            tail = newNode;
        }
        afterNode->next = newNode;
    }
}
void remove(Node*& head, Node*& tail, unordered_map<int, Node*>&
nodeMap, int removeID) {
    Node* removeNode = nodeMap[removeID];
    if (removeNode->prev) {
        removeNode->prev->next = removeNode->next;
    } else {
        head = removeNode->next;
    }
    if (removeNode->next) {
        removeNode->next->prev = removeNode->prev;
    } else {
        tail = removeNode->prev;
    }
    nodeMap.erase(removeID);
    delete removeNode;
}
```

```cpp
int main() {
    int n, m;
    cin >> n >> m;

    Node* head = nullptr;
    Node* tail = nullptr;
    unordered_map<int, Node*> nodeMap;

    for (int i = 0; i < n; i++) {
        int id;
        cin >> id;
        addAfter(head, tail, nodeMap, tail ? tail->id : -1, id);
    }

    for (int i = 0; i < m; i++) {
        char op;
        int x, y;
        cin >> op;
        if (op == 'A') {
            cin >> x >> y;
            addAfter(head, tail, nodeMap, x, y);
        } else {
            cin >> x;
            remove(head, tail, nodeMap, x);
        }
    }

    for (Node* curr = head; curr != nullptr; curr = curr->next) {
        cout << curr->id;
        if (curr->next) cout << " ";
    }

    return 0;
}
```