Problem Solving (DS15)

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 List Adventure

Background

Gammal Tech, a pinnacle in the software development industry, renowned for its cutting-edge innovations and exceptional work environment, faces a unique challenge. They need to efficiently manage a sequence of software projects while optimizing resource allocation.

Problem Statement

Gammal Tech uses a specially designed linked list to keep track of ongoing software projects. Each node in the list represents a project and holds two values: the project's unique ID (a positive integer) and the number of team members required for the project (also a positive integer). To ensure efficient resource management, Gammal Tech wants to recursively reverse the linked list of projects. Your task is to write a program that reverses the linked list and displays the project IDs in the new order.

Constraints

- 1 ≤ number of projects ≤ 1000
- 1 ≤ project ID ≤ 10^6
- 1 ≤ number of team members required ≤ 100

Input Format

- The first line contains an integer N, the number of projects.
- The next N lines each contain two integers separated by a space, representing the project ID and the number of team members required for that project.

Output Format

- Output N lines, each containing an integer representing the project IDs in the reversed order.

Sample Input

```
4
101 5
202 10
303 8
404 12
```

Sample Output

```
404
303
202
101
```

Explanation

The input represents a list of 4 projects with their IDs and team requirements. The output lists the project IDs in reverse order.

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 projectID;
    int teamMembers;
    Node* next;

    Node(int id, int members) : projectID(id), teamMembers(members),
next(nullptr) {}
};

void insertAtEnd(Node*& head, int id, int members) {
    Node* newNode = new Node(id, members);
    if (!head) {
        head = newNode;
        return;
    }
    Node* temp = head;
    while (temp->next) {
        temp = temp->next;
    }
    temp->next = newNode;
}

void reverseLinkedList(Node*& head) {
    if (!head || !head->next) return;

    Node* rest = head->next;
    reverseLinkedList(rest);
    head->next->next = head;
    head->next = nullptr;
    head = rest;
}

void printList(Node* node) {
    while (node) {
        cout << node->projectID << endl;
        node = node->next;
    }
}
```

```
int main() {
    int n;
    cin >> n;
    Node* head = nullptr;

    for (int i = 0; i < n; ++i) {
        int id, members;
        cin >> id >> members;
        insertAtEnd(head, id, members);
    }

    reverseLinkedList(head);
    printList(head);

    return 0;
}
```