Problem Solving (DS13)

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 Innovation Queue

Background

Gammal Tech, a pioneering software development company, renowned for its advanced office facilities and an innovative team, is at the forefront of technological evolution. They have recently embarked on a project to revolutionize data management systems, with a focus on the efficient and dynamic use of linked lists. As part of their R&D team, your challenge is to contribute to this groundbreaking work.

Problem Statement

Gammal Tech is developing a new technology for managing their internal task queue, where tasks are dynamically added and removed. The system uses a special kind of linked list to store tasks. Each node in the list contains the task ID and its priority. The list is sorted by priority in descending order (higher priority tasks appear first). However,

the twist is that every time a task is added or completed, the priorities of all tasks are recalculated based on a proprietary algorithm.

Your task is to implement this system. Specifically, you must be able to add new tasks, complete the highest priority task, and display the task list.

Constraints

- 1 ≤ Number of tasks ≤ 10^5
- 1 ≤ Task ID ≤ 10^6
- 1 ≤ Priority ≤ 10^9
- Task IDs are unique.

Input Format

- The first line contains an integer `N`, the number of operations to perform.
- The next `N` lines describe the operations and are of the following types:
  - `ADD id priority`: Add a task with given `id` and `priority`.
  - `COMPLETE`: Complete the task with the highest priority.
  - `SHOW`: Show all tasks in the queue in the format `id-priority`.

Output Format

- For each `SHOW` operation, output a single line containing all the tasks in the queue in the given format, separated by a space. If the queue is empty, output `EMPTY`.

Sample Input

```
5
ADD 101 500
ADD 102 300
SHOW
COMPLETE
SHOW
```

Sample Output

```
101-500 102-300
102-300
```

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 <set>
#include <map>
using namespace std;

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

    // Using a set to store tasks with a custom comparator for sorting
by priority
    set<pair<int, int>, greater<pair<int, int>>> tasks;
    map<int, int> idToPriority;

    while (N--) {
        string command;
        cin >> command;

        if (command == "ADD") {
            int id, priority;
            cin >> id >> priority;
            tasks.insert({priority, id});
            idToPriority[id] = priority;
        } else if (command == "COMPLETE") {
            if (!tasks.empty()) {
                auto it = tasks.begin();
                idToPriority.erase(it->second);
                tasks.erase(it);
            }
        } else if (command == "SHOW") {
            if (tasks.empty()) {
                cout << "EMPTY";
            } else {
                for (auto &task : tasks) {
                    cout << task.second << "-" << task.first;
                    if (task != *tasks.rbegin()) {
                        cout << " ";
                    }
                }
            }
            cout << endl;
        }
    }

    return 0;
}
```