Toj Theme Contest 001 資料結構入門

Toj Theme Contest 001 資料結構入門 試題本

競賽規則

1. 競賽時間：2025/03/30 14:00 ~ 16:00，共 2 小時。
2. 本次競賽試題共 5 題，每題皆有子任務。
3. 為了愛護地球，本次競賽題本僅提供電子檔，不提供紙本。
4. 每題的分數為該題所有子任務得分數加總；單筆子任務得分數為各筆繳交在該筆得到的最大分數。
5. 全部題目的輸入皆為標準輸入。
6. 全部題目的輸出皆為標準輸出。
7. 所有輸入輸出請嚴格遵守題目要求，多或少的換行及空格皆有可能造成裁判系統判斷為答案錯誤。
8. 每題每次上傳間隔為 30 秒，裁判得視情況調整。
9. 所有試題相關問題請於競賽系統中提問，題目相關公告也會公告於競賽系統，請密切注意。
10. 競賽中請勿交談。請勿做出任何會干擾競賽的行為。
11. 如需使用 C++ 的 `std::cin` 或 `std::cout` 可將以下程式碼插入 main function 以及將 `endl` 取代為 '`\n`' 來優化輸入輸出速度。唯須注意不可與 `cstdio` 混用。

```
std::ios::sync_with_stdio(false);
std::cin.tie(nullptr);
```

A. 有很多牌就能榮和

Problem ID: 2025TTC_Many
Time Limit: 1.0s
Memory Limit: 512MiB

– 本題輸入方式為非典型，請注意!!! –

Figure 1: 四暗刻

Blame，一位年輕的麻將新手，剛踏入這個充滿計算與推理的世界。他眼中的麻將桌，每一張牌不僅僅是一個數字，而是智慧、策略與運氣緊密相連的符號。這段旅程的起點，是因為 Blame 被台南一中資訊社的夥伴們吸引，看著他們每天中午聚在一起快樂對局，雖然偶爾有人眉頭微蹙，但整體氛圍依舊歡愉。而那一瞬間，當牌局節奏突然加快、有人胡出 ‘‘四暗刻’’，螢幕上絢麗的特效與驚人的點數展示，令 Blame 瞬間神魂顛倒。

這一幕深深地觸動了 Blame 的心，他決定也要加入這麻將的世界，並以最快的速度晉升為高手。雖然起初他對規則不甚了解，常因操作失誤而「放槍」，但很快他發現，一局麻將的勝負，不僅僅靠運氣，更需要在極短的反應時間內進行快速判斷與計算。於是，Blame 不再僅憑運氣，而是將目光放在一項關鍵技術上：只要能夠迅速掌握牌堆中出現次數最多的牌，就能為自己的策略提供強大支援，進而掌握勝利的先機。

然而，現實並不如想像中那般簡單。每局對戰中，牌堆內的牌數千變萬化，儘管每張牌代表的數字皆不相同，但數字卻可能極為龐大，甚至有時候出現的張數高達難以置信的數量。面對如此快速且變化莫測的局勢，Blame 的腦海忙亂運轉，卻難以在短暫的時間內逐一計算各牌的出現次數。於是，他決定求助於智慧的力量，設計出一套能夠迅速運算、動態更新的系統。

這個系統能夠處理多達 (Q) 次操作，每一次操作都會讓牌堆狀況發生改變：

1. 當一張牌（其牌面數字為 (A)）被放入牌堆時，程式會自動記錄並更新這張牌的數量。

2. 當一張牌（牌面為 (A)）從牌堆中被取出時，系統同樣會調整該牌的數量。

3. 當 Blame 查詢牌堆中出現最多次數的牌時，程式會立即回傳該牌的數字以及它在牌堆中的數量。如果多種牌的數量相同，則會回傳數字最小的一組。

這個系統不僅讓 Blame 能夠隨時掌握牌堆的動態，更在極短的反應時間內，提供他所需的關鍵資訊。牌堆中哪一種牌出現得最多，系統會在瞬間告知，並告訴他具體數量，從而協助他迅速作出策略調整。

這套簡單而高效的系統，迅速成為 Blame 在《雀魂麻將》中的制勝利器。曾經那個迷失方向的新手，逐漸變得從容、冷靜且充滿智慧。在每一局快節奏的對戰中，他不再只是依賴運氣，而是用精準的數據判斷，將每一步走得更為堅實。隨著時間推進，Blame 也漸漸領悟到麻將的真髓—---它不僅是一場運氣與技巧的較量，更是一場關於邏輯、數據與快速反應的智慧之戰。

從此，Blame 的麻將人生因這個程式而煥然一新，他不再是那個在牌桌上徬徨的小白，而是以科技與智慧佔據了優勢。每一次操作、每一個決策，都在證明著他從失誤中學習、從挑戰中成長。這段充滿挑戰與機遇的旅程，不僅見證了他對麻將藝術的追求，更象徵著一位年輕人對於智慧與技術的無限熱情與堅持。

– 作答方式 –

本題採用函數呼叫方式。

請實作以下三個函式:

```
void add(int x) {
    // 給你 x 要你把 x 加入進系統當中
}

void remove(int x) {
    // 給你 x 要你把 x 從系統中刪除
    // 假如 x 不存在系統中，請直接回傳
}

pair<int, int> query() {
    // 詢問當前系統中的眾數值以及數量
    // 假如有相同的請輸出值最小的那個
    // 假如系統中沒東西，回傳 {-87, -87}
}
```

– 輸入限制 –

- $1 \leq x \leq 10^9$
- $1 \leq$ 函式呼叫次數 $\leq 10^5$

– 子任務 –

編號	分數	額外限制
0	100	無額外限制