程式設計（113-1）

作業十

作業設計：孔令傑
國立臺灣大學資訊管理學系

繳交作業時，請至 PDOGS（`http://pdogs.ntu.im/`）爲第一、二、三題上傳一份 C++ 原始碼（以複製貼上原始碼的方式上傳），並且到 NTU COOL 上傳一份 PDF 檔。每位學生都要上傳自己寫的解答。不接受紙本繳交；不接受遲交。這份作業的程式碼和書面報告的截止時間都是 ***11 月 26 日早上八點***。爲這份作業設計測試資料並且提供解答的助教是許群佑。

在你開始前，請閱讀課本的第 12–13 章[1]。

本次作業滿分爲 120 分，得幾分就算幾分。若整學期有 n 份作業，則學期的作業總成績即爲 n 份作業的總分除以 n（不論超過 100 與否）。

第一題

（30 分）本題延續自作業九的第一題，大部份規則與該題一樣，但有一些細節與之前的不盡相同，也有新增新的要求。以下將從頭完整地敘述本題，建議大家在此都還是要詳細閱讀一次題目。

有一家診所，該診所有兩位醫師，編號分別爲 1 和 2，且兩位醫師的看診速度相同。編號 2 的醫師要兼顧管理工作，所以在診所剛開門的時候不看診，而是在一位新抵達的患者使在醫師 1 的診間排隊的患者人數達到 L 時才加入看診。

在醫師 2 投入看診前，所有患者都會去醫師 1 的診間接受看診或排隊。每位患者都有各自的年齡、抵達時間和看診時間長。我們稱呼第 i 個抵達的患者爲患者 i，而患者 i 的年齡爲 g_i、抵達時間爲 a_i、看診時間長爲 s_i。這間診所不想讓年長者等太久，所以當一位患者抵達時，診所會引導他詢問排在他前面一位的患者的年齡，如果他比他前面一位年長至少 b 歲（包含 b 歲）時，診所會讓這位年長患者跟前面這位相對年輕的患者交換序位，然後再繼續往前看是否可以繼續交換，直到不能交換爲止[2]。無論如何，都只能跟等待中的患者交換位置；已經在接受看診的患者無論如何都會被看完，不會被中斷。

在醫師 2 投入看診之後，如果有一位新患者出現，診所會檢視兩個診間的狀況，然後將他安排到總人數（包含被看診中和排隊中）較少的診間；若兩個診間的總人數相同，則將該患者安排至醫師 1 的診間中。如果某位患者抵達時，恰好有一位或兩位醫師完成看診（亦即有患者離開），我們說這些抵達與離開都發生在同一瞬間，但新抵達的患者會在要離開的患者離開前被安排到正確的診間，然後被看診完成的患者才離開。這位患者接著就按照醫師 2 投入看診前的規則在他所在的診間排隊、交換排隊順序、接受看診。

在醫師 1 的診間排隊人數達到 L 的那個瞬間，有許多事件可能同時發生，其中一定會同時發生的包含新患者抵達與醫師 2 加入，另外也可能同時醫師 1 完成一位患者的看診。如果在那個瞬間有以上兩或三個事件發生，我們說這數個事件都發生在醫師 1 的診間排隊人數達到 L 的那個瞬間，但還是有先後順序。如果在那個瞬間醫師 1 完成一位患者的看診，則三個事件依序是新患者抵達並且進入醫師 1 的診間排隊或就診並且完成必要的排隊順序交換，接著是醫師 1 完成一位患者的看診，最後是醫師 2 加

[1]課本是 Deitel and Deitel 著的 *C++ How to Program: Late Objects Version* 第七版。

[2]這個流程很像 insertion sort 中找到正確位置插入的流程。

入；如果在那個瞬間醫師 1 並沒有完成一位患者的看診，則兩個事件依序是新患者抵達並且進入醫師 1 的診間排隊或就診並且完成必要的排隊順序交換，接著是醫師 2 加入。

給定一系列依序抵達的患者，我們可以計算當天醫師 1 和醫師 2 的看診人數，分別記爲 D_1 和 D_2。舉例來說，假設患者們的資訊如表 1 所示，$b = 15$、$L = 3$：

項目	抵達順序 i					
	1	2	3	4	5	6
年齡 g_i	38	40	34	52	65	80
到達時間 a_i	6	7	8	9	10	25
看診時間長 s_i	5	8	4	6	2	3

表 1: 患者資訊

- 患者 1 到達時由醫師 1 看診，看診完成時間爲第 $f_1 = 6 + 5 = 11$ 分鐘。
- 患者 2、3、4 依序在第 7、8、9 分鐘到達。他們到達時，醫師 1 在爲患者 1 看診，因此他們三位都需要排隊。因爲 $g_4 \geq g_3 + 15$，患者 4 比比他早到的患者 3 年長 15 歲以上，因此患者 4 會跟患者 3 交換看診序號。在患者 4 加入的那瞬間且完成交換後，患者 2、3、4 在醫師 1 的診間依序排在第一、三、二位。
- 同樣在第 9 分鐘的瞬間，患者 4 的抵達會讓醫師 2 開始看診。患者 2 會開始接受醫師 2 看診，其看診完成時間爲第 $f_2 = 9 + 8 = 17$ 分鐘，等待時間爲 $w_2 = 9 - 7 = 2$；患者 4 會在醫師 1 的診間排第一位、患者 3 會在醫師 2 的診間排第一位。
- 患者 5 在第 10 分鐘到達，此時兩個診間的總人數都是 2，因此診所將安排患者 5 進入醫師 1 的診間。患者 5 並沒有比排在他前面的患者 4 年長 15 歲以上，因此患者 5 排在第二位。
- 患者 1 在第 11 分鐘離開之後，醫師 1 會繼續爲患者 4 看診，看診完成時間爲第 $f_4 = 11 + 6 = 17$ 分鐘，等待時間爲 $w_4 = 11 - 9 = 2$。
- 在第 17 分鐘，患者 2 會離開醫師 2 的診間，患者 4 則離開醫師 1 的診間。醫師 1 會繼續爲患者 5 看診，等待時間 $w_5 = 17 - 10 = 7$，看診完成時間則爲第 $f_5 = 17 + 2 = 19$ 分鐘；醫師 2 會繼續爲患者 3 看診，等待時間 $w_3 = 17 - 8 = 9$，看診完成時間則爲第 $f_3 = 17 + 4 = 21$ 分鐘。
- 患者 6 於第 25 分鐘到達時，兩位醫師都在閒置，因此診所會安排由醫師 1 爲他看診，看診完成時間則是 $f_6 = 25 + 3 = 28$。

上述推算資訊可以被整理如表 2。

在本題中，請按照前述規則讓每位患者排隊與接受看診，最終輸出醫師 1 和醫師 2 的看診人數 D_1 和 D_2。以前面這個例子來說，$D_1 = 4$ 且 $D_2 = 2$。

特別說明 1：在實作本題時，建議以類別和物件來追蹤每位患者在診所內的狀態。大家可以建立一個名爲 `Patient` 的類別，當有新患者到達時，就爲該患者建立一個物件。透過此物件，我們可以記錄患者的到達時間、年齡、看診時間長等資訊，更可以在後續追蹤並更新每位患者在任一時刻的排隊位置和看診時間，進而做出正確的計算。

項目	抵達順序 i					
	1	2	3	4	5	6
爲其看診之醫師編號	1	2	2	1	1	1
在醫師 1 的看診序號	1	–	–	2	3	4
在醫師 2 的看診序號	–	1	2	–	–	–
開始接受看診時間	6	9	17	11	17	25
看診完成時間 f_i	11	17	21	17	19	28
等待時間長 w_i	0	2	9	2	7	0

表 2: 推算結果

特別說明 2：當患者完成看診並離開診所後，應該及時刪除對應的患者物件，以避免大家的程式佔用過多記憶體。爲了讓大家感受到這麼做的好處，在本題中的患者人數可能會很多，如果大家不用合適的方式建立與刪除物件，就可能會因爲使用過多記憶體而無法得到分數。

輸入輸出格式

系統會提供一共 15 組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有 $n+1$ 列，第一列含有三個整數，依序是 n、b、L；在從第二列到第 $n+1$ 列中，每一列存著三個正整數，依序是患者 i 的 g_i、a_i、s_i。已知 $1 \le n \le 70000$、$0 \le b \le 20$、$1 \le L \le 1000$、$10 \le g_i \le 100$、$1 \le a_1 < a_2 < a_3 < \cdots < a_n \le 100000000$、$1 \le s_i \le 1000$。

讀入這些資訊後，請依照題目指定的規則計算 D_1 和 D_2，並將這些數字依序印出，相鄰的兩個數字中間用一個逗點隔開。舉例來說，如果輸入是

```
6 15 3
38 6 5
40 7 8
34 8 4
52 9 6
65 10 2
80 25 3
```

則輸出應該是

```
4,2
```

如果輸入是

```
6 10 4
38 6 5
37 7 8
34 8 4
52 9 6
65 19 2
```

```
80 25 3
```

則輸出應該是

```
6,0
```

如果輸入是

```
8 5 2
20 10 3
25 11 4
30 12 5
35 14 6
40 15 4
45 16 5
50 17 6
55 20 3
```

則輸出應該是

```
5,3
```

你上傳的原始碼裡應該包含什麼

你的 .cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。針對這個題目，你可以使用任何方法。

評分原則

這一題的所有分數都根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。

第二題

（20 分）在本題中，我們想寫一個簡單的「行事曆」應用程式，讓多位使用者各擁有一個行事曆，以記錄他們建立的「活動」（event，也可以說是「事件」，但總之在本題中我們稱之爲活動），包括活動的名稱、日期、開始時間與結束時間，並允許使用者新增、修改和刪除指定活動。在我們的行事曆中，不同的活動可以有相同的名稱，但不同的活動彼此不能有時間上的衝突（重疊），且一個活動的時長至少要有一分鐘。當然，一個活動的結束時間不能早於開始時間。每個行事曆中能記錄的活動數量有共同的上限，而這個上限可能會隨著應用程式擴充資源而增加（如果要增加，也是所有行事曆的活動數量上限一起增加到同一個數字），但一定不會減少。

爲了實現這一目標，我們首先，我們會定義兩個結構 `Date` 和 `Time` 以代表日期和時間（時間方面只精確到分），以及一個類別 `Event`。這些結構和類別大概會長得像這樣：

```
struct Date
{
    int year;
    int month;
    int day;
};

struct Time
{
    int hour;
    int minute;
};

class Event
{
protected:
    Date date;
    Time begin;
    Time end;
    string name;
    // some other things that you want to add
public:
    Event(Date date, Time begin, Time end, string name = "");
    void modify(Date date, Time begin, Time end, string name = "");
    void print() const;
    bool isConflictWith(const Event& e) const;
    // some other things that you want to add
};
```

根據以上程式碼，Event 包含以下內容，

- Date date：活動的日期。在本應用程式中，我們不允許活動跨日，因此每個活動都只會有一個活動日期。
- Time begin 和 Time end：分別表示活動的開始和結束時間。
- string name：活動的名稱。
- Event(Date date, Time begin, Time end, string name = "")：帶有日期、開始時間、結束時間和名稱的 constructor。如果使用者新增一個活動時沒有給定活動名稱，就預設爲空字串。
- modify(Date date, Time begin, Time end, string name = "")：當這個函數被呼叫，該活動的日期、開始時間、結束時間和名稱應該被按照傳入的引數修改；如果呼叫時沒有傳入名稱，則名稱應該被設定爲空字串。

- `bool isConflictWith(const Event& e)`：判斷呼叫此函數的活動物件和被傳入此函數的活動物件是否在時間上有衝突（重疊），如果有則回傳 true，否則回傳 false。如果一個活動的結束時間恰好爲另一個活動的開始時間，視爲不衝突。
- `void print()`：印出活動的基本資訊。

爲了有效管理這些活動，我們進一步定義了一個類別 `Calendar`，用來裝一個使用者的多個活動物件（精確地來說，是指向那多個活動物件）：

```
class Calendar
{
private:
    string ownerName;
    Event** events;
    int eventCnt;
    static int eventCntMax;
    // some other things that you want to add
public:
    Calendar(string ownerName);
    Calendar(const Calendar& c);
    ~Calendar();
    void setOwnerName(string newOwnerName);
    string getOwnerName() const;
    int getEventCnt() const;
    bool addEvent(Date date, Time begin, Time end, string name);
    bool removeEvent(Date date, Time begin, string name);
    bool modifyEvent(Date date, Time begin, string name, Date newDate,
                     Time newBegin, Time newEnd, string newName);
    void printEvent(Date date, Time begin, string name) const;
    void printCalendarOldToNew() const;
    static void setEventCntMax(int eventCntMax);
    // some other things that you want to add
};
```

根據以上程式碼，`class Calendar` 包含以下內容：

- `string ownerName`：該行事曆的擁有者名稱。
- `Event** events`：指向活動指標陣列的指標，相當於此使用者的多個活動。運作時，`events[i]` 是指向此使用者的第 $i+1$ 個活動物件的指標，而 `*events[i]` 即爲此使用者的第 $i+1$ 個活動物件。
- `int eventCnt`：目前行事曆中的活動數量。
- `static int eventCntMax`：靜態變數，表示整個應用程式中所有行事曆可以儲存的活動數量上限。

- `Calendar(string ownerName)`：根據傳入的引數值初始化擁有者名稱，並將 `events` 設定爲 `nullptr`、將 `eventCnt` 設爲 0 的 constructor。
- `Calendar(const Calendar& c)`：copy constructor，將傳入的行事曆參照的資料深層複製後建立一個新的行事曆物件。
- `~Calendar()`：釋放動態配置的記憶體的 destructor。
- `void setOwnerName(string newOwnerName)`：設定行事曆的擁有者名稱。
- `string getOwnerName() const`：回傳行事曆的擁有者名稱。
- `int getEventCnt() const`：回傳行事曆中的活動數量。
- `bool addEvent(Date date, Time begin, Time end, string name)`：若活動數量尚未達到上限，且欲新增的活動沒有跟既有活動衝突，則將新活動新增至行事曆中。此函數會根據傳入的引數值動態建立一個新的活動物件，並讓正確的指標指向這個活動物件，並且回傳 true。如果活動數量已經達到上限了，則不做任何事情並且直接回傳 false。
- `bool removeEvent(Date date, Time begin, string name)`：從行事曆中移除在指定日期、指定開始時間、指定名稱的活動。若存在符合條件的活動，則釋放其記憶體空間，並且回傳 true。如果符合條件的活動不存在，則不做任何事情並且直接回傳 false。
- `bool modifyEvent(Date date, Time begin, string name, Date newDate, Time newBegin, Time newEnd, string newName)`：在行事曆中按照前面三個引數值尋找指定日期、指定開始時間、指定名稱的活動。若存在符合條件的活動，則嘗試按照後面四個引數值修改該活動的資訊；如果這樣的修改不會和其他活動衝突，則進行這樣的修改並且回傳 true，反之則不修改該物件並且回傳 false。如果符合條件的活動不存在，則不做任何事情並且直接回傳 false。
- `void printEvent(Date date, Time begin, string name) const`：在行事曆中按照前面三個引數值尋找指定日期、指定開始時間、指定名稱的活動，如果存在則將之印出，不存在則印出空字串。
- `void printCalendarOldToNew() const`：將行事曆中的所有活動按照開始時間的先後順序依序印出。
- `static void setEventCntMax(int eventCntMax)`：設定 `eventCntMax` 的值。

大家應該有注意到題目一開始有關於行事曆和活動的一些限制，換言之，我們寫的 `Calendar` 和 `Event` 的成員函數和靜態函數應該做相對應的檢查，以避免系統中出現不合理的行事曆（例如活動個數超過上限）與活動（例如開始時間和結束時間相同、開始時間晚於結束時間、發生在 13 月或 35 號、發生在 -3 點或 78 分等等）。舉例來說，在 `addEvent` 這個函數中，應該要檢查傳入的資訊是否可以構成一個合理的活動，如果不行的話就拒絕新增活動，依此類推。具體的實作細節就留給大家規劃了。

系統的使用者會不定時地進行多種操作，包括新增活動、修改活動資訊、刪除活動等，並且這些操作可能會交錯發生。在每次遇到新增活動的任務時，如果該活動的資訊合理、該使用者還有新增活動的額度、該準備被新增的活動沒有跟該行事曆的既有活動衝突，請根據給定的活動名稱、日期、開始時間和結束時間，創建對應的活動物件，並將其加入到使用者的行事曆中。在每次遇到修改活動的任務時，如果找得到指定任務、新資訊合理、該準備被修改的活動在修改後沒有跟該行事曆的既有活動衝突，請

作相對應的修改。若是刪除任務，如果找得到指定任務，則依照指定的活動名稱將該活動從行事曆中移除。此外，應用程式可能會在某些時候調整每本行事曆的活動個數上限，如果那次調整是提升上限而非減少上限，就請照著提升上限。

在本題中，你將會收到許多的操作記錄，我們要請你根據這些操作記錄維護數個行事曆。如果有任何操作是不合理的（例如想調降活動個數上限），則跳過那個操作。最後你會收到一個查詢任務，要查詢某位使用者的所有活動，屆時請請先印出該擁有者的名稱，再將該使用者的所有活動按照時間先後順序由先到後逐一印出。

特別說明：最後，雖然這一題不使用 `class` 也能完成，但我們其實是透過這些題目幫大家示範合適的程式架構，並協助大家提升開發效率與程式的可讀性。因此，我們建議使用如前所述 `Event` 類別和 `Calendar` 類別來架構你的程式，並運用物件導向的思維來設計。上方的 `struct` 和 `class` 只是範例，你可以進行延伸，也可以自行修改（例如將 `struct` 改成 `class`，爲 `class` 新增成員函數等），不必拘泥於範例的具體寫法。

輸入輸出格式

系統會提供一共 10 組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有 $t+2$ 列：

- 第一列包含兩個正整數，第一個數字是 t，代表後面對於這個系統總共會有 t 次操作（包含創建、設定、添加等等）；第二個數字是剛開始時行事曆中可以儲存的活動數量上限。
- 第二列起到第 $t+1$ 列中的每一列代表一次操作，可能的操作類型與輸入格式如下：
  - 若要建立一個行事曆，則該列會依序包含兩個大寫字元「CC」、一個空白字元，和一個頭尾爲雙引號的字串代表建立行事曆的使用者名稱。若此使用者的行事曆已經存在，則無視此指令。
  - 若要新增一個活動，則該列會依序包含兩個大寫字元「AE」、一個空白字元、一個頭尾爲雙引號的字串代表行事曆擁有者的名稱、一個空白字元、一個「`YYYY/MM/DD`」格式的字串代表活動日期、一個空白字元、一個「`HH:MM`」格式的字串代表活動開始時間、一個空白字元、一個「`HH:MM`」格式的字串代表活動結束時間、一個空白字元，以及一個頭尾爲雙引號的字串代表活動名稱。
  - 若要刪除一個活動，則該列會依序包含兩個大寫字元「RE」、一個空白字元、一個頭尾爲雙引號的字串代表行事曆擁有者的名稱、一個空白字元、一個頭尾爲雙引號的字串代表想刪除之活動的名稱、一個空白字元、一個「`YYYY/MM/DD`」格式的字串代表想刪除之活動的日期、一個空白字元、一個「`HH:MM`」格式的字串代表想刪除之活動的開始時間。
  - 若要修改一個活動，則該列會依序包含兩個大寫字元「ME」、一個空白字元、一個頭尾爲雙引號的字串代表行事曆擁有者的名稱、一個空白字元、一個頭尾爲雙引號的字串代表想修改之活動的名稱、一個空白字元、一個「`YYYY/MM/DD`」格式的字串代表想修改之活動的日期、一個空白字元、一個「`HH:MM`」格式的字串代表想修改之活動的開始時間、一個空白字元、一個「`YYYY/MM/DD`」格式的字串代表該活動的新日期、一個空白字元、一個「`HH:MM`」格式的字串代表該活動的新開始時間、一個空白字元、一個「`HH:MM`」格式的字串代表該活動的新結束時間、一個空白字元、一個頭尾爲雙引號的字串代表該活動的新名稱。
  - 若要擴充行事曆的活動個數上限，則該列會依序包含兩個大寫字元「EX」、一個空白字元、一個整數代表欲設定的活動個數上限。

- 最後一列是針對一個行事曆的查詢，該列會依序包含一個大寫字元「Q」、一個空白字元、一個頭尾爲雙引號的字串代表行事曆擁有者的名稱。

在所有的輸入資料中，當出現頭尾爲雙引號的字串時，該字串去除掉頭尾的雙引號之後留下的即爲使用者名稱或活動名稱；使用者名稱和活動名稱都只會包含大小寫英文和空白字元，長度不超過 100 個字元，同一個字母的大寫和小寫被視爲不同字元；時間一律以二十四小時制表達，且一天的最後一個時刻爲 24:00，某一天的 24:00 即爲隔天的 00:00；給定的日期和時間都可能不合理（例如可能會出現 2023/2/29 或 2023/14/5 這種日期，也有可能出現 25:15 或 13:65 這種時間），給定的使用者也可能不存在，但年份一定會不早於 1900。已知 $1 \leq t \leq 1000$，且最後一列要查詢的使用者一定存在。

讀取以上資訊後，請就每個操作檢查是否應該執行該操作，如果應該執行則執行，若不該執行則不做任何動作，最後根據最後一列的要求輸出一位使用者的相關資訊。針對最後一列被查詢的使用者，請先輸出一列字串是他的使用者名稱，接著輸出若干列字串，每一列是該使用者的一個活動的相關資訊，按照活動開始時間由早到晚依序輸出。輸出一個活動的資訊時，請依序輸出其名稱、一個冒號和一個空白字元、一個「YYYY/MM/DD」格式的活動日期、一個逗號和一個空白字元、一個「HH:MM」格式的活動的開始時間、一個逗號和一個空白字元、「HH:MM」格式的活動的結束時間、一個句號。

舉例來說，如果輸入是

```
12 4
CC "Bob"
CC "Mary"
CC "John"
AE "Bob" 2024/11/13 13:00 15:00 "PD Quiz"
AE "Mary" 2024/11/14 15:00 16:00 "Calculus"
AE "Bob" 2024/11/20 19:00 20:00 "Concert"
AE "Bob" 2024/11/20 19:59 20:01 "Call John"
AE "Bob" 2024/11/20 20:30 20:01 "Call Mary"
ME "Bob" "PD Quiz" 2024/11/13 13:00 2024/11/13 16:00 18:00 "PD Quiz"
RE "Mary" "Calculus" 2024/11/14 15:00
ME "Bob" "Calculus" 2024/11/20 20:00 2024/11/13 16:00 18:00 "Calculus"
RE "Bob" "Concert" 2024/11/14 15:00
Q "Bob"
```

則輸出應該是

```
Bob
PD Quiz: 2024/11/13, 16:00, 18:00.
Concert: 2024/11/20, 19:00, 20:00.
```

請留意嘗試新增 Bob 的「Call John」會失敗，因爲時間和「Concert」衝突；嘗試新增 Bob 的「Call Mary」也會失敗，因爲結束時間早於開始時間；嘗試修改 Bob 的「Calculus」也會失敗，因爲 Bob 沒有名爲「Calculus」的活動；嘗試刪除 Bob 的「Concert」也會失敗，因爲 Bob 的「Concert」活動不是在 2024/11/14 的 15:00 開始。

如果輸入是

```
10 3
CC "Will"
AE "Will" 2024/13/12 21:00 24:00 "Sleep"
AE "Will" 2024/12/13 21:00 24:00 "Sleep"
AE "Will" 2024/12/14 00:00 08:00 "Sleep"
AE "Will" 2024/12/03 06:00 07:00 "Running"
AE "Will" 2024/12/03 08:00 10:00 "Breakfast"
EX 1
EX 5
AE "Will" 2024/12/03 12:00 14:00 "Lunch"
ME "Will" "Running" 2024/12/03 06:00 2024/12/03 16:00 20:00 "Running"
Q "Will"
```

則輸出應該是

```
Will
Lunch: 2024/12/03, 12:00, 14:00.
Running: 2024/12/03, 16:00, 20:00.
Sleep: 2024/12/13, 21:00, 24:00.
Sleep: 2024/12/14, 00:00, 08:00.
```

請留意第一次嘗試新增「Sleep」會失敗，因爲日期不合理；嘗試新增「Breakfast」也會失敗，因爲 Will 的行事曆已經滿了，此時無法再新增活動；嘗試把行事曆的活動個數上限修改成 1 也會失敗，因爲活動個數上限只能增加，不能減少。

你上傳的原始碼裡應該包含什麼

你的 .cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。針對這個題目，你可以使用任何方法。

評分原則

這一題的所有分數都根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。

第三題

（50 分）承上題，在現有的基礎上，系統現在需要引入兩種類型的活動，分別是「帶提醒的活動」以及「整日活動」。爲此，我們要請你寫兩個新的類別 `AlertEvent` 和 `WholeDayEvent`，並且建議你繼承自 `Event`。

`AlertEvent` 這個子類別大概長得像下面這樣：

```
class AlertEvent : public Event
```

```
{
private:
    int aheadAlertMinutes;
    // some other things that you want to add
public:
    AlertEvent(Date date, Time begin, Time end, string name,
               int aheadAlertMinutes);
    int getAheadAlertMinutes() const;
    void modify(Date date, Time begin, Time end, string name,
                int aheadAlertMinutes);
    void print() const;
    // some other things that you want to add
};
```

根據以上程式碼，AlertEvent 比起 Event 新增了一個成員變數 aheadAlertMinutes，代表在活動開始前幾分鐘要傳訊息通知使用者活動即將開始。此外，AlertEvent 還可以（必須）新定義自己的 constructor、aheadAlertMinutes 的 getter、自己的 modify() 函數、自己的 print() 函數（行爲和 Event 的 print() 應該不同，例如應該印出提前通知的時間）等等。

WholeDayEvent 代表整日活動，亦即使用者要將這一整天的 24 小時都留給這個活動。因此，這個子類別大概長得像下面這樣：

```
class WholeDayEvent : public Event
{
public:
    WholeDayEvent(Date date, string name);
    void modify(Date date , string name);
    void print() const;
    // some other things that you want to add
};
```

WholeDayEvent 的 constructor 應該將時間設爲整天。除此之外，WholeDayEvent 還可以（必須）新定義自己的 modify() 函數（只能改日期及名稱，不能改起訖時間）、自己的 print() 函數（印出資訊的方式不同）等等。

在本題中，我們要請你實作 AlertEvent 和 WholeDayEvent，並且完成和前一題一樣的任務。當然，你讀取的一系列操作中，會有跟帶通知的活動以及整日活動有關的操作。上面的 AlertEvent 和 WholeDayEvent 的宣告只是給你參考而已，你可以按照這個架構去實作你自己的這兩個程式、實作我們建議的成員函數，當然也可以新增自己需要的成員變數或成員函數。

你應該會發現，如果在前一題你已經完成功能完整、正確、合理的 Event 和 Calendar 了，那對你這一題要完成 AlertEvent 和 WholeDayEvent 應該有很大的幫助。如果前一題的這兩個類別真的寫得很好，那在這一題你應該幾乎不用改 Event 和 Calendar 的任何內容（當然，如果有必要的話，請就回去修改吧）。此外，多型會讓我們很自然地可以用前一題的 Event** events 陣列去指向 AlertEvent 和 WholeDayEvent 的指標和物件。這些東西真的是很神奇，請試試看吧！

特別提醒：在本題中，如果使用複數個陣列來分別儲存三種活動，會在程式碼品質方面被扣分。

輸入輸出格式

系統會提供一共 15 組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有 $t+2$ 列，其架構與第二題的完全相同，既有操作類型的格式也與第二題的完全相同，但在第二列到第 $t+1$ 列之間，會有一些新的操作類型：

- 若要新增一個整日活動，則該列會依序包含三個大寫字元「AWE」、一個空白字元、一個頭尾爲雙引號的字串代表行事曆擁有者的名稱、一個空白字元、一個「YYYY/MM/DD」格式的字串代表活動日期、一個空白字元，以及一個頭尾爲雙引號的字串代表活動名稱。
- 若要刪除一個整日活動，則該列會依序包含三個大寫字元「RWE」、一個空白字元、一個頭尾爲雙引號的字串代表行事曆擁有者的名稱、一個空白字元、一個頭尾爲雙引號的字串代表想刪除之活動的名稱、一個空白字元，以及一個「YYYY/MM/DD」格式的字串代表想刪除之活動的日期。
- 若要修改一個整日活動，則該列會依序包含三個大寫字元「MWE」、一個空白字元、一個頭尾爲雙引號的字串代表行事曆擁有者的名稱、一個空白字元、一個頭尾爲雙引號的字串代表想修改之活動的名稱、一個空白字元、一個「YYYY/MM/DD」格式的字串代表想修改之活動的日期、一個空白字元、一個「YYYY/MM/DD」格式的字串代表該活動的新日期、一個空白字元，以及一個頭尾爲雙引號的字串代表該活動的新名稱。
- 若要新增一個帶提醒的活動，則該列會依序包含三個大寫字元「AAE」、一個空白字元、一個頭尾爲雙引號的字串代表行事曆擁有者的名稱、一個空白字元、一個「YYYY/MM/DD」格式的字串代表活動日期、一個空白字元、一個「HH:MM」格式的字串代表該活動的開始時間、一個空白字元、一個「HH:MM」格式的字串代表該活動的結束時間、一個空白字元、一個頭尾爲雙引號的字串代表活動名稱、一個空白字元，以及一個正整數代表在開始時間的幾分鐘前要傳訊息提醒使用者。
- 若要刪除一個帶提醒的活動，則該列會依序包含三個大寫字元「RAE」、一個空白字元、一個頭尾爲雙引號的字串代表行事曆擁有者的名稱、一個空白字元、一個頭尾爲雙引號的字串代表想刪除之活動的名稱、一個空白字元、一個「YYYY/MM/DD」格式的字串代表想刪除之活動的日期、一個空白字元、一個「HH:MM」格式的字串代表想刪除之活動的開始時間。
- 若要修改一個帶提醒的活動，則該列會依序包含三個大寫字元「MAE」、一個空白字元、一個頭尾爲雙引號的字串代表行事曆擁有者的名稱、一個空白字元、一個頭尾爲雙引號的字串代表想修改之活動的名稱、一個空白字元、一個「YYYY/MM/DD」格式的字串代表想修改之活動的日期、一個空白字元、一個「HH:MM」格式的字串代表想修改之活動的開始時間、一個空白字元、一個「YYYY/MM/DD」格式的字串代表該活動的新日期、一個空白字元、一個「HH:MM」格式的字串代表該活動的新開始時間、一個空白字元、一個「HH:MM」格式的字串代表該活動的新結束時間、一個空白字元、一個頭尾爲雙引號的字串代表該活動的新名稱、一個空白字元，以及一個正整數代表在開始時間的幾分鐘前要傳訊息提醒使用者。

三個型態的活動在要被修改和刪除時，該列開頭都會是對應該型態的提示詞，不會發生型態間的混淆。換言之，開頭如果是「RWE」或「MWE」，後面跟著的使用者與活動如果存在，該活動就一定是一個整日活動；開頭如果是「RAE」或「MAE」，後面跟著的使用者與活動如果存在，該活動就一定是一個帶

提醒的活動；開頭如果是「RE」或「ME」，後面跟著的使用者與活動如果存在，該活動就一定是一個一般活動。

讀取以上資訊後，請就每個操作檢查是否應該執行該操作，如果應該執行則執行，若不該執行則不做任何動作，最後根據最後一列的要求輸出一位使用者的相關資訊。針對最後一列被查詢的使用者，請先輸出一列字串是他的使用者名稱，接著輸出若干列字串，每一列是該使用者的一個活動的相關資訊，按照活動開始時間由早到晚依序輸出。輸出一個活動的資訊時：

- 如果是一個一般活動（在第二題中定義的），請依序輸出其名稱、一個冒號和一個空白字元、一個「YYYY/MM/DD」格式的活動日期、一個逗號和一個空白字元、一個「HH:MM」格式的活動的開始時間、一個逗號和一個空白字元、「HH:MM」格式的活動的結束時間、一個句號。
- 如果是一個整日活動，請依序輸出其名稱、一個冒號和一個空白字元、一個「YYYY/MM/DD」格式的活動日期、一個逗號和一個空白字元、一個字串「whole day」、一個句號。
- 如果是一個帶提醒的活動，請依序輸出其名稱、一個冒號和一個空白字元、一個「YYYY/MM/DD」格式的活動日期、一個逗號和一個空白字元、一個「HH:MM」格式的活動的開始時間、一個逗號和一個空白字元、「HH:MM」格式的活動的結束時間、一個逗號和一個空白字元、一個整數代表要提早幾分鐘提醒、一個句號。

舉例來說，如果輸入是

```
8 6
CC "El ST"
AWE "El ST" 2024/07/19 "Field Trip"
AE "El ST" 2024/06/20 19:00 20:00 "JJ meeting"
ME "Charlie" "Science" 2024/08/01 10:00 2024/08/01 10:00 16:00 "Science"
CC "Diana"
RE "Alice" "Health Checkup" 2024/10/01 10:00
AWE "Diana" 2024/12/15 "Holiday Celebration"
AAE "El ST" 2024/07/18 11:00 13:00 "Client Call" 15
Q "El ST"
```

則輸出應該是

```
El ST
JJ meeting: 2024/06/20, 19:00, 20:00.
Client Call: 2024/07/18, 11:00, 13:00, 15.
Field Trip: 2024/07/19, whole day.
```

請留意修改「Charlie」的「Science」活動會失敗，因爲使用者「Charlie」不存在；刪除「Alice」的「Health Checkup」活動會失敗，因爲使用者「Alice」不存在。

如果輸入是

```
15 3
CC "Jane"
AWE "Jane" 2024/06/21 "Summer Solstice Festival"
```

```
AAE "Jane" 2024/06/23 09:00 11:00 "Meeting" 30
AAE "Jane" 2024/06/20 08:00 10:00 "Yoga Class" 10
AWE "Jane" 2024/12/31 "New Year Eve"
EX 9
AE "Jane" 2024/05/15 13:00 15:00 "Lunch with Sarah"
CC "Alice"
AWE "Jane" 2024/12/24 "Christmas Eve"
AE "Alice" 2024/06/15 17:00 18:00 "Project Discussion"
RAE "Jane" "Yoga Class" 2024/06/20 08:00
MAE "Jane" "Meeting" 2024/06/23 09:00 2024/06/23 09:00 11:00 "Meeting" 20
AE "Jane" 2024/05/17 15:00 15:30 "Reading"
MWE "Jane" "Christmas Eve" 2024/12/24 2024/12/34 "Christmas Eve"
AWE "Jane Chen" 2025/01/01 "New Year"
Q "Jane"
```

則輸出應該是

```
Jane
Lunch with Sarah: 2024/05/15, 13:00, 15:00.
Reading: 2024/05/17, 15:00, 15:30.
Summer Solstice Festival: 2024/06/21, whole day.
Meeting: 2024/06/23, 09:00, 11:00, 20.
Christmas Eve: 2024/12/24, whole day.
```

請留意新增「New Year Eve」會失敗，因爲行事曆空間不足；把「Christmas Eve」改到 2024/12/34 也會失敗，因爲日期不合理；新增「New Year」也會失敗，因爲使用者「Jane Chen」不存在。

你上傳的原始碼裡應該包含什麼

你的 .cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。針對這個題目，你***不可以***使用上課沒有教過的方法：

- 確定可以使用的語法包含繼承、多型，以及在之前的作業中正面表列過的語法。
- 確定不可以使用的語法包含 `printf`、`scanf`、template、`<vector>`、例外處理等等。

請注意正面表列的固然是都確定可以用，但沒有被負面表列的不表示可以用喔！

評分原則

- 這一題的其中 30 分會根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。
- 這一題的其中 20 分會根據你所寫的程式的品質來給分。助教會打開你的程式碼並檢閱你的程式的結構、運算邏輯、可讀性（包含排版、變數命名、註解等等）、可擴充性、模組化程度，以及是否使用了還沒教過的語法。請寫一個「好」的程式吧！

第四題

特別說明：本題爲作文題而非程式題，請不要上傳東西到 PDOGS；請把你的答案放在 PDF 檔中上傳到 COOL。由於班上同學很多，但助教人力有限，所以原則上我們會從所有繳交中隨機批改一部份（可能是 50%、90%、100%，但不會低於 50%）。沒被抽到的同學如果有認真回答，在本題就會得到滿分，但不會得到直接的助教回饋，如果沒回答或隨便亂寫當然就不會得分；有被抽到的同學們可能會被扣分，但相對應地也會得到助教的回饋。

（20 分；每小題 10 分）請回答以下兩題。在回答時，都不用寫程式碼，只要文字敘述即可。

1. 如果你希望能有會員等級制，讓不同等級的使用者有不同的活動個數上限（例如免費會員是 500 個、白銀會員是 5000 個、黃金會員是無上限），你會怎麼修改你的程式？你會使用繼承嗎？爲什麼？

2. 如果在第三題的程式中，你要允許系統中出現既帶提醒又整日的活動（有點怪，但也沒有不行），並且不能使用多重繼承（multiple inheritance），你該怎麼修改你的程式？你要新增一個類別，還是修改既有類別？請列出幾種作法，從中選擇一種，並且說明你爲什麼覺得你的作法是好的。