程式設計（113-1）
作業八

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/`）為第一、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 12 日早上八點***。為這份作業設計測試資料並且提供解答的助教是梁安哲。

在你開始前，請閱讀課本的第 11 章（operator overloading）[1]。

本次作業滿分為 110 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為 n 份作業的總分除以 n（不論超過 100 與否）。

第一題

說明：本題和作業五的第二題基本上是一樣的，但是有一些新的資訊和要求。為了讓大家不用回頭看題目，底下我們重新敘述這個題目一次。

（20 分）在一個國家裡有 n 個小鎮，某些小鎮間有路相連，總共有 m 條路，每條路都是起自一個小鎮也結束於一個小鎮，路上沒有別的小鎮。小鎮間的道路關係及道路長度可以用一個 $n \times n$ 的對稱矩陣 R 表示，其中 $R_{ij} = 0$ 表示小鎮 i 和小鎮 j 之間沒有路，若 $R_{ij} > 0$ 則表示小鎮 i 和小鎮 j 之間有路且該道路的長度即為 R_{ij}。以

$$R = \begin{bmatrix} 0 & 5 & 4 & 7 & 9 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 6 & 0 \\ 7 & 0 & 0 & 0 & 0 & 3 \\ 9 & 0 & 6 & 0 & 0 & 5 \\ 0 & 0 & 0 & 3 & 5 & 0 \end{bmatrix}$$

為例，表示 $n = 6$、$m = 7$，小鎮 1 有總共四條路通往小鎮 2、小鎮 3、小鎮 4 和小鎮 5，但沒有路通往小鎮 6，而從小鎮 1 到小鎮 2、小鎮 3、小鎮 4 和小鎮 5 的道路的長度依序是 5、4、7 和 9，依此類推。

已知在小鎮 i 上住有居民 h_i 人。國王想要看看各小鎮發展的情況，也拜訪民眾，因此想要從小鎮 1（其實是首都）出發，經過 q 個不同的小鎮後回到小鎮 1。他擬定了路線 $(1, p_1, p_2, ..., p_q, 1)$，表示先從小鎮 1 走到小鎮 p_1，再從小鎮 p_1 走到小鎮 p_2，依此類推，最後從小鎮 p_q 回到小鎮 1。舉例來說，$(1, 5, 3, 1)$ 就是一個合情合理的路線，沿路可以拜訪 $h_1 + h_5 + h_3$ 這麼多個民眾（小鎮 1 雖然被經過兩次，但被拜訪的民眾數只被計算一次），而此路線的總距離為 $R_{15} + R_{53} + R_{31} = 9 + 6 + 4 = 19$。

以上一切都很完美，唯一就是國王不太擅長擬定路線，因此國王把路線交給你，請你檢查他擬的路線上是否每條路都確實存在。舉例來說，如果國王擬的路線是 $(1, 5, 3, 2, 1)$，這個路線就行不通，因為從小鎮 3 沒有路通往小鎮 2。國王給你的任務是，如果給定的路線是可行的，就計算可以拜訪的民眾數和該路線的總距離；如果給定的路線不可行，就依照路線上的順序依序列舉不存在的路段。舉例來

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

說，如果國王給的路線是 $(1,5,1)$，就輸出 h_1+h_5 以及 $R_{15}+R_{51}=18$；如果是 $(1,5,4,2,1)$，就輸出 $(5,4)$、$(4,2)$。

特別說明 1：在本題中，小鎮數量可能高達一萬，若直接使用 adjacency matrix 儲存每個小鎮間的距離，將會造成記憶體大量浪費[2]。因此，在作業五的第二題我們已經改用 adjacency list 儲存道路資訊了，在本題你也需要這麼做才有機會獲得滿分。

特別說明 2：由於本題中新增了道路資訊，如果我們不使用 `struct`（結構）或 `class`（類別），我們會需要兩個 adjacency list，一個記錄每個小鎮的相鄰小鎮的編號，另一個記錄每個小鎮到相鄰小鎮的距離。由於這顯然不是太理想，在本題中我們建議你用 `struct` 和 `class` 來讓自己的程式更結構化。具體來說，我們建議你這麼做：

1. 寫一個 `Neighbor` 結構，裡面有兩個 instance variable，分別是一個整數 `ID` 代表小鎮編號，以及一個整數 `distance` 代表道路長度。之所以我們建議 `Neighbor` 是一個結構而非類別，是因為我們遵循一般寫 C++ 的原則，如果我們只是單純想把一些變數綁在一起，沒有要做複雜的操作，就建議把這個資料型態寫成結構，而非類別。由於結構的 instance variable 預設都是 public，因此也不用為它們寫 getter、setter 等函數。

2. 接著寫一個 `Town` 類別，裡面有四個 instance variable，分別是一個整數 `ID` 代表小鎮編號、一個整數 `population` 表示小鎮人口、一個整數 `neighborCnt` 代表小鎮的相鄰小鎮數量，以及一個型態為 `Neighbor**` 的指標 `neighbors`，這個指標指向一個動態生成的、長度為 `neighborCnt` 的 `Neighbor*` 指標陣列，該陣列的每個元素指向一個 `Neighbor` 物件，以記錄一個相鄰之小鎮的相關資訊。

3. 最終你會在 main function 裡面有一個型態為 `Town**` 的指標 `towns`，這個指標指向一個動態生成的、長度為 n 的 `Town*` 指標陣列，該陣列的每個元素指向一個 `Town` 物件，以記錄一個相鄰之小鎮的相關資訊。

圖 1 是用上述建議寫法儲存前述範例的示意圖。如圖所示，`towns` 是一個指標，指向一個長度為 $n=6$ 的指標陣列；`towns[0]` 也是一個指標，指向一個 `Town` 物件；`*towns[0]` 是一個 `Town` 物件，代表第一個小鎮；`towns[0]->neighbors` 是一個指標，指向一個長度為 4 的指標陣列；`towns[0]->neighbors[0]` 也是一個指標，指向一個 `Neighbor` 結構；`*(towns[0]->neighbors[0])` 是一個 `Neighbor` 結構，代表第一個小鎮的第一個相鄰的小鎮；最後，我們知道 `towns[0]->neighbors[0]->ID` 是 2，於此同時 `towns[0]->neighbors[0]->distance` 是 5，表示小鎮 1 距離小鎮 2 的距離是 5。

有了以上的資料結構後，你還可以再幫 `Town` 寫一個 member function。首先當然應該要有 constructor 和 destructor，另外如果有個 `int distanceToNeighbor(int townID)` 之類的 member function，可以回傳到編號為 `townID` 的相鄰小鎮的距離（如果不為鄰居則回傳 -1 之類的），那應該也很好用。這些就留給大家自行設計了[3]。

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 $m+3$ 列，第一列裝著三個正整數，依序是 n、m、q；在第二列到第 $m+1$ 列中，從第二列起算的第 j 列存有三個

[2]這樣的矩陣中絕大部分的值都是 0，被稱為「稀疏矩陣」，用 adjacency matrix 存很沒有效率。

[3]最後，本題是只看正確性、不看程式碼的，所以如果只是想要滿分，大家也可以完全不照上面的建議去實作。但我們當然是想透過這題引導大家練習實作和使用結構和類別，所以我們建議大家還是試試看吧！

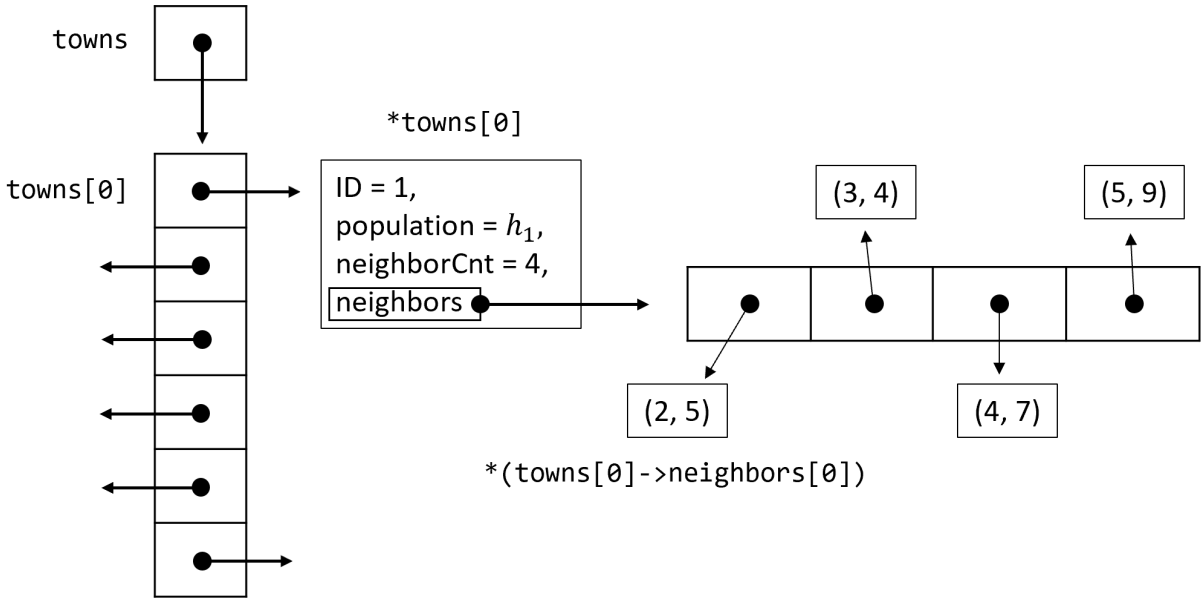

圖 1: 第一題的建議寫法示意圖

整數 u_j、v_j 和 R_{ij}，表示第 j 條路是連結小鎮 u_j 和小鎮 v_j，而此道路的長度為 R_{ij}；第 $m+2$ 列存了 n 個正整數，依序是 h_1、h_2 直到 h_n；第 $m+3$ 列存了 q 個正整數，依序是 p_1、p_2 直到 p_q。已知 $1 \le n \le 10000$、$1 \le m \le 100000$、$1 \le d_{ij} \le 1000$、$1 \le q \le n-1$、$u_j \in \{1, ..., n\}$、$v_j \in \{1, ..., n\}$、$1 \le h_i \le 1000$、$p_i \in \{2, ..., n\}$、$p_1 \ne p_2 \ne \cdots \ne p_q$、$[u_1, v_1] \ne [u_2, v_2] \ne \cdots \ne [u_m, v_m]$。每一列的任兩個相鄰的整數間以一個空白字元隔開。

讀入這些資訊後，如果指定路線上的每個路段都存在，就輸出一個整數代表路線上會拜訪的總民眾數，接著輸出一個逗點，接著輸出一個整數代表該路線的總距離；反之則按照路線順序輸出所有不存在的路段，每當要輸出一個路段時，先輸出該路段的起點小鎮編號，接著輸出一個逗點，再輸出該路段的終點小鎮編號，如果後面還有下一個路段，兩個路段間用一個分號隔開。舉例來說，如果輸入是

```
6 7 3
1 2 5
1 3 4
1 4 7
1 5 9
3 5 6
4 6 3
5 2 5
100 200 300 400 500 600
3 5 2
```

則因為 $(1, 3, 5, 2, 1)$ 路線上每個路段都存在，輸出應該是

```
1100,20
```

如果輸入是

```
6 7 3
1 2 4
1 3 6
1 4 9
1 5 10
4 5 8
4 6 15
4 2 18
100 200 300 400 500 600
3 5 2
```

則因為 $(1,3,5,2,1)$ 路線上 $(3,5)$、$(5,2)$ 路段不存在，輸出應該是

```
3,5;5,2
```

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只包含指定的函數 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）接續作業七的第二、三題，以下將再從頭敘述一次該題的相關題目要求，有一些細節與之前的不盡相同，建議大家在此詳細閱讀一次題目。

有一家診所，該診所有兩位醫師，編號分別為 1 和 2，且兩位醫師的看診速度相同。當兩位醫師都閒置（沒有在幫患者在看診）時，如果有一位新的患者出現，診所會安排編號為 1 的醫師看診；若新患者到達時只有一位醫師正在看診，診所會將該患者安排給閒置的醫師；若患者到達時兩位醫師都在看診或前面有更早來的患者在排隊，則診所會安排這位剛到的患者排隊。雖然有兩位醫師，但該診所總共只有一個等待序列，亦即所有患者都排同一個隊。

每天有許多患者陸續前來，這些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 個抵達的患者為患者 i，而患者 i 的年齡為 g_i、抵達時間為 a_i、看診時間長為 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，直到不能交換為止[4]。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。

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

根據以上資訊，我們可以推算每位病患在每個時間點的狀態。具體來說，對患者 i 我們可以推算他的開始接受看診時間、看診完成時間 f_i、等待時間長 w_i，以及患者 i 抵達的前一瞬間的診所內人數 p_i（因為是前一瞬間，所以不包含他自己；診所中的人數包含被看診中的和等待中的）。以下是兩個例子：

- 假設患者們的資訊如表 1 所示，$b = 15$：

項目	抵達順序 i					
	1	2	3	4	5	6
年齡 g_i	38	37	34	52	47	25
到達時間 a_i	6	15	18	19	27	38
看診時間長 s_i	5	8	4	6	2	3

表 1: 範例一的患者資訊

  - 第一位患者到達時由醫師 1 看診，看診完成時間為第 $f_1 = 6 + 5 = 11$ 分鐘；
  - 在第一位患者離開後，第二位患者到達，並由醫師 1 看診，其看診完成時間為第 $f_2 = 15 + 8 = 23$ 分鐘；
  - 第三位患者在第 18 分鐘到達，只有醫師 2 在空閒，因此第三位患者由醫師 2 看診，看診完成時間為第 $f_3 = 18 + 4 = 22$ 分鐘；
  - 第四位患者在第 19 分鐘到達診所，此時第二、三位抵達的患者已經在接受看診了，因此即使第四位患者的年齡比第三位患者大 15 歲以上，第四位患者仍需等待。醫師 1 需要等到第二位患者在第 23 分鐘離開後才能幫下一位患者看診，而醫師 2 則在第 22 分鐘第三位患者離開後可以幫下一位患者看診。因此，第四位患者將由醫師 2 看診，等待時間為 $w_4 = 22 - 19 = 3$，看診完成時間為 $f_4 = 22 + 6 = 28$；
  - 在第四位患者離開前，第五位患者於第 27 分鐘到達，且此時醫師 1 已空閒，因此第五位患者由醫師 1 看診，到達時間即為開始接受看診的時間，其看診完成時間為 $f_5 = 27 + 2 = 29$；
  - 當第四位和第五位患者都離開後，第六位患者於第 38 分鐘到達，此時兩位醫師均空閒，因此第六位患者由醫師編號 1 看診，其看診完成時間為 $f_6 = 38 + 3 = 41$。

根據以上討論，推算的資訊可以被整理如表 2。

項目	抵達順序 i					
	1	2	3	4	5	6
醫師編號	1	1	2	2	1	1
看診序號	1	2	3	4	5	6
開始接受看診時間	6	15	18	22	27	38
看診完成時間 f_i	11	23	22	28	29	41
等待時間長 w_i	0	0	0	3	0	0
患者 i 到達前一瞬間的診所內人數 p_i	0	0	1	2	1	0

表 2: 範例一的推算結果

項目	抵達順序 i					
	1	2	3	4	5	6
年齡 g_i	38	37	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 3: 範例二的患者資訊

- 在範例二中，假設患者們的資訊如表 3 所示，$b = 10$：
  - 第一位患者到達時由醫師 1 看診，看診完成時間為第 $f_1 = 6 + 5 = 11$ 分鐘；
  - 第二位患者到達時由醫師 2 看診，看診完成時間為第 $f_2 = 7 + 8 = 15$ 分鐘；
  - 第三、四、五位患者依序在第 8、9、10 分鐘到達。他們到達時，醫師編號 1 在為患者 1 看診，醫師編號 2 在為患者 2 看診，因此他們三位都需要排隊。因為 $g_4 \geq g_3 + 10$，患者 4 比比他早到的患者 3 年長 10 歲以上，因此患者 4 會跟患者 3 交換看診序號；又因為 $g_5 \geq g_3 + 10$，患者 5 抵達時會先跟患者 3 交換看診序號，再因為 $g_5 \geq g_4 + 10$，患者 5 接著再跟患者 4 交換看診序號。最終結果是患者 3、4、5 的看診序號依序是 5、4、3；
  - 第一位患者在第 11 分鐘離開後，醫師 1 會為第五位患者看診，看診完成時間為第 $f_5 = 11 + 2 = 13$ 分鐘，等待時間為 $f_5 = 11 - 10 = 1$。請留意雖然患者 5 有跟比他早到的患者交換看診序號，但這跟患者 5 抵達時有多少人在診所中無關；當患者 5 到達時有兩位患者在看診中，兩位在排隊，因此 $p_5 = 4$；
  - 第五位患者在第 13 分鐘離開後，醫師 1 會繼續為第四位患者看診，看診完成時間為第 $f_4 = 13 + 6 = 19$ 分鐘，等待時間為 $w_4 = 13 - 9 = 4$；
  - 在第四位患者正在接受看診時，第二位患者於第 $f_2 = 15$ 分鐘離開，此時第三位患者會由醫師 2 看診，因此第三位患者的等待時間長為 $w_3 = 15 - 8 = 7$，看診完成時間則為 $f_3 = 15 + 4 = 19$；
  - 第六位患者於第 25 分鐘到達時，兩位醫師都在閒置，因此診所會安排由醫師編號 1 為他看診，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 4。

項目	抵達順序 i					
	1	2	3	4	5	6
醫師編號	1	2	2	1	1	1
看診序號	1	2	5	4	3	6
開始接受看診時間	6	7	15	13	11	25
看診完成時間 f_i	11	15	19	19	13	28
等待時間長 w_i	0	0	7	4	1	0
患者 i 到達前一瞬間的診所內人數 p_i	0	1	2	3	4	0

表 4: 範例二的推算結果

在本題中，診所希望在任意指定一個抵達順序 k 之後，你可以幫他們找出患者 k 的看診完成時間 f_k、從抵達診所到接受看診的等待時間長 w_k，以及他到達診所的前一瞬間診所中的總人數 p_k。診所也希望在給定時間點 t 後，你可以幫他們算出在第 t 分鐘那個瞬間的前一瞬間的診所中的總人數 $x(t)$。

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。在每個檔案中會有五列，其中第一列裝著兩個正整數，依序是 n 和 b；在第二列裝著 n 個正整數，依序是 g_1、g_2、g_3 直到 g_n；在第三列裝著 n 個正整數，依序是 a_1、a_2、a_3 直到 a_n；第四列裝著 n 個正整數，依序是 s_1、s_2、s_3 直到 s_n；第五列裝著一個正整數 k、一個逗點、一個正整數 t。已知 $1 \le n \le 100$、$0 \le b \le 20$、$10 \le g_i \le 100$、$1 \le a_1 < a_2 < a_3 < \cdots < a_n \le 100000$、$1 \le s_i \le 1000$、$k \in \{1, 2, ..., n\}$、$1 \le t \le 100000$。

讀入這些資訊後，請依照題目指定的規則計算 f_k、w_k、p_k 和 $x(t)$，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
38 37 34 52 47 25
6 15 18 19 27 38
5 8 4 6 2 3
3,20
```

則輸出應該是

```
22,0,1,3
```

如果輸入是

```
6 10
38 37 34 52 65 80
6 7 8 9 10 25
5 8 4 6 2 3
4,17
```

則輸出應該是

```
19,4,3,2
```

如果輸入是

```
5 5
20 25 30 35 40
10 11 12 13 14
5 5 5 5 5
5,100000
```

則第五個抵達的患者會一路往前交換，變成第三個接受看診，但他抵達的時候（第 14 分鐘）診所內共有四個人，因此輸出應該是

```
20,1,4,0
```

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只包含指定的函數 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）矩陣是數學上非常廣用的工具，其應用涵蓋了數學、統計學、電腦科學以及各種自然科學。在線性代數之中，矩陣是線性轉換在實數空間的一種表示方法。在本題中要請大家練習使用 C++ 實作矩陣的基本運算，並將之運用在簡易的線性轉換。

題目中會給定一個線性轉換，暫且稱為 L，以及一個要用來被轉換的向量，暫且稱為 v。L 是由多個線性轉換組合而成的，在本題中會用一個公式表示之。舉例來說，如果 $L = 3A + B^{\mathrm{T}}C$，其中 B^{T} 是矩陣 B 的轉置矩陣，且

$$A = \begin{bmatrix} 9 & 7 & 1 \\ 2 & 3 & 5 \\ 7 & 0 & 8 \end{bmatrix}, B = \begin{bmatrix} 2 & 4 & 3 \\ 5 & 1 & 4 \end{bmatrix}, C = \begin{bmatrix} 2 & 2 & 0 \\ 6 & 5 & 4 \end{bmatrix}, v = \begin{bmatrix} 9 \\ 4 \\ 1 \end{bmatrix},$$

則可以計算出

$$L = \begin{bmatrix} 61 & 50 & 23 \\ 20 & 22 & 19 \\ 51 & 26 & 40 \end{bmatrix},$$

因此 v 經過 L 線性轉換後的結果就是

$$Lv = \begin{bmatrix} 772 \\ 287 \\ 603 \end{bmatrix}。$$

為了使矩陣的運算簡單化，請實作一個 class 叫 `Matrix`，並以 operator overloading 的方式定義其中的加、減、乘、轉置四種運算，以及輸入及輸出的 streaming operators。

輸入輸出格式

系統會提供數組測試資料，每組測試資料裝在一個檔案裡。每個檔案中會有若干列，包含一個運算式以及多個矩陣或向量。第一行是一個合乎數學規則的運算式，由大寫英文字母、數字、加號、減號和驚嘆號組成，不含其他沒提到的字元（包括空白字元）。在這個運算式中，驚嘆號代表將驚嘆號前的矩陣

轉置；已知最多只有一個驚嘆號。若這個運算式之中包含 k 個矩陣（可能會重複），則從第二列起，將會有最多 $k+1$ 個矩陣的資料，依序代表運算式之中的 k 個矩陣以及向量 v，矩陣出現的順序與其在運算式中第一次出現的出現順序相同。在第 i 個矩陣的表示中，第一列會有矩陣的名字（如上例中的 A、B、C、v），以及 2 個正整數 m_i 及 n_i，代表這個矩陣的列數跟行數，三者分別以空格隔開。接下來的 m_i 列中，每列會有 n_i 個整數，分別以空格隔開，代表這個矩陣中的元素 x_{ij}。已知 $1 \le k \le 25$、$-10 \le x_{ij} \le 10$、$1 \le m_i, n_i \le 100$，v 的寬度必定為 1。運算式之中只會出現加、減、乘、轉置之運算，不會出現括號，且若數字出現在一個交乘項中，一定會出現在最開頭的地方（例如 $4AB$ 是有可能的，$A4B$ 或 $2AB3$ 是不可能的）。矩陣的名字必定為一個大寫英文字母，向量的名字必定為一個小寫英文字母，任意一個交乘項中的數字一定是整數且為個位數（例如 $4AB$ 是有可能的，$24AB$ 或 $3.8AB$ 則不可能）。你可以假設運算過程中的所有數字都可以用 `int` 存得下，不會發生任何溢位。

讀入資料後，請計算出將最後一個向量（可能叫 v 也可能叫其他名稱）進行線性轉換後的向量，並輸出轉換後的結果，一列一個數字。如果遇到任何無效的運算（例如矩陣維度不合無法相乘），則輸出 -1。舉例來說，如果輸入是

```
3A+B!C
A 3 3
9 7 1
2 3 5
7 0 8
B 2 3
2 4 3
5 1 4
C 2 3
2 2 0
6 5 4
v 3 1
9
4
1
```

則輸出應該是

```
772
287
603
```

如果輸入是

```
AB+B
A 3 3
1 1 1
1 1 1
1 1 1
B 3 2
2 5
```

```
4 1
3 4
u 2 1
-1
0
```

則輸出應該是

```
-11
-13
-12
```

如果輸入是

```
A
A 3 3
1 1 1
1 1 1
1 1 1
u 2 1
-1
0
```

則輸出應該是

```
-1
```

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。

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運算邏輯、可讀性，以及可擴充性（順便檢查你有沒有使用上課沒教過的語法，並且抓抓抄襲）。除此之外，你***必須***實做題目指定的 class 和 operator overloading ，並且利用這些東西來完成運算。若你沒有這麼做，「程式的品質」部份將被扣分。請寫一個「好」的程式吧！

第四題

特別說明：本題為作文題而非程式題，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）請把你為第三題寫的 `Matrix` 類別拿出來重新檢視，並且判斷每一個成員函數是否應該是 constant 成員函數、每個成員函數的回傳值是否應該是 constant，以及每個成員函數的參數是否應該被加上 constant。作答時，請把這個類別的宣告（包含成員變數和成員函數的宣告，但不包含成員函數的定義）貼到這一題，在合適的地方加上關鍵字 `const`，並且在每一個有 `const` 的地方簡要地說明為什麼這裡應該要有 `const`。最後，請說明 `Matrix` 為什麼需要 constructor、copy constructor、assignment operator、destructor，以及這些函數應該扮演什麼功能、完成什麼任務。