

程式設計 (112-1)

作業七

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 7 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林元婷。

在你開始前，請閱讀課本的第 22.1–22.6 節 (關於 self-defined data types in C) 和第 9、10 章 (class) 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

(20 分) 有家公司請你寫程式維護他們家關於產品的資訊，並且提供某些搜尋、彙總功能，任務如下。該公司有 n 種產品，每種產品都有其產品名稱 (不重複且只包含英數和空白字元的字串)、單位零售價、單位生產成本和截至去年底為止的總銷售量資訊。我們稱呼產品 i 的產品名稱為 x_i 、單位零售價為 p_i 、單位生產成本為 c_i 、截至去年底為止的總銷售量為 q_i 。為了簡單起見，我們假設這些產品的成本和價格都從未改變過。

舉例來說，該公司可能有 $n = 4$ 種產品，其基本資訊如表 1 所示。若是如此，表 2 中即為各產品的單位毛利 (單位零售價減單位生產成本)，以及歷史總營收 (單位零售價乘以歷史總銷售量)、歷史總利潤 (單位毛利乘以歷史總銷售量)。

產品編號	產品名稱	單位零售價	單位生產成本	截至去年底的總銷售量
1	iPhone 41	200	120	1000
2	iPhone 41 Pro	250	140	800
3	WaterPods	40	20	1200
4	WaterPods Max	50	30	900

表 1: 範例產品資訊 (第一題)

該公司會對你的程式發動查詢，請你回傳根據某指標由大到小排序後前 k 名之產品的有史以來的利潤總和。可能用來排序的指標包含各產品的單位零售價、單位生產成本、單位毛利 (單位零售價減單位生產成本)、歷史總銷售量、歷史總營收 (單位零售價乘以歷史總銷售量)，以及歷史總利潤 (單位毛利乘以歷史總銷售量) 共六種。排序時如果遇到平手，一律依 `<cstring>` 函式庫裡面的 `strcmp` 對產品名稱的排序 (字典順序較前的排名在前)。

舉例來說，如果排序指標是總銷售量且 $k = 3$ ，則前三名分別是「WaterPods」、「iPhone 41」和

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

產品編號	產品名稱	單位毛利	總營收	總利潤
1	jPhone 41	80	200000	80000
2	jPhone 41 Pro	110	200000	88000
3	WaterPods	20	48000	24000
4	WaterPods Max	20	45000	18000

表 2: 範例歷史總資訊（第一題）

「WaterPods Max」，歷史利潤總和是 $24000 + 80000 + 18000 = 122000$ 。如果排序指標是總營收且 $k = 1$ ，則第一名是「iPhone 41」，歷史利潤總和是 80000；請注意「iPhone 41」和「iPhone 41 Pro」的總營收相同，則因為 `strcmp` 把「iPhone 41」排在「iPhone 41 Pro」前面，所以「iPhone 41」是第一名。

在本題中，我們建議你建立以下類別：

```
class Product
{
private:
    char* name;
    int price;
    int cost;
    int salesQty;
public:
    // n: name, p: price, c: cost, sq: sales quantity
    Product(char* n, int p, int c, int sq);
    ~Product();
    bool isInFrontOf(const Product& prod, int criterion);
};
// you may define more members by yourselves
```

其中 `isInFrontOf` 會視情況回傳 `true` 或 `false`：根據給定的排序條件 `criterion`，若呼叫此函數的物件應該排在傳入的物件 `prod` 之前，那就回傳 `true`，反之則回傳 `false`。接著請在 `main function` 裡面建立一個動態陣列（`productCnt` 是產品個數）：

```
int productCnt = 0;
cin >> productCnt;
Product** products = new Product*[productCnt];
```

每當讀入一個產品的相關資訊，就使用 `constructor` 建立一個 `Product` 物件（記得幫 `name` 動態地開一塊記憶體空間），並且把 `products` 裡的一個指標指向該物件。當看到排序任務時，就根據物件們的資訊「將指標們重新排序」，把指向排名較前之產品的指標放在 `products` 陣列的前面，最後再透過指標讀取 `products` 陣列中前 k 個產品的總利潤即可。你可以考慮看看「C strings」那一堂課接近尾聲處一個幫一堆人名排序的例子當參考。

雖然這一題不這樣寫也寫得出來，但後面還有其他題目是從此題做延伸，屆時若沒有 `Product` 類別、`products` 陣列，寫起來會更痛苦，所以我們建議在這一題就花點功夫熟悉語法，並且幫你的程式

建立良好的結構喔。

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 $n + 2$ 行，第一行包含一個整數 n ；第二行起的 n 行中，第 $i + 1$ 行包含產品 i 的產品名稱 x_i （只含有英數和空白字元的字串，且頭尾一定不會是空白）、單位零售價 p_i （整數）、單位成本 c_i （整數），以及截至去年底的總銷售量 q_i （整數）；每一行的 x_i 和 p_i 之間被一個逗點隔開，剩下的相鄰整數之間被一個空白字元隔開。最後一行包含兩個整數 y 和 k ，中間以一個空白字元隔開；當 y 的值是 1、2、3、4、5、6 時，排序條件分別是單位零售價、單位成本、單位毛利、總銷售量、總營收、總利潤。已知 $1 \leq k \leq n \leq 100$ 、 $y \in \{1, 2, \dots, 6\}$ 、 x_i 的長度介於 1 和 100 之間（包含 1 和 100）、 $1 \leq c_i \leq p_i \leq 1000$ 、 $0 \leq q_i \leq 10000$ 、任兩個 x_i 均不完全相同。

讀入這些資訊後，請依照指定的排序條件將產品排序，接著印出前 k 個產品的利潤總和。舉例來說，如果輸入是

```
4
jPhone 41,200 120 1000
jPhone 41 Pro,250 140 800
WaterPods,40 20 1200
WaterPods Max,50 30 900
4 3
```

則輸出應該是

```
122000
```

如果輸入是

```
4
jPhone 41,200 120 1000
jPhone 41 Pro,250 140 800
WaterPods,40 20 1200
WaterPods Max,50 30 900
5 1
```

則輸出應該是

```
80000
```

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) 承上題，任務和輸出格式都一樣，但現在輸入資料中沒有 n 的值。

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。在每個檔案中會有若干行，前 n 行中第 i 行包含產品 i 的產品名稱 x_i (只含有英數和空白字元的字串，且頭尾一定不會是空白)、單位零售價 p_i (整數)、單位成本 c_i (整數)，以及截至去年底의總銷售量 q_i (整數)；每一行的 x_i 和 p_i 之間被一個逗點隔開，剩下的相鄰整數之間被一個空白字元隔開。最後一行包含兩個整數 y 和 k ，中間以一個空白字元隔開；當 y 的值是 1、2、3、4、5、6 時，排序條件分別是單位零售價、單位成本、單位毛利、總銷售量、總營收、總利潤。若令產品資訊行數為 n ，則已知 $1 \leq k \leq n \leq 100$ 、 $y \in \{1, 2, \dots, 6\}$ 、 x_i 的長度介於 1 和 100 之間 (包含 1 和 100)、 $1 \leq c_i \leq p_i \leq 1000$ 、 $0 \leq q_i \leq 10000$ 、任兩個 x_i 均不完全相同。

讀入這些資訊後，請依照指定的排序條件將產品排序，接著印出前 k 個產品的利潤總和。舉例來說，如果輸入是

```
jPhone 41,200 120 1000
iPhone 41 Pro,250 140 800
WaterPods,40 20 1200
WaterPods Max,50 30 900
4 3
```

則輸出應該是

```
122000
```

如果輸入是

```
jPhone 41,200 120 1000
iPhone 41 Pro,250 140 800
WaterPods,40 20 1200
WaterPods Max,50 30 900
5 1
```

則輸出應該是

```
80000
```

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) 本題是第一題的延伸，但差異不小。為了方便大家閱讀，底下我們完整地敘述此題。

有家公司請你寫程式維護他們家關於產品的資訊，並且提供某些搜尋、彙總功能，任務如下。該公司有 n 種產品，每種產品都有其產品名稱（不重複且只包含英數和空白字元的字串）、單位零售價、單位生產成本和截至去年底為止的總銷售量資訊。我們稱呼產品 i 的產品名稱為 x_i 、單位零售價為 p_i 、單位生產成本為 c_i 、截至去年底為止的總銷售量為 q_i 。除了上述產品資訊，該公司還會給你 s 筆今年各產品的銷售紀錄，在每筆紀錄中有某產品的名稱及該產品在該次銷售中的銷售量。為了簡單起見，我們假設這些產品的成本和價格都從未改變過。

舉例來說，該公司可能有 $n = 4$ 種產品，其基本資訊如表 3 所示，而今年度的 $s = 6$ 次銷售紀錄則如表 4 所示。若是如此，表 5 中即為各產品的單位毛利（單位零售價減單位生產成本），以及包含今年所有銷售在內的歷史總銷售量、歷史總營收（單位零售價乘以歷史總銷售量），以及歷史總利潤（單位毛利乘以歷史總銷售量）。

產品編號	產品名稱	單位零售價	單位生產成本	截至去年底的總銷售量
1	iPhone 41	200	120	1000
2	iPhone 41 Pro	250	140	800
3	WaterPods	40	20	1200
4	WaterPods Max	50	30	900

表 3: 範例產品資訊（第三題）

該公司會時不時地對你的程式發動一共 t 次查詢，且查詢任務和銷售紀錄可能會交錯出現。在每次遇到查詢任務時，請根據當下的產品資訊（在計算歷史銷售量、營收和利潤時只計算到當下為止）按照指定排序指標由大到小將產品排序，再求出前 k 名的產品的有史以來的利潤總和。可能用來排序的指標包含各產品的單位零售價、單位生產成本、單位毛利（單位零售價減單位生產成本）、累積到查詢當下的歷史總銷售量、累積到查詢當下的歷史總營收（單位零售價乘以歷史總銷售量），以及累積到查詢當下的歷史總利潤（單位毛利乘以歷史總銷售量）共六種。排序時如果遇到平手，一律依 `<cstring>` 函式庫裡面的 `strcmp` 對產品名稱的排序（字典順序較前的排名在前）。

舉例來說：

- 如果有一次查詢任務在表 4 的第 6 次銷售後發生，排序指標是包含今年的總銷售量且 $k = 3$ ，則

銷售編號	產品名稱	銷售量
1	jPhone 41	40
2	WaterPods Max	50
3	WaterPods	80
4	jPhone 41 Pro	85
5	jPhone 41	45
6	jPhone 41 Pro	75

表 4: 範例銷售資訊

產品編號	產品名稱	單位毛利	總銷售量	總營收	總利潤
1	jPhone 41	80	1085	217000	86800
2	jPhone 41 Pro	110	960	240000	105600
3	WaterPods	20	1280	51200	25600
4	WaterPods Max	20	950	47500	19000

表 5: 範例歷史總資訊（後三欄均包含今年所有銷售）

前三名分別是「WaterPods」、「iPhone 41」和「iPhone 41 Pro」，累積到查詢當下的歷史利潤總和是 $25600 + 86800 + 105600 = 218000$ ；請注意是用累積到查詢當下的歷史總銷售量排序，不是到去年為止的。

- 如果有一次查詢任務在表 4 的第 3 次銷售後發生，排序指標是單位成本且 $k = 2$ ，則前兩名分別是「iPhone 41 Pro」和「iPhone 41」，累積到查詢當下的歷史利潤總和是 $88000 + 83200 = 171200$ ；請注意是用累積到查詢當下的歷史總銷售量排序，不是到去年為止的。
- 如果有一次查詢任務在表 4 的第 6 次銷售後發生，排序指標是單位毛利且 $k = 3$ ，則前三名分別是「iPhone 41 Pro」、「iPhone 41」和「WaterPods」，累積到查詢當下的歷史利潤總和是 $105600 + 86800 + 25600 = 218000$ ；請注意「WaterPods」和「WaterPods Max」的單位毛利相同，則因為 `strcmp` 把「WaterPods」排在「WaterPods Max」前面，所以「WaterPods」是第三名。

在本題中，你將被給定前述資訊及數次查詢，請依照查詢任務出現的順序，依序印出這些查詢得到的利潤總和。

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 $n + s + t + 1$ 行：

- 第一行包含三個整數 n 、 s 和 t ，相鄰兩個整數間以一個空白字元隔開。
- 第二行起的 n 行中，第 $i + 1$ 行包含產品 i 的產品名稱 x_i （只含有英數和空白字元的字串，且頭尾一定不會是空白）、單位零售價 p_i （整數）、單位成本 c_i （整數），以及截至去年底의總銷售量 q_i （整數）；每一行的 x_i 和 p_i 之間被一個逗點隔開，剩下的相鄰整數之間被一個空白字元隔開。

- 最後 $s + t$ 行包含兩種資訊。若該行代表一筆銷售紀錄，則該行會依序包含一個大寫字元「S」、一個逗點、一個字串代表產品名稱、一個逗點、一個整數 z 代表該產品在該筆銷售的銷售量；代表產品名稱的字串一定曾經出現在第二行到第 $n + 1$ 行的產品名稱中。若該行代表一次查詢，則該行會依序包含一個大寫字元「T」、一個空白字元、一個整數 y 、一個空白字元和一個整數 k ；當 y 的值是 1、2、3、4、5、6 時，排序條件分別是單位零售價、單位成本、單位毛利、總銷售量、總營收、總利潤。

已知 $1 \leq k \leq n \leq 100$ 、 $0 \leq s \leq 100$ 、 $1 \leq t \leq 10$ 、 $y \in \{1, 2, \dots, 6\}$ 、 x_i 的長度介於 1 和 100 之間（包含 1 和 100）、 $1 \leq c_i \leq p_i \leq 1000$ 、 $0 \leq q_i \leq 10000$ 、 $0 \leq z \leq 1000$ 、任兩個 x_i 均不完全相同。

讀入這些資訊後，請執行 t 次查詢，在每次查詢中依照指定的排序條件將產品排序，接著印出前 k 個產品的利潤總和。除了最後一次查詢，在印出利潤總和後請印出一個逗點；最後一次查詢印出的利潤總和後面應該沒有任何字元。舉例來說，如果輸入是

```
4 6 3
jPhone 41,200 120 1000
jPhone 41 Pro,250 140 800
WaterPods,40 20 1200
WaterPods Max,50 30 900
S,jPhone 41,40
S,WaterPods Max,50
S,WaterPods,80
T 2 2
S,jPhone 41 Pro,85
S,jPhone 41,45
S,jPhone 41 Pro,75
T 4 3
T 3 3
```

則輸出應該是

```
171200,218000,218000
```

如果輸入是

```
4 0 1
jPhone 41,200 120 1000
jPhone 41 Pro,250 140 800
WaterPods,40 20 1200
WaterPods Max,50 30 900
T 1 4
```

則輸出應該是

```
210000
```

你上傳的原始碼裡應該包含什麼

你的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自定義資料型態、類別，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`、運算子多載、C++ 字串、繼承、多型等等。

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約 50%。沒被抽到的同學在本題就會得到滿分，但不會得到直接的助教回饋；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）請敘述一個 C++ 程式開發的任務，是「用物件導向程式設計做會比用程序式程式設計」好很多的，說明其原因。所謂「程序式程式設計」（procedural programming）就是只使用 `if`、`else`、`for`、`while`、`array`、`function` 這些語法，整個程式的主體是演算法；「物件導向程式設計」（object-oriented programming）則是再加上使用 `structure`、`class` 等語法，整個程式的主體是物件之間的互動；要敘述一個「程式開發的任務」，則是要告訴大家所要完成的程式要有哪些功能、能做到哪些事。舉例來說，一個「行事曆」可能就是個好例子，因為一個使用者的行事曆中會有一堆日期和時間，如果完全不使用 `structure` 或 `class`，程式的可讀性、可維護性、可擴充性大概都不會太好，開發上也比較困難。

在本題中，我們強烈建議大家問 PCATS 這個問題，並且透過多次互動得到令人滿意、合適、夠完整的答案。之所以這樣建議，是因為 PCATS 以及各種生成式 AI 其實很能幫助大家學習程式設計。每當我們教一個新觀念（例如 `class`、動態記憶體配置、`constants`、`copy constructor` 等等），很多時候我們的困擾都是「為什麼要這麼做」，此時如果有很多例子，就可能有一些例子幫助我們理解，而能讓每個人理解的例子可能都不一樣，一個老師能舉的例子也有限，那麼請 PCATS 或生成式 AI 幫我們多舉幾個例子，讀完沒感覺的話就再一個，就很可能幫助大家瞭解這些新觀念要解決的問題、原理、適合使用的場景，以及使用的方法了！

以下是老師跟 PCATS 互動的過程，供大家參考：

- 老師：請敘述一個 C++ 程式開發的任務，是「用物件導向程式設計做會比用程序式程式設計」好很多的，並且說明原因。

- PCATS: One example where Object-Oriented Programming (OOP) in C++ can be more beneficial than Procedural Programming is in the development of a Graphical User Interface (GUI) for an application.
 - OOP allows for the creation of objects (like buttons, text boxes, etc.) that can be reused throughout the application, reducing code redundancy.
 - The encapsulation feature of OOP ensures that the data and methods within an object are safe from accidental alteration.
 - Inheritance allows new objects to be created from existing ones, promoting code reusability and efficiency.
 - Polymorphism allows objects to take on many forms depending on the situation, increasing flexibility. Can you think of how these features might be applied in a GUI application?
- 老師：請給我另一個例子，不要視窗，也不要繼承和多型
- PCATS: Sure, let's consider the development of a complex simulation system, such as a traffic simulation.
 - In a traffic simulation, each car, pedestrian, or traffic light can be represented as an object. Each object can have its own properties (like speed, location) and behaviors (like move, stop).
 - Using OOP, you can encapsulate these properties and behaviors within each object. This makes the code easier to understand and maintain, as each object's data and methods are grouped together.

Even without inheritance and polymorphism, the ability to create multiple instances of an object (like multiple cars or pedestrians) can greatly simplify the code and make it more efficient. Can you see how OOP would be beneficial in this scenario?

- 老師：這個很好！但可以告訴我這樣的模擬系統可能有什麼實務應用嗎？為什麼要開發一個模擬系統？
- PCATS: Traffic simulation systems have a wide range of practical applications:
 - Urban Planning: They can help city planners understand the impact of new roads, traffic lights, or other infrastructure before they are built.
 - Traffic Management: They can assist in optimizing traffic light timings to reduce congestion.
 - Emergency Planning: They can model the impact of road closures or other disruptions, helping authorities plan for emergencies.
 - Autonomous Vehicles: They can be used to test the behavior of autonomous vehicles in various traffic scenarios.

Developing a simulation system allows us to test and predict outcomes in a controlled environment before implementing changes in the real world, which can save time, resources, and potentially lives. Can you think of other potential applications?

- 老師：請把你的前兩個敘述合併起來。先說明我們現在要開發一個程式去最佳化某個路口的紅綠燈秒數，因此我們要開發一個交通模擬系統。接著說明為什麼這樣的一個系統適合用 OOP 的精神去開發，比用程序式的精神更好
- PCATS: We're tasked with developing a program to optimize traffic light timings at an intersection, for which we're creating a traffic simulation system. Here's why Object-Oriented Programming (OOP) is a suitable approach:
 - Encapsulation: In OOP, we can encapsulate properties (like speed, location) and behaviors (like move, stop) within each object (like car, pedestrian, traffic light). This makes the code easier to understand and maintain, as each object's data and methods are grouped together.
 - Multiple Instances: OOP allows us to create multiple instances of an object (like multiple cars or pedestrians), simplifying the code and making it more efficient.

In contrast, procedural programming might require us to keep track of each entity and their behaviors separately, which can become complex and difficult to manage as the system grows. Can you see how OOP would be beneficial in this scenario?

綜合以上，本題一個還可以的答案是：

- 假設今天我們要開發一個程式，用於優化交叉路口的紅綠燈時間長度。為此我們需要建立一個交通模擬系統，這個系統中會有一個十字路口，許多車和行人會隨機地出現並且有隨機的移動路線，但都會遵守紅綠燈的規範。這個系統要讓我們可以在指定一組紅綠燈秒數後，透過系統大量地模擬每輛車和每個行人的等待時間，進而計算出總平均等待時間，以便我們最終可以求取能最小化總平均等待時間的紅綠燈秒數。

物件導向程式設計（OOP）是一個適合的方法，這是因為：

1. 封裝：在 OOP 中，我們可以在每個物件（如汽車、行人、紅綠燈）內封裝屬性（如速度、位置）和行為（如移動、停止）。這使得程式碼更容易被理解和維護，因為每個物件的屬性和行為都被組織在一起。
2. 方便建立複數個物件實體：OOP 允許我們創建一個類別的多個物件實體（如多輛汽車或行人），再讓他們互動，這會方便我們維持程式碼的整潔與可讀性。

相較之下，在程序式程式設計中，每個實體的屬性都會獨立存在，我們要對一個實體做更新（例如更改一個行人的位置）時都得要去存取複數個彼此獨立的資料結構（例如陣列），所有讓實體發生行為的函數也會全部都是無法被結構化管理的全域函數（global function）。隨著系統的擴大，程式可能會變得複雜且難以管理。

大家可以發現這個解答是老師把 PCATS 提供的解答改一改之後得到的，所以 PCATS 確實幫上忙了。但大家應該也有發現老師確實有換句話說，並且按照自己的認知完成答案，這也是大家得要做的（善用生成式 AI，而不是完全依賴）。最後，上面的例子只是個簡單的範例，你的答案應該要更完整一點，包含任務敘述和 OOP 的好處。

你舉的例子和你列舉的原因可以包含目前沒教過的（例如繼承和多型），但你的敘述必須要正確並且讓我們相信你確實理解。與其寫你不確定的東西，我們建議大家還是寫你確定的就好了。