程式設計（112-1）
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 14 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詹景棠。

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

（50 分）接續作業七的第一題，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。另外，本次作業最主要的就是這一題（配分 50 分也是最高的），而不是第三題。

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、單位人力成本（與作業第一題不同）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、單位人力成本為 c_i^L、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。

每一項產品由數種零件組成，每種零件都會有其零件名稱（不重複且只包含英數和空白字元的字串），以及單位原料成本。我們稱呼零件 j 的零件名稱為 z_j、單位原料成本為 c_j^M，而產品 i 使用到的零件被放在 M_i 這個多重集合（multiset）裡[2]。一樣為了簡單起見，我們假設這些零件的成本都從未改變過。

有了以上零件的資訊，產品 i 的單位生產總成本（簡稱單位總成本）即為

$$c_i = c_i^L + \sum_{j \in M_i} c_j^M \text{。}$$

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，同時可能有 $m = 5$ 種零件，其基本資訊如表 2 所示，最後，4 種產品與 5 種零件之間的關係如表 3 所示，記錄每一種產品分別由哪些零件構成。請注意組成關係表內沒有排序，且一個產品可能用到一個零件複數次（像是 jPhone 41 Pro 產品與零件 Lens 分別在關係 7 與 13 都有出現，這代表著一支 jPhone 41 Pro 是由兩個 Lens 零件組成的）。給定上述資訊，表 4 中即為各產品的單位毛利（單位零售價減單位總成本），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）。

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、單位總成本、單位毛利（單位零售價減單位總

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

[2] 所謂多重集合，就是一個元素可以出現複數次的集合。詳情可以參閱 `https://en.wikipedia.org/wiki/Multiset`。

產品編號	產品名稱	單位零售價	單位人力成本	截至去年底的總銷售量
1	jPhone 41	200	100	1000
2	jPhone 41 Pro	250	90	800
3	WaterPods	40	5	1200
4	WaterPods Max	50	15	900

表 1: 範例產品資訊

零件編號	零件名稱	單位原料成本
1	Screen	5
2	Speaker	10
3	Battery	5
4	Motherboard	10
5	Lens	10

表 2: 範例零件資訊

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」、「jPhone 41」和「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「jPhone 41」，歷史利潤總和是 80000；請注意「jPhone 41」和「jPhone 41 Pro」的總營收相同，則因為 `strcmp` 把「jPhone 41」排在「jPhone 41 Pro」前面，所以「jPhone 41」是第一名。

在本題中，我們建議你建立以下類別：

```
class Item
{
private:
    char* name;
    int materialCost;
public:
    // n: name, mc: material cost
    Item(char* n, int mc);
    ~Item();
};

class Product
{
private:
    char* name;
    int price;
    int laborCost;
```

關係編號	產品名稱	零件名稱
1	jPhone 41	Screen
2	jPhone 41	Battery
3	jPhone 41 Pro	Screen
4	jPhone 41 Pro	Speaker
5	jPhone 41 Pro	Battery
6	jPhone 41 Pro	Motherboard
7	jPhone 41 Pro	Lens
8	jPhone 41	Motherboard
9	WaterPods	Screen
10	WaterPods	Speaker
11	WaterPods Max	Screen
12	WaterPods Max	Motherboard
13	jPhone 41 Pro	Lens

表 3: 範例產品與零件關係資訊

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 4: 範例歷史總資訊

```
    int salesQty;
    int itemCnt;
    Item** itemList;
public:
    Product(char* name, int price, int laborCost,
            int salesQty, int itemCnt);
    ~Product();
    bool isInFrontOf(const Product& prod, int criterion);
    void addItem(Item* itemPtr);
};
// you may define more members by yourselves
```

在 `Product` 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。另外，在 `Product` 的 constructor 中應該初始化 `itemList` 為一個存有 `itemCnt` 個 `Item*` 的動態陣列，並且將這些指向 `Item` 的指標都先指向 `nullptr`。對於 `Item` 和 `Product` 是否需要 copy constructor 和 assignment operator，則由你決定（如果不確定是否需要，那原則上建議就直接寫出來）。

接著請在 main function 裡面建立一個動態陣列（`productCnt` 是產品個數、`itemTotalCnt` 是所有零件的個數）：

```
int productCnt = 0, itemTotalCnt = 0, relationships = 0;
cin >> productCnt >> itemTotalCnt >> relationships;
Product** products = new Product*[productCnt];
Item** items = new Item*[itemTotalCnt];
```

每當讀入一個產品或零件的相關資訊，就使用 constructor 建立一個 `Product` 或 `Item` 物件（記得幫 `name` 動態地開一塊記憶體空間），並且把 `products` 以及 `items` 裡的一個指標指向該物件。當讀入一個產品與零件的組成關係時，就呼叫該產品的 `addItem()` 函數，把傳入的某 `Item` 物件的位址存入該產品的 `itemList` 中還是 `nullptr` 的元素中索引值最小的那個元素。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。

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 $n+m+u+1$ 行：

- 第一行包含三個整數，依序為 n、m、u，依序代表產品數量、零件數量，以及產品零件關係數量；
- 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）、單位人力成本 c_i^L（整數）、截至去年底的總銷售量 q_i（整數），以及組成該產品的零件數量（若某產品要用到 1 個零件 A 和 2 個零件 B，則其零件數量為 3 而非 2）；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。
- 第 $n+2$ 行起的 m 行中，第 $n+1+j$ 行包含零件 j 的零件名稱 z_j（只含有英數和空白字元的字串，且頭尾一定不會是空白）、單位原料成本 c_j^M（整數）；每一行的 z_j 和 c_j^M 之間被一個逗點隔開。
- 第 $n+m+2$ 行起的 u 行中，第 $n+m+1+k$ 行包含關係 k 的產品名稱 x_k、零件名稱 z_k；兩兩之間被一個逗點隔開。
- 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 $1 \le k \le n \le 100$、$1 \le m \le 1000$、$n \le u \le 10000$、$y \in \{1,2,...,6\}$、x_i 的長度介於 1 和 100 之間（包含 1 和 100）、z_j 的長度介於 1 和 100 之間（包含 1 和 100）、$1 \le |M_i| \le 2m$（$|M_i|$ 是產品 i 使用的零件數，這裡是表示一個產品用到的零件數最多是 $2m$ 個）、$0 \le c_j^M \le 50$、$1 \le \sum_{j \in M_i} c_j^M + c_i^L \le p_i \le 1000$、$0 \le q_i \le 10000$、任兩個 x_i 或 z_j 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 5 13
jPhone 41,200 100 1000 3
jPhone 41 Pro,250 90 800 6
WaterPods,40 5 1200 2
WaterPods Max,50 15 900 2
Screen,5
Speaker,10
Great Battery,5
Motherboard,10
Lens,10
jPhone 41,Screen
jPhone 41,Great Battery
jPhone 41 Pro,Screen
jPhone 41 Pro,Speaker
jPhone 41 Pro,Great Battery
jPhone 41 Pro,Motherboard
jPhone 41 Pro,Lens
jPhone 41,Motherboard
WaterPods,Screen
WaterPods,Speaker
WaterPods Max,Screen
WaterPods Max,Motherboard
jPhone 41 Pro,Lens
4 3
```

則輸出應該是

```
122000
```

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自定義資料型態、類別、運算子多載，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`、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！

第二題

（20 分）有兩個類別 `Point` 和 `Polygon`，是二維平面上的格子點和以格子點為端點的多邊形：

```
#include<iostream>
using namespace std;

class Point
{
private:
    int x;
    int y;
public:
    Point(int x, int y) : x(x), y(y) {}
friend class Polygon;
};

class Polygon
{
private:
    int pointCnt;
    Point** endPoints;
public:
    Polygon();
    Polygon(int pc, int x[], int y[]);
    Polygon(const Polygon& p);
    ~Polygon();
    void print();
    void movePoint(int index, int x, int y);
};

Polygon::Polygon()
{
    this->pointCnt = 0;
    this->endPoints = nullptr;
```

```
}

Polygon::Polygon(int pc, int x[], int y[])
{
    this->pointCnt = pc;
    this->endPoints = new Point*[pc];
    for(int i = 0; i < pc; i++)
        endPoints[i] = new Point(x[i], y[i]);
}

Polygon::~Polygon()
{
    for(int i = 0; i < this->pointCnt; i++)
    {
        delete endPoints[i];
        endPoints[i] = nullptr;
    }
    delete [] endPoints;
    endPoints = nullptr;
}

void Polygon::print()
{
    cout << "(";
    for(int i = 0; i < this->pointCnt; i++)
    {
        cout << "(" << this->endPoints[i]->x
             << ", " << this->endPoints[i]->y << ")";
    }
    cout << ")";
}

void Polygon::movePoint(int index, int x, int y)
{
    this->endPoints[index]->x = x;
    this->endPoints[index]->y = y;
}
```

在本題中，我們會以如下的 main function 使用上述兩個類別：

```
int main()
{
    int pointCnt = 0;
    cin >> pointCnt;
```

```
    int* x = new int[pointCnt];
    int* y = new int[pointCnt];
    for(int i = 0; i < pointCnt; i++)
        cin >> x[i];
    for(int i = 0; i < pointCnt; i++)
        cin >> y[i];

    Polygon p1(pointCnt, x, y);
    Polygon p2(p1);
    p2.movePoint(0, 0, 0);
    p1.print();
    cout << "\n";
    p2.print();
    delete[] x;
    x = nullptr;
    delete[] y;
    y = nullptr;
    return 0;
}
```

如大家所見，Polygon 的定義缺了 copy constructor。在本題中，我們就要請你實作這個函數。請只上傳這個函數，PDOGS 會自動把你上傳的函數跟已經在 PDOGS 上的程式（如附件的 HW08_p2.cpp）拼起來去編譯。換言之，在本題你被迫必須要實作本題指定的內容；如果你上傳了任何帶有其他部分的程式，你會無法得到分數的！

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。在每個檔案中會有三行，第一行含有一個整數 n；第二行含有 n 個整數，依序是 n 個格子點的 x 座標；第二行含有 n 個整數，依序是 n 個格子點的 y 座標。已知 $1 \leq n \leq 20$、每個座標值都介於 -1000 和 1000 之間（包含兩端點）。

讀入這些資訊後，程式應該執行前述 main function 並且輸出執行結果。舉例來說，如果輸入是

```
4
1 4 5 3
2 6 9 5
```

根據以上的寫好程式碼為例，則輸出應該是

```
((1, 2)(4, 6)(5, 9)(3, 5))
((0, 0)(4, 6)(5, 9)(3, 5))
```

請注意如果沒有好好實作正確的 constructor，那多半就會 run-time error 和（或）印出不對的結果。

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

（20 分）承上題，現在我們要請你幫 `Polygon` 建立幾個被多載（overload）的運算子（operator）：

```
class Polygon
{
private:
    int pointCnt;
    Point** endPoints;
public:
    Polygon();
    Polygon(int pc, int x[], int y[]);
    ~Polygon();
    void print();
    void movePoint(int index, int x, int y);
    bool operator==(const Polygon& p);
    Point operator[](int index);
    const Polygon& operator=(const Polygon& p);
};

// those functions defined in Problem 2
```

其中 `==` 不修改 invoking object，在兩個 `Polygon` 物件一模一樣（點的個數、位置、順序都一模一樣）時回傳 true，反之則回傳 false；`[]` 不修改 invoking object，單純回傳該 `Polygon` 物件的 `endPoints` 中索引值為 `index` 的那個 `Point` 物件的複本（而非參照），且你可以假設傳入的 `index` 一定介於 0 和 `pointCnt - 1` 之間（包含這兩個邊界值）。

為了測試你透過 `[]` 回傳的 `Point` 物件，我們將幫 `Point` 類別新增一個成員函數：

```
class Point
{
private:
    int x;
    int y;
```

```
public:
    Point(int x, int y) : x(x), y(y) {}
    void print();
friend class Polygon;
};

void Point::print()
{
    cout << "(" << this->x << ", " << this->y << ")";
}
```

在本題中，我們會以如下的 main function 使用上述兩個類別：

```
int main()
{
    int pointCnt = 0;
    cin >> pointCnt;
    int* x = new int[pointCnt];
    int* y = new int[pointCnt];
    for(int i = 0; i < pointCnt; i++)
        cin >> x[i];
    for(int i = 0; i < pointCnt; i++)
        cin >> y[i];

    Polygon p1(pointCnt, x, y);
    Polygon p2;
    p2 = p1;
    if(p2 == p1)
        cout << "Identical\n";
    else
        cout << "Different\n";
    p2[0].print();
    cout << "\n";
    p2.movePoint(0, 0, 0);
    p1[0].print();
    cout << "\n";
    if(p2 == p1)
        cout << "Identical\n";
    else
        cout << "Different\n";
    delete[] x;
    x = nullptr;
    delete[] y;
    y = nullptr;
```

```
    return 0;
}
```

在本題中，我們要請你實作 `Polygon` 的那三個被多載的運算子 `==`、`[]` 和 `=`。請只上傳這三個運算子的定義，PDOGS 會自動把你上傳的程式碼片段跟已經在 PDOGS 上的程式（如附件的 HW08_p3.cpp）拼起來去編譯。換言之，在本題你被迫必須要實作本題指定的內容；如果你上傳了任何帶有其他部分的程式，你會無法得到分數的！

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。在每個檔案中會有三行，第一行含有一個整數 n；第二行含有 n 個整數，依序是 n 個格子點的 x 座標；第二行含有 n 個整數，依序是 n 個格子點的 y 座標。已知 $1 \le n \le 20$、每個座標值都介於 -1000 和 1000 之間（包含兩端點）。

讀入這些資訊後，程式應該執行前述 main function 並且輸出執行結果。舉例來說，如果輸入是

```
4
1 4 5 3
2 6 9 5
```

根據以上的寫好程式碼為例，則輸出應該是

```
Identical
(1, 2)
(1, 2)
Different
```

請注意如果沒有正確地多載的那三個運算子，那多半就會 run-time error 和（或）印出不對的結果。

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）請把你為第一題寫的 `Item` 和 `Product` 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，請把這兩個類別的宣告（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`。